

Axon.ivy 7.2

Engine Guide

Axon.ivy 7.2: Engine Guide

Publication date 13.11.2018

Copyright © 2008-2018 AXON Ivy AG

1. Introduction	1
What is Axon.ivy	1
About this guide	2
Installation Environment	2
Engine Types	3
2. Getting Started	6
Introduction	6
Windows (with UI tools)	6
Debian Linux	18
Linux (with console tools only)	24
Docker	31
3. Installation	33
Upgrading from an older version	33
Standard Edition Installation	34
Enterprise Edition Installation	35
Install Axon.ivy Engine	36
System Database	39
4. Configuration	49
Engine Configuration	49
Configuration File Reference	52
5. Security	77
General	77
Front-end Server	77
HTTPS	80
Axon.ivy Engine	80
6. Integration	82
Introduction	82
Apache Integration	83
Microsoft IIS Integration	85
Axon.ivy Cluster Integration	99
Web Application Firewall	101
7. Administration	104
Deployment	104
Standard Processes	107
Miscellaneous	112
8. Monitoring	114
Logging	114
Process Element Performance Statistic and Analysis	116
Java Management Extensions (JMX)	118
VisualVM	121
9. Tool Reference	124
AxonIvyEngine	124
EngineConfigCli	124
Engine Service	125
Launch Configuration	126
Engine Configuration UI	128
Admin UI	135
Control Center	166
10. Troubleshooting	170
Troubleshooting	170

Chapter 1. Introduction

What is Axon.ivy

Axon.ivy is a *Digital Business Platform* that simplifies and automates the interaction of humans with their digital systems. The platform is typically in charge of the most precious business cases where companies produce value. Here is how we do it:

1. **Visualize:** Our platform allows you to **document business processes fast and intuitive**. A shared view on users, roles, departments and technical systems that are involved in a business process improves your work. HR recruitment profiles become clearer, bottle necks become obvious, ideas for effective improvements arise by anyone who is involved in the process.
2. **Automate:** Documented processes are good. But what you really want is to **drive your highly valuable processes automatically**. Often the daily work of employees is interrupted by searching and filtering data from various tools and by feeding this data into other technical systems. Even though value is produced in a well-known business case, there is a lack of a clear interface which guides the involved users through the process. Highly valuable data is often divided and stored in various dedicated technical systems. With Axon.ivy you can drive your process automatically. People, data and technical systems can easily be orchestrated by our platform. An initial application that leads users through the process can be generated without the need to hire a software engineer. People can contribute to the process by using their favourite device such as a smartphone or workstation.
3. **Improve:** The digitalization of your company can **evolve over time**, we favour small predictable improvements over big bang solutions. The Axon.ivy Digital Business Platform allows you to start simple and fast with your existing environment. You may start with just task notifications that are sent to users that should contribute to a running process. And eventually the platform becomes your single interface for all your business interactions. You will be able to measure KPIs based on the highly valuable data that is produced during the execution of your business processes. Based on these insights, you can advance your business constantly and effectively. The cost of business transformations become reasonable and predictable.

The Axon.ivy Digital Business Platform consists of:

- The *Axon.ivy Designer* - where you draw, simulate and implement automated business processes.
- The *Axon.ivy Engine* - an application server that executes your business cases and provides a shared interface for process users.

Why Axon.ivy?

Axon.ivy is exciting for everyone that partakes on your digital transformation journey.

- **Business:** We enable you to start your personal digital transformation journey and make new business opportunities possible. You are still the captain of your ship, start with simple automations and transform essential parts of your business when you gain trust and confidence.
- **Business Analysts:** It has never been easier to document processes fast and intuitive. The process simulation allows you to verify that you have a shared view how processes should be executed. Setup a simple structure for the data of a processes and you even get a simple executable application with generated front ends that are meaningful. No software engineer is required to create an already powerful application from scratch.
- **Developers:** Develop your application on a rich stack of Java frameworks that withstood the test of time. We minimize your technology evaluation effort by giving you a set of libraries and an IDE that match perfectly together. This allows you to quickly jump into projects and deliver value. While you always have the ability to break out of our predefined tooling and use advanced features of Tomcat, JPA, JSF, JAX-RS or whatever you require.
- **Operations:** We deliver packages for popular platforms (Linux, Windows). No big change, we orchestrate your existing systems. We support many DB vendors (Oracle, Microsoft SQL Server, MySQL, PostgreSQL). Effective monitoring and logging interfaces are provided to give you a safety that the application is healthy and accessible.

About this guide

You are now reading Axon.ivy Engine documentation.

In case you want to know more on

- Getting the latest Axon.ivy version: Go to <https://developer.axonivy.com/download/>
- System requirements: Please read *Readme.html* in the installation directory.
- Working with Axon.ivy Engine: Start with the Getting Started chapter.
- Demo projects: The Axon.ivy Designer ships with several demo projects, which can be deployed to the Axon.ivy Engine.
- How to draw, simulate and implement automated business processes: Please read the Designer Guide (in the installation folder of an Axon.ivy Designer)
- Upgrading an existing installation: Please read *MigrationNotes.html* (located in the installation folder).

All above mentioned documentations are brief and tend to describe only necessary functionality. We highly encourage reading these documentations to speed up your development, to get to know new features or to eliminate potential problems.

Installation Environment

The following diagram shows the installation environment of an Axon.ivy Engine:

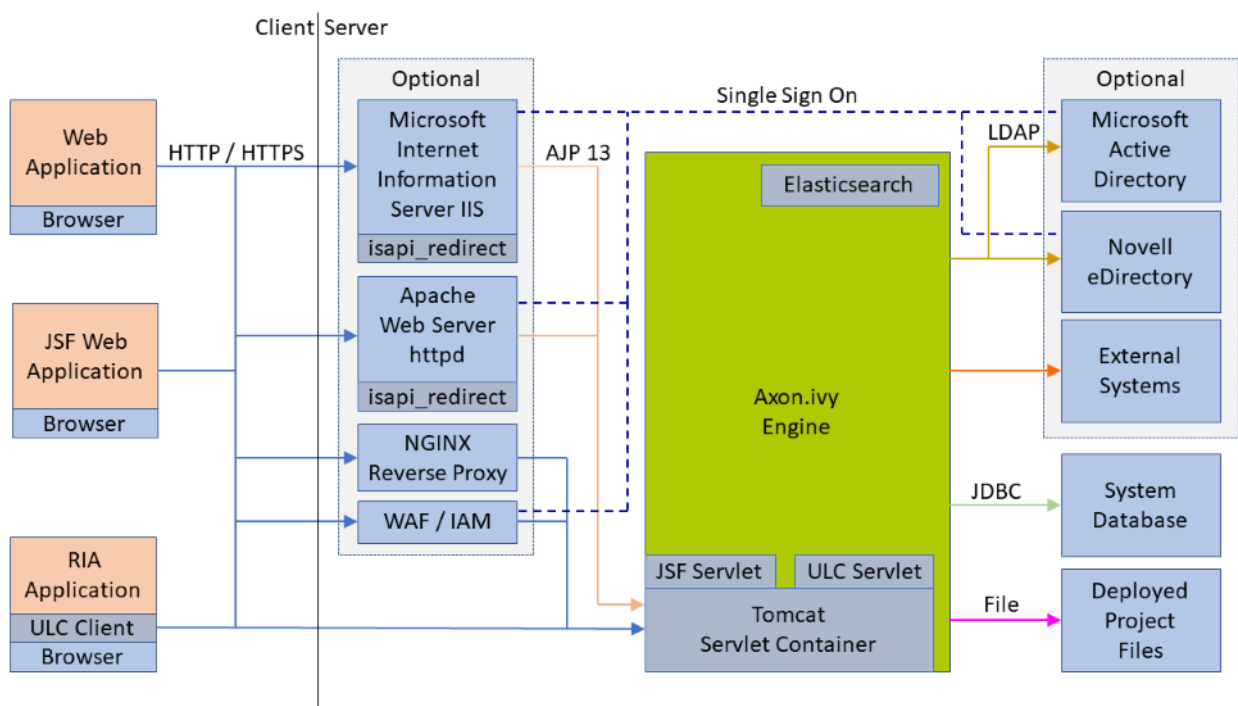


Figure 1.1. Axon.ivy Engine Installation

The Axon.ivy Engine needs a system database to store its configuration, users, roles and assigned permissions and the states, cases, tasks from the deployed applications. Next, it needs file directories where the deployed projects are stored. The Axon.ivy Engine integrates a Tomcat Servlet Engine that is responsible to receive HTTP or HTTPS requests from client applications

and to send back appropriate responses (similar to a web server). The client applications itself are either Web Applications that run in a web browser or Rich Internet Applications that run in a Java Virtual Machine (JVM). Both kind of client applications communicate over HTTP or HTTPS directly with the Servlet Engine. In a productive environment, normally a Microsoft Internet Information Server (IIS), Apache Web Server, NGINX reverse proxy or a web application firewall (WAF) often combined with an identity and access management (IAM) system is put in front of the Axon.ivy Engine. The front-end servers are then responsible to forward the requests to the Axon.ivy Engine Servlet Container. Also, the users are imported from an external security system like a Microsoft Active Directory or Novell eDirectory. Axon.ivy applications can integrate with third party external systems like databases, web services or application servers. Axon.ivy Engine also integrates an Elasticsearch server. Instead using the integrated Elasticsearch server also an external Elasticsearch server can be used..



Tip

It is good practice to separate the data directory where you store the deployed project and other data files from the Engine installation directory. This will later simplify a migration to newer Engine versions.

Example:

Path	Description
.../AxonIvyEngine/Data/Applications/HRM/...	Directory where the deployed projects for the HRM application are stored.
.../AxonIvyEngine/Data/Applications/FinTech/...	Directory where the deployed projects for the FinTech application are stored.
.../AxonIvyEngine/Data/ElasticSearch/...	Directory where the Business Data search indexes are stored.
.../AxonIvyEngine/6.0.1/...	Installation directory of Axon.ivy Engine version 6.0.1 (can be removed if no longer needed)
.../AxonIvyEngine/6.1.0/...	Installation directory of Axon.ivy Engine version 6.1.0 (can be removed if no longer needed)
.../AxonIvyEngine/6.2.0/...	Installation directory of Axon.ivy Engine version 6.2.0

Table 1.1. How to organize data and installation directory

Engine Types

There are two different Axon.ivy Engine types:

- Standard Edition (formerly known as "Standard ")
- Enterprise Edition (Clustered Engine, formerly known as "Cluster ")

Axon.ivy Engine Standard Edition

The Axon.ivy Engine Standard Edition is installed on a single machine. A DBMS that can hold the Axon.ivy system database is the only special infrastructure it needs. The deployed projects can be stored on a local harddisk on same machine that the Axon.ivy Engine Standard Edition is running on.

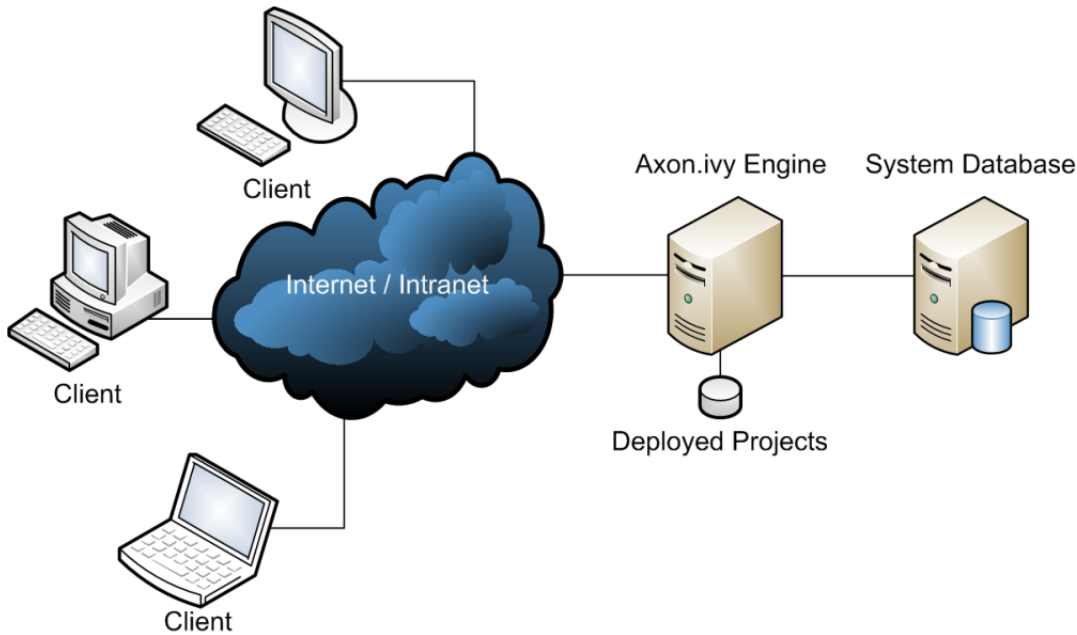


Figure 1.2. Axon.ivy Engine Standard Edition

Axon.ivy Engine Enterprise Edition

The Axon.ivy Engine Enterprise Edition is a cluster of multiple Axon.ivy Engine instances. It is built on a load balancer that receives requests from the clients and forwards them to multiple Axon.ivy Engine nodes typically running on different machines. The different nodes of an Axon.ivy Engine Enterprise Edition all share the same system database which is normally stored on a dedicated database. The deployed projects are stored on a file system that can be accessed by all nodes.

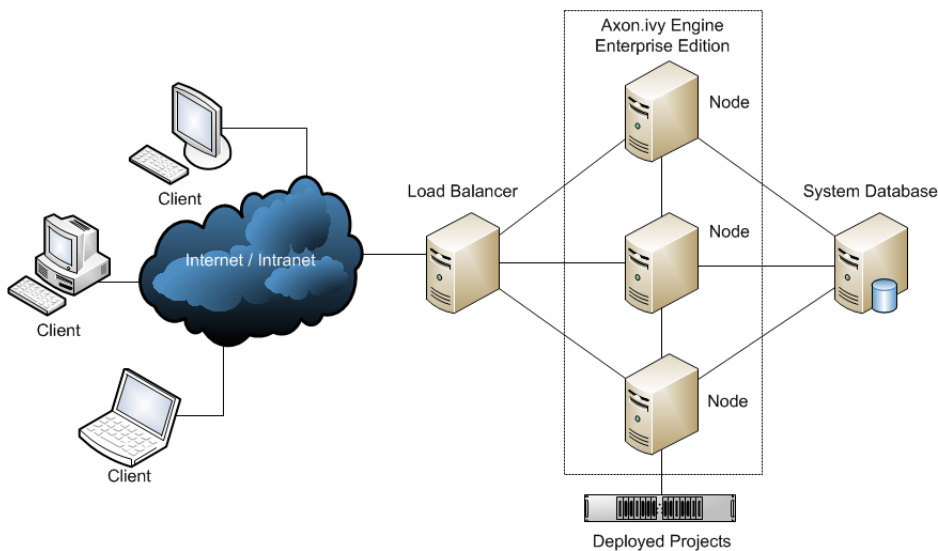


Figure 1.3. Axon.ivy Engine Enterprise Edition

Axon.ivy Engine Nodes are typically installed on multiple server machines, but it is also possible to install more than one Axon.ivy Engine Node on a single server machine. The load balancer can be realized either by a hardware load balancer or by an IIS or Apache web server that distributes the incoming requests to the installed Axon.ivy Engine Nodes.

What engine edition do I need?

The Axon.ivy Engine Enterprise Edition has two major *advantages* compared to the Standard Edition:

- **Performance and scalability:** An Axon.ivy Engine Enterprise Edition can serve more clients than the Axon.ivy Engine Standard Edition. If your number of clients increases, you can add another Engine node to your Axon.ivy Engine cluster. If you have a lot of sessions it may even be better to have two Axon.ivy Engine nodes on the same server machine instead of having a single Standard Edition. Because each session needs memory on the engine and Axon.ivy can handle two processes with medium memory footprints (i.e. Engine nodes) faster than one process with a large memory footprint (i.e. Standard Edition).
- **High availability:** In an Axon.ivy Engine Enterprise Edition installation, a single node may crash without affecting the other nodes, which still serve clients. However, if you require high availability of your Axon.ivy Engine you also need to ensure that all other components the engine is depending on (Load Balancer, Database Server, File Share) have a high availability.

The *disadvantages* of the Axon.ivy Engine Enterprise Edition compared to the Standard Edition are:

- higher complexity of the system
- higher hardware costs
- higher licence fees

Chapter 2. Getting Started

Introduction

This chapter helps you getting started with the Axon.ivy Engine. You will learn how to install and configure the Engine and finally, how to deploy your Axon.ivy projects. If you are a Windows user and used to install and configure software using the UI then the first section Windows is a good starting point for you. If you are an experienced Linux user and you are used to install and configure software using the console then the second section Linux is the right starting point for you. Note, that on both systems, Windows and Linux, Axon.ivy Engine can be installed and configured using the UI or the console. The Axon.ivy engine also runs in a container environment like Docker.

Windows (with UI tools)

Download the Engine

Open a web browser and navigate to <https://developer.axonivy.com>. Press on the **Download** button to navigate to the download page. Press the **Download Axon.ivy Engine** button:

AXON iVY
digitalize your business

Download Features Documentation Tutorial Q & A Team

Digital Business Platform AddOns Community Archive Sprint Release Nightly

⬇️ Axon.ivy Digital Business Platform

Welcome to the download area of Axon.ivy Digital Business Platform. We are excited to see you here!

Download Axon.ivy Designer
Leading Edge 7.0.0 / Windows x64
[Other versions/platforms](#)

Download Axon.ivy Engine
Leading Edge 7.0.0 / Windows x64
[Other versions/platforms](#)

The **Axon.ivy Designer** is a state of the art process modelling tool that enables you to turn your business processes into real and fully functional web applications. You can easily integrate your existing systems using standard connectors such as web services, SQL and many more.

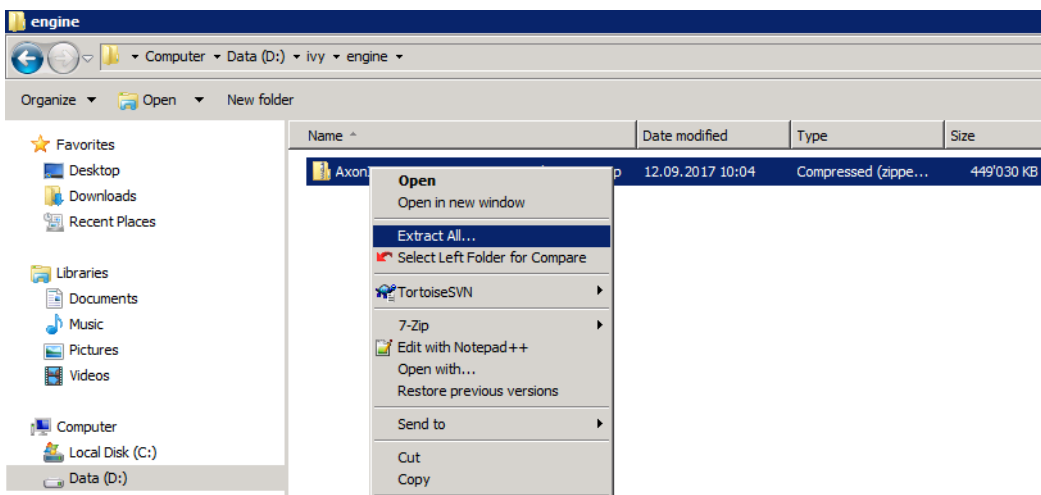
With the **Axon.ivy Engine** you can run your designed business processes on a single or clustered environment. Please order an evaluation licenses and further information at ivy@axonivy.com by providing your contact details.

Save the *AxonIvyEngine*.zip* file to your temporary download folder.

Install the Engine

We suggest that you install Axon.ivy Engine into a new folder called *ivy\engine* on one of your drives (e.g. *c:\ivy\engine*). To do so open a Windows Explorer and navigate to *C:* and create those new folders. Then, navigate to your temporary download folder and copy the file *AxonIvyEngine*.zip* to the newly created folder.

Right click the *AxonIvyEngine*.zip* file and press **Extract All ...** from the context menu.



On the appearing dialog press the **Extract** button. After the *AxonIvyEngine*.zip* is extracted navigate into the new *AxonIvyEngine** folder. The content of the installation folder looks like this:

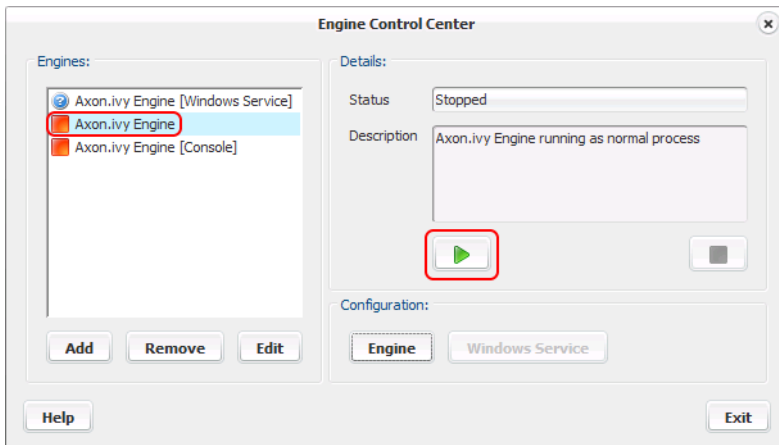
Name ^	Date modified	Type	Size
applications	14.09.2017 10:08	File folder	
bin	14.09.2017 10:10	File folder	
clientlib	14.09.2017 10:10	File folder	
configuration	14.09.2017 10:11	File folder	
doc	14.09.2017 10:11	File folder	
dropins	14.09.2017 10:25	File folder	
elasticsearch	14.09.2017 10:12	File folder	
jre	14.09.2017 10:24	File folder	
misc	14.09.2017 10:13	File folder	
projects	14.09.2017 10:13	File folder	
system	14.09.2017 10:24	File folder	
webapps	14.09.2017 10:14	File folder	
MigrationNotes.html	14.09.2017 10:25	Firefox HTML Docu...	8 KB
NewAndNoteworthy.html	14.09.2017 10:25	Firefox HTML Docu...	4 KB
ReadMe.html	14.09.2017 10:25	Firefox HTML Docu...	84 KB
ReleaseNotes.txt	14.09.2017 10:25	Text Document	212 KB

The most important sub folders in the Axon.ivy Engine installation folder are:

- the *bin* folder which contains all the executables
- the *configuration* folder which contains the configuration files.

Start the Engine

Navigate to the *bin* folder and double click on the *ControlCenter.exe* file to start the Control Center. You can use this tool to start and stop the Axon.ivy Engine in different ways (as Windows Service, as normal user process, with a console window). Select the **Axon.ivy Engine** and press the green play button to start the Axon.ivy Engine as a normal user process:

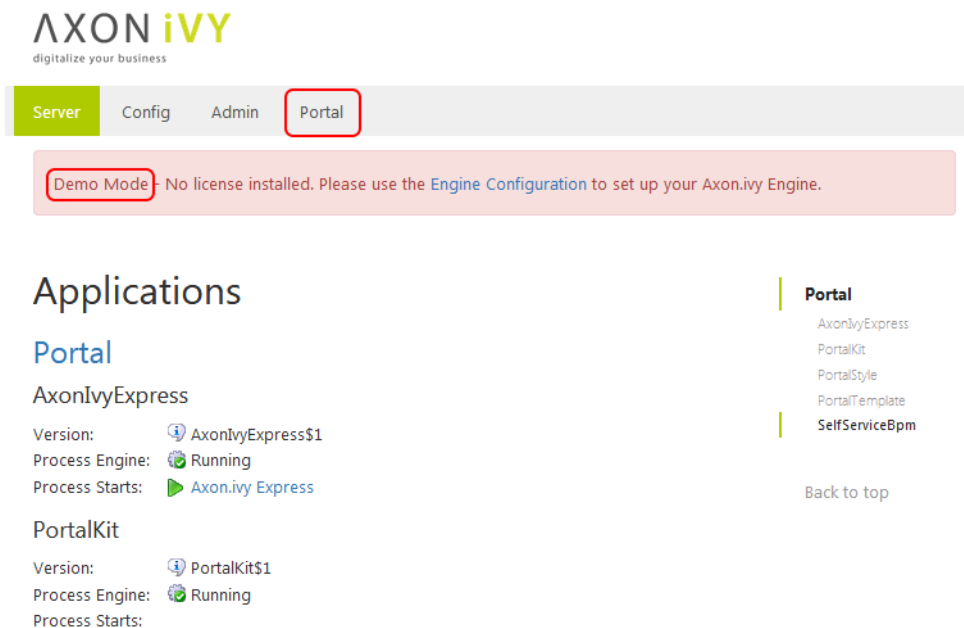


You can use the Control Center also to register Axon.ivy Engine as Windows Service. Moreover, you can add other existing Windows Services to the list of Engines to start and stop them with the Control Center. For example, if you have installed your database server on the same machine you can add the Windows Service of the database server.

After the Axon.ivy Engine has started a web browser is opened and the main page of the Axon.ivy Engine is displayed.

Use the Engine

The main page of the Axon.ivy Engine looks like this:



The Axon.ivy Engine is running in Demo Mode. This is because you did not install a valid license yet nor did you configure a system database. Note, that everything that you do with the Axon.ivy Engine running in Demo Mode is lost when you shut down the engine. However, you can use the engine also in Demo Mode and tryout the pre-installed Portal application by clicking on the **Portal Home** link. To login use one of the predefined demo users: **demo**, **guest** or **admin**. The passwords of the demo users are equal to the user names (E.g. **demo** for the **demo** user). Login as **demo** user and try to create a **TODO** task for the guest user using the **Axon.ivy Selfservice** process:

The screenshot shows the AXON iVY dashboard. At the top, the logo "AXON iVY" is displayed with the tagline "digitalize your business". A search bar on the right contains "demo". The main area is divided into two columns. The left column, titled "Processes", has a link "Show all processes" and a list of processes: "Axon.ivy Express" and "Axon.ivy Selfservice". The "Axon.ivy Selfservice" item is highlighted with a red box. Below the list are links for "Add new process" and "Delete processes". The right column, titled "Tasks", has a link "Show full task list" and a search bar for "Name or ID". It also includes sorting options "Sort by priority" and "Sort by expiry time", and a message "No tasks.".

Configure the **TODO** task as follows:

The screenshot shows the configuration screen for a "Self Service Ad-hoc Workflow" (#159). The page has tabs for "Request" and "Case Information". The main heading is "Self Service Ad-hoc Workflow" with a sub-heading "Define and launch a Workflow at your fingertip: Select a flow pattern, add some tasks and start the Workflow." The "Case" section contains a "Subject" field with the value "Setup Axon.ivy Engine" (highlighted with a red box) and a "Description" field with the value "Please setup a new Axon.ivy Engine on our prod server!" (highlighted with a red box). The "Pattern" section has radio buttons for "ToDo", "Approval", "Question-Answer", and "Ad-hoc", with "ToDo" selected. A "Details" link is also present. The "Workflow" section shows a sequence of steps: a "CREATOR" (demo, 15.09.2017 11:51) followed by a "TODO" task assigned to "guest" (18.09.2017 11:51). A "Task" button with a plus sign is visible. At the bottom, there is a "Start Workflow" button with a checkmark.

Start the workflow by pressing the **Start Workflow** button. After that, logout, then login again as **guest** user. On the task list, you now see the new **TODO** task that you created before. Try to work on the task by clicking on the arrow located on the right side of the task:

The screenshot displays the Axon.ivy dashboard for a guest user. The top navigation bar includes the Axon.ivy logo and the text 'digitalize your business'. The main content area is divided into three sections: 'Processes', 'Tasks', and 'Statistics'. The 'Processes' section lists 'Axon.ivy Express' and 'Axon.ivy Selfservice'. The 'Tasks' section shows a search bar and a list of tasks. One task, 'TODO Setup Axon.ivy Engine', is highlighted with a red box around its right arrow. The 'Statistics' section shows a pie chart for 'Task priority' with 100% 'Normal'.

Go ahead and play around with the Selfservice process. Try out different types of tasks.

In Axon.ivy a task is a piece of work (a part of a process) that is assigned to a user or role. A user to whom a task is assigned or who has the role to which a task is assigned can work on the task. When a user works on a task the task disappears from the task list of other users who might also be able to work on the task. This means only one user can really work on a task at the same time. In a process, it is possible to define parallel tasks. Therefore, it is possible that two or more users work parallel on different tasks of the same process instance. In Axon.ivy a process instance is called a case.

Configure the Engine

Now, let's configure the Axon.ivy Engine with a license and system database.

To start with that you must first request a valid Axon.ivy Engine license. Either you get a license for your productive system through one of our sales personal or contact our support for time limited tryout licenses. If you do not have a license you can skip this section and continue with the next section.

Moreover, you need to have a supported database server up and running with a database user that has the rights to create new databases. The configuration and creation of the system database differs a little bit depending on the database system you use. We will use a PostgreSQL database server.

You can start the configuration on the main page of the Axon.ivy Engine (e.g. <http://localhost:8080/ivy>) by clicking on the **Config** menu.

On the first page use the **Upload Licence** button to install your license file.

AXON iVY digitalize your business Engine Configuration [Help](#)

▼ Licence

No licence installed. Please upload a valid licence.

+ Upload Licence Next

> System Database

> Administrators

> Web Server

> Summary

Save Discard Changes

Powered by Axon.ivy Engine 7.0.0.55921. Copyright © 2001 - 2017 AXON IVY AG

Some information of your license will be displayed on the page as soon as you have installed it. Press the **Next** button to continue.

On the next screen choose the correct **Database** and **Driver** (in our case **PostgreSQL**). Configure the **Host** and **Port** where your database server is running and listening. Configure the **Username** and **Password** of a database user that has the right to create a new database on the database server.

▼ System Database

Connection state unknown Check Connection
Please check the connection to the Database.

Database PostgreSQL

Driver PostgreSQL

Host localhost

Port 5432 Default

Database Name AxonIvySystemDatabase

Username admin

Password

Additional Properties

Create Database Convert Database Next

Press the **Create Database** button. On the appearing dialog configure the name of the Axon.ivy system database. Press the **Create Database** button to create the system database.

As soon as the database creation is finished the following dialog appears:

Press the **Save and connect** button to save the configuration and connect to the newly created system database.

The system database is used by the Axon.ivy Engine to store configurations, users, roles, process instances, tasks and process data.

Press the **Next** button.

On the next page, you can configure system administrators.

Fill in the form and press the **Add Administrator** button.

Administrators can administrate the Axon.ivy Engine. For example, they can add or remove users, assign user to roles, deploy projects, etc.

Therefore, you need at least one administrator to administrate the Axon.ivy Engine.

The Email addresses of administrators are used to send mail notifications if license problems occur.

Press the **Next** button.

On the next page configure which protocol connectors and ports the Axon.ivy Engine internal web server should provide. You do not need the AJP protocol. So let's disable it.

Web Server

HTTP Connector HTTPS Connector AJP Connector

Enabled: Enabled: Enabled:

Port: 8080 Port: 8443 Port: 8009

Advanced Next

The Axon.ivy Engine internal web server provides connectors for the following protocols:

- HTTP** Protocol used by web browser to communicate with a web server. This protocol is not secure since the communication is not encrypted. Axon.ivy Engine uses port 8080 by default.
- HTTPS** Like HTTP but secure. It uses TLS to encrypt the communication between the web browser and server. Axon.ivy Engine uses port 8443 by default.
- AJP** This protocol is used to communicate with a Microsoft IIS or Apache httpd front-end server. AJP means Apache Jakarta Protocol. Axon.ivy Engine uses port 8009 by default.

Press the **NEXT** button.

On the next page, the configuration is summarized.

Summary

Licence Type
Axon.ivy Engine Demo

Expiry Date
Never

Database Product
PostgreSQL

Database Host
ZugTstDbsPos

Database Name
AxonIvySystemDatabase

Database User
admin

Database Connection Url
jdbc:postgresql://ZugTstDbsPos:5432/AxonIvySystemDatabase

Administrators
admin

Save Discard Changes

Press the **Save** button to save the configuration. Switch back to the **Control Center** and restart the Axon.ivy Engine by stopping and starting it again.

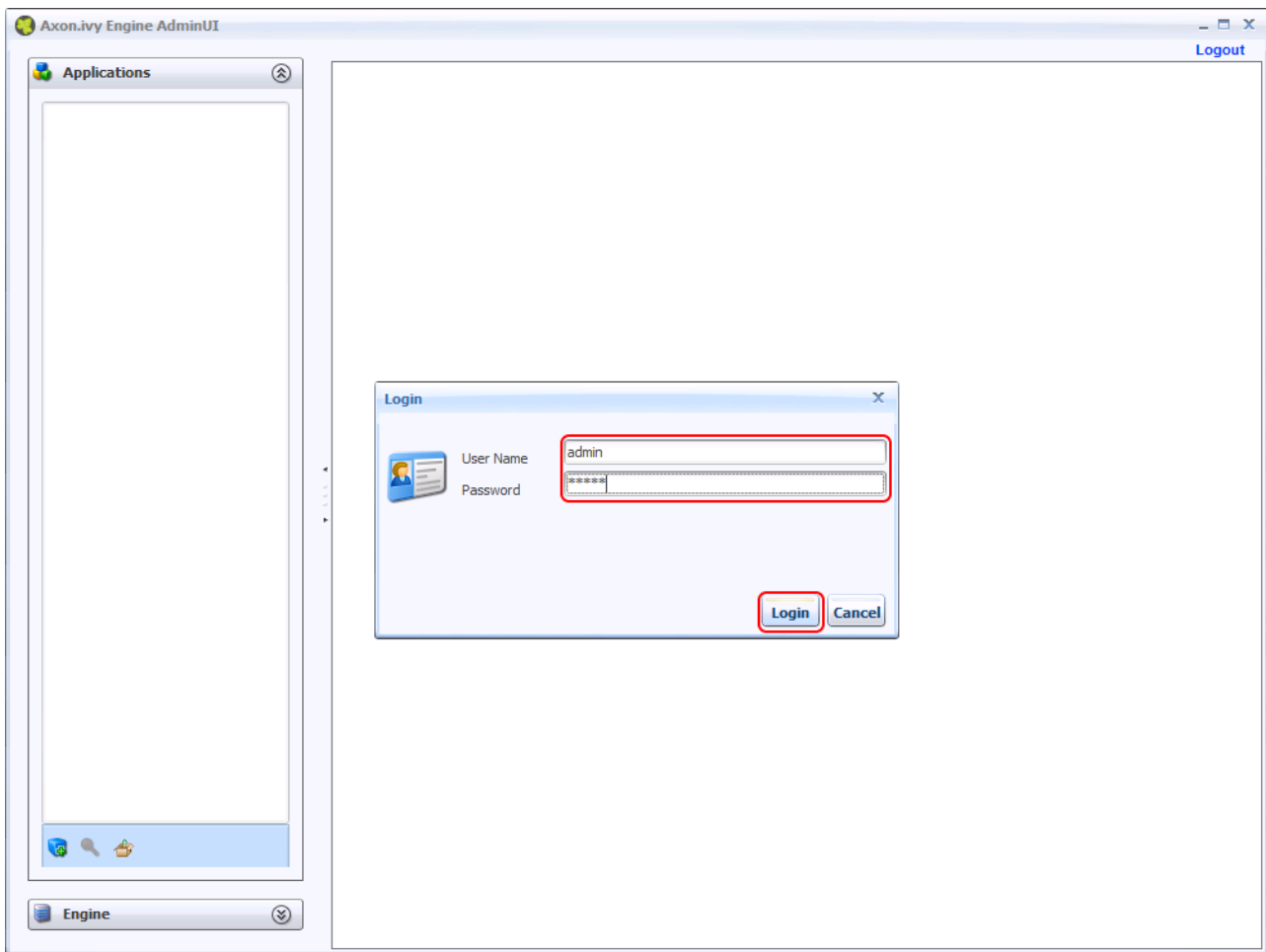
Note, that the HTTP port of the Axon.ivy Engine may have changed. If you did change the HTTP settings. So open again a web browser and navigate to `http://localhost:8080/ivy`. Have you seen that the header with the demo mode message is gone? You now have a production ready Axon.ivy Engine.

The **Config** menu is only available in demo mode. If you want to reconfigure an Axon.ivy Engine not running in demo mode use the **AxonIvyEngineConfig.exe** tool in the **bin** folder instead.

Deploy an Axon.ivy project to the Engine

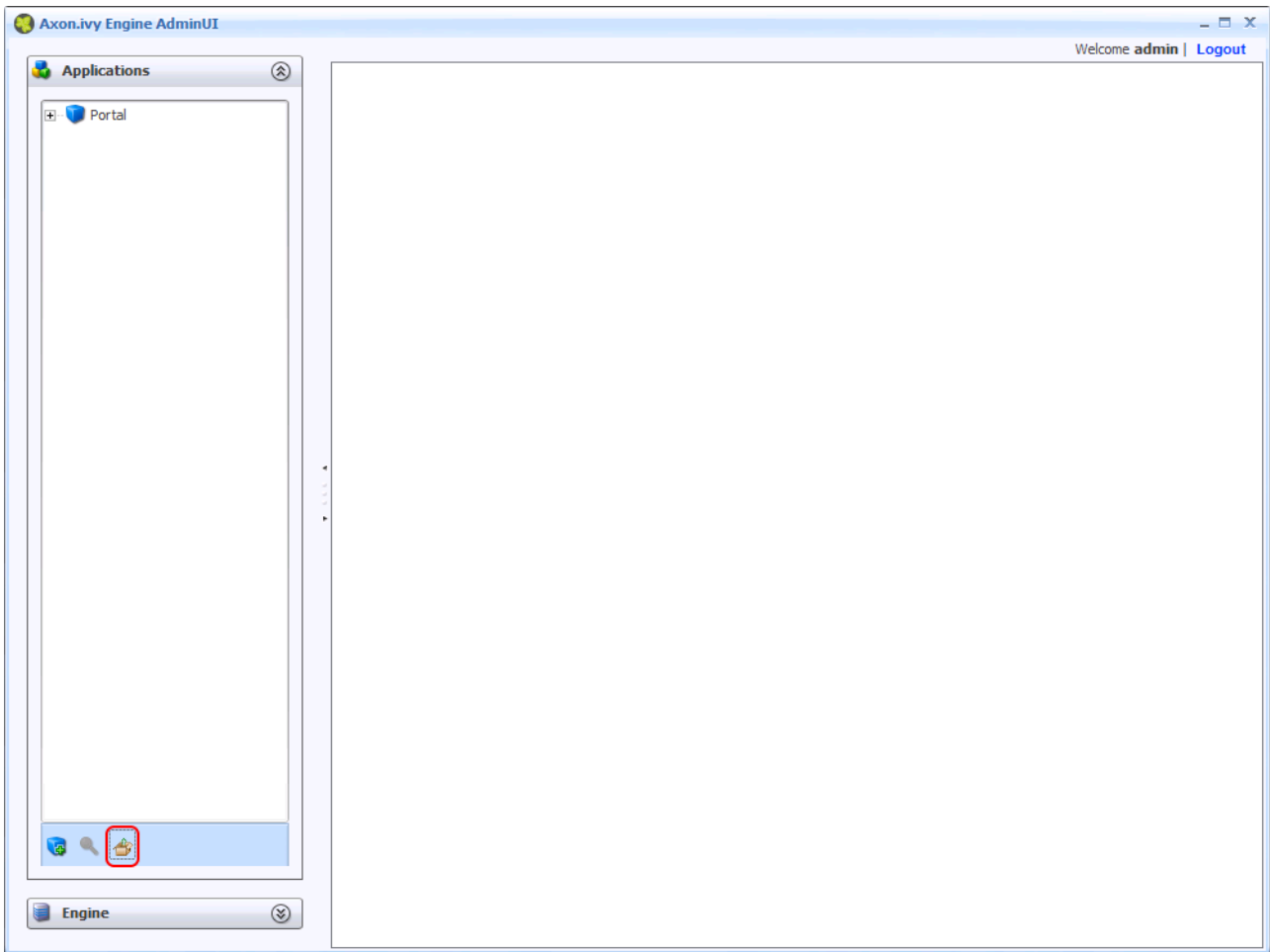
Let's deploy an Axon.ivy project to the Axon.ivy Engine. To do so install an Axon.ivy Designer first so that you have demo projects available that you can deploy. Simple download it from our download page and extract the AxonIvyDesigner*.zip file to the folder `c:\ivy\designer`.

Now start the **Admin UI** tool by clicking on the **Admin** menu on the Axon.ivy Engine main page. This will start Java Web Start. A dialog may appear asking if you want to run and trust the **Axon.ivy Rich Internet Application** . Press the **Run** button to proceed. After a few seconds, the Axon.ivy Engine Admin UI appears.

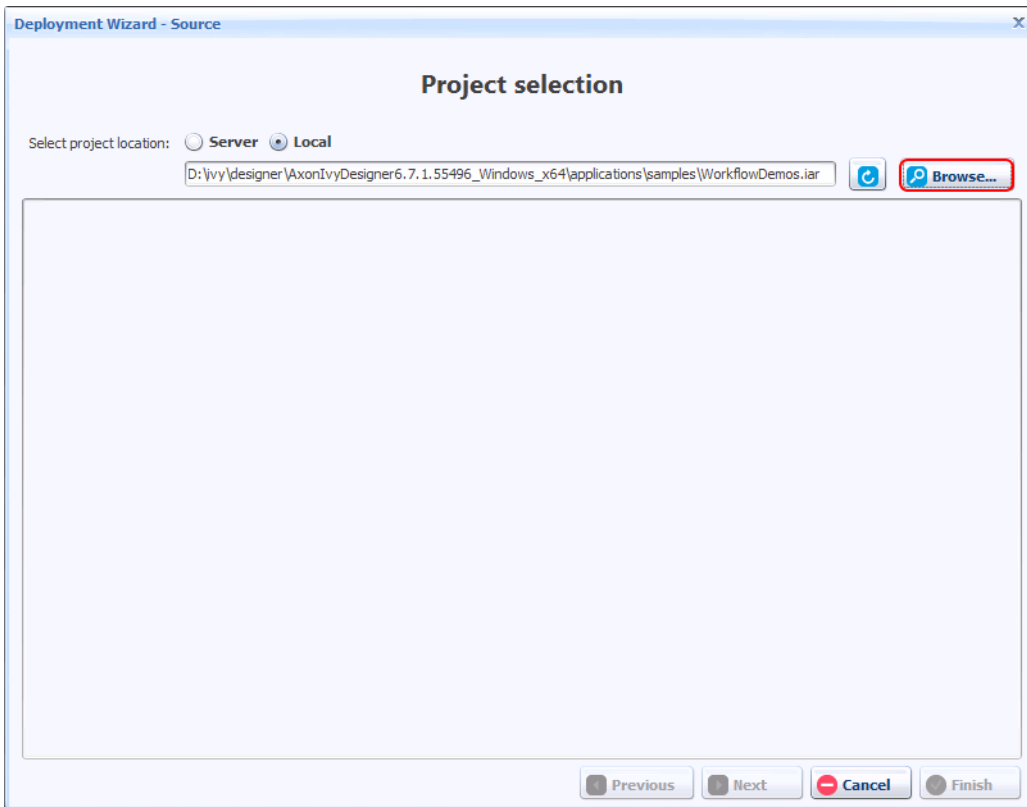


Use the credentials of the administrator that you have added in the configuration section to login. If you are still running in demo mode you can use **AxonIvy/AxonIvy** as credentials.

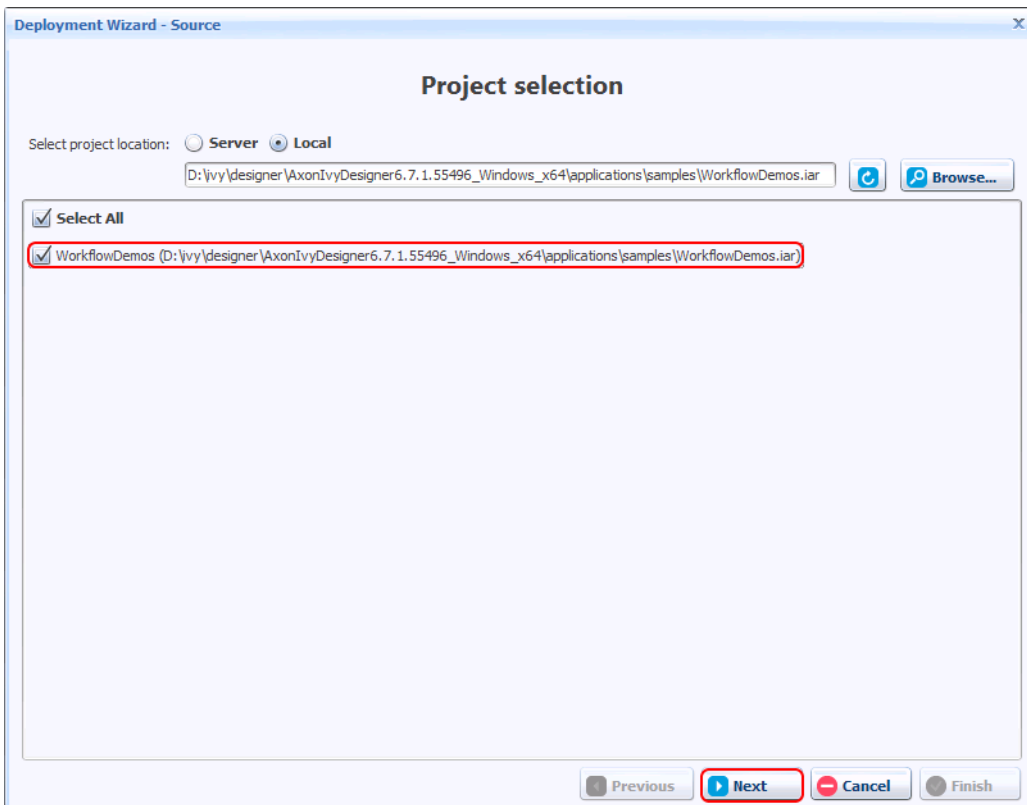
Then, press the **Deployment Wizard** button on the toolbar of the **Applications** section on the left side.



On the deployment wizard press the **Browse** button.



Choose the file *WorkflowDemos.iar* in the *applications/demos* folder of the Axon.ivy Designer installation folder and press the **Open** button.



Select the *WorkflowDemos* project and press the **Next** button.

An Axon.ivy Project is normally stored in a folder. However, the deployment wizard also supports to deploy *.iar files. An *.iar file is more or less a packed (zipped) Axon.ivy Project folder.

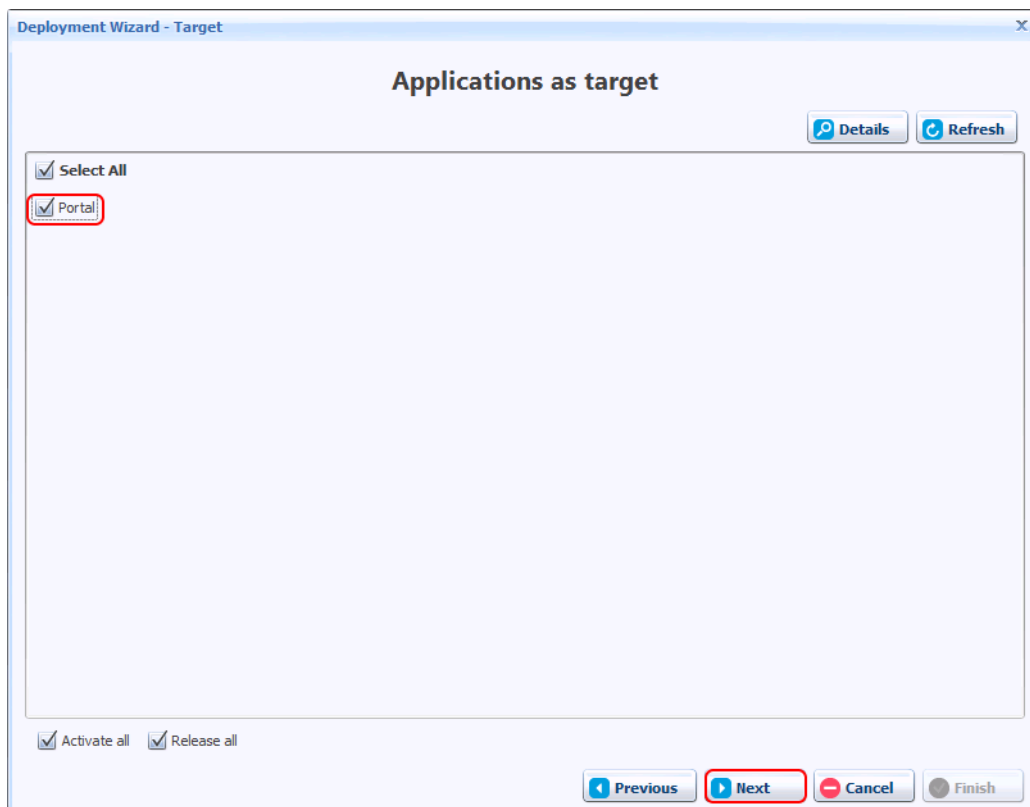
On the next page select the target application **Portal**.

An Axon.ivy Engine can manage multiple applications. Each application has its own independent user and task management. A user of one application can only work in that application and not in another application. A task of one application will never be visible in another application. Therefore, applications can be used either to build multi tenancy systems or run staging environments (DEV, Q&A, PROD) on the same Axon.ivy Engine.



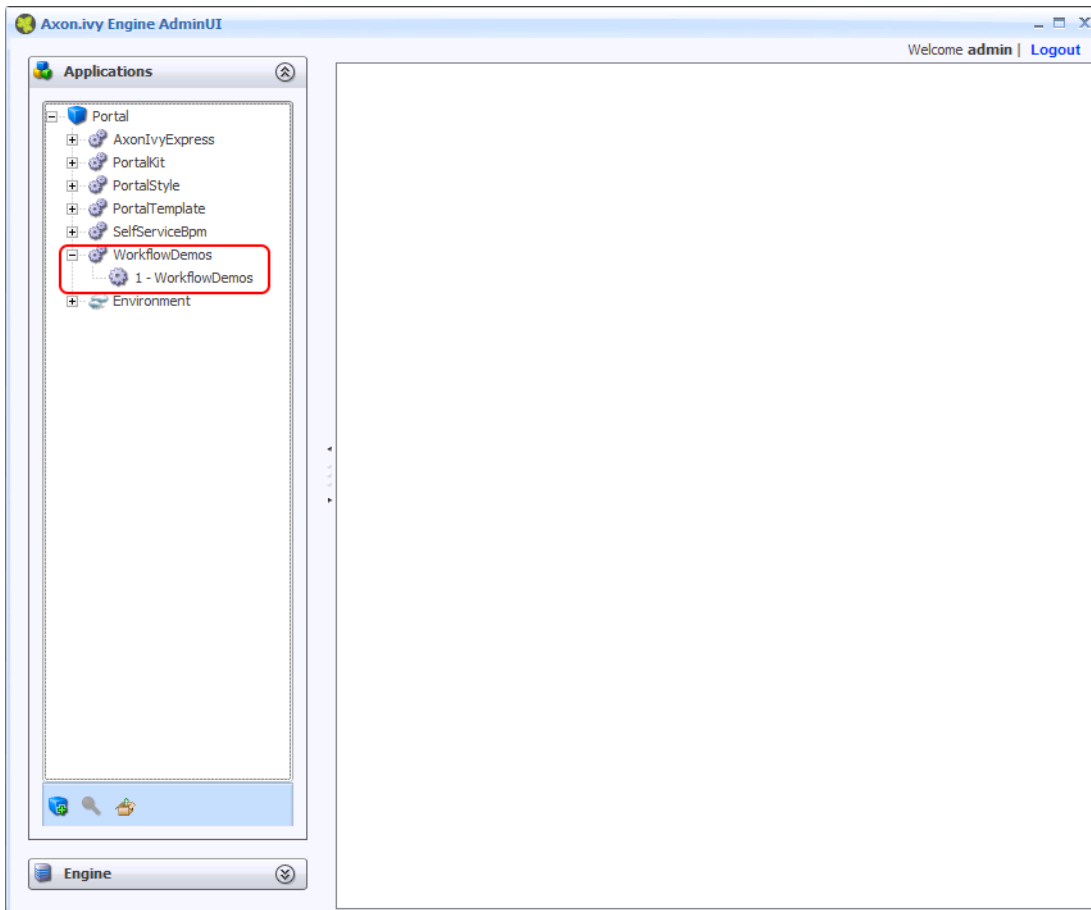
Warning

There is a default **Portal** application which should only be used for demo purpose. Create always a new application for the production environment, because the shipped **Portal** application has a special lifecycle and no real migration path.



Press the **Next** button.

On the next two pages press also the **Next** button and on the final page press the **Deploy** button. The project *WorkflowDemos* is now deployed to the application *Portal*. Press the button **Finish** to close the deployment wizard after the deployment has finished. The *WorkflowDemos* is now visible in the **Applications** tree on the left side of the **Admin UI**.



Close the **Admin UI** and go back to the web browser. Refresh the main page of the Axon.ivy Engine. There is now a new section called **WorkflowDemos** available with new links to start processes.

WorkflowDemos

Version:	WorkflowDemos\$1	
Process Engine:	Running	
Case Map Starts:	CaseMap/Lending.icm (Business Data)	
Process Starts:	5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	5.2: Concurrent Modification (Business Data)	This use case handles the problematic of concurrent modification of the same business data values.
	5.3 Migrate BusinessData format	Starts the migration of the internal Business Data JSON format. This is an administrative tasks, that should not be made accessible to workflow users.

Congratulations you have installed and configured your first Axon.ivy Engine and deployed your first Axon.ivy project. If you are interested in how to do the same but only with a console then read the next section. In the next chapter, you will learn the main concept of the Axon.ivy Engine and the details of how to configure, administrate and monitor it.

Debian Linux

In this section, you will learn how to install and configure an Axon.ivy Engine on a Debian based linux machine.

Install the Engine

There is a convenient DEB package available to install the Axon.ivy Engine. You can download and install it with the following bash script:

```
cd /tmp
wget https://developer.axonivy.com/permalink/latest/axonivy-engine.deb -O axonivy-engine.d
sudo dpkg -i axonivy-engine.deb
rm -f /tmp/axonivy-engine.deb
```

The most important folders of the Axon.ivy Engine are:

- the `/usr/lib/axonivy-engine-7x/` is the root installation folder with symlinks to all locations which are relevant to the engine.
- the `/etc/axonivy-engine-7x/` folder which contains the configuration files.
- the `/var/lib/axonivy-engine-7x/deploy` folder which is used to deploy Axon.ivy projects.

Examine the Engine

After the installation the engine will automatically be started as systemd service. You can check the operative state with `systemctl status axonivy-engine-7x`.

The output of the service status will expose an URI where the Axon.ivy Engine page is accessible.

```
locadmin@zugts
File Edit View Search Terminal Help
● axonivy-engine-7x.service - Axon.ivy Engine
   Loaded: loaded (/etc/systemd/system/axonivy-engine-7x.service; e
   Active: active (running) since Wed 2018-10-24 10:00:34 CEST; 6h
   Main PID: 1302 (java)
   Tasks: 67 (limit: 2319)
   CGroup: /system.slice/axonivy-engine-7x.service
           └─1302 /usr/bin/java -server -cp /usr/lib/axonivy-engin
             1307 tail -f -n0 --pid=1302 /usr/lib/axonivy-engine-7x

Oct 24 10:00:31 zugtstivydeb AxonIvyEngine[1142]: [ 90%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: [ 92%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: [ 94%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: [ 96%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: [ 98%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: [100%] Service Pr
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: Go to http://zugt
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: Axon.ivy Engine 1
Oct 24 10:00:34 zugtstivydeb AxonIvyEngine[1142]: Type 'shutdown' a
Oct 24 10:00:34 zugtstivydeb systemd[1]: Started Axon.ivy Engine.
lines 1-19/19 (END)
```

Copy this URL. On your client machine open a web browser and navigate to that URL. This will display the Axon.ivy Engine main page.

Use the Engine

The main page of the Axon.ivy Engine looks like this:

Server Config Admin **Portal**

Demo Mode - No license installed. Please use the [Engine Configuration](#) to set up your Axon.ivy Engine.

Applications

Portal

AxonIvyExpress

Version: AxonIvyExpress\$1
 Process Engine: Running
 Process Starts: Axon.ivy Express

PortalKit

Version: PortalKit\$1
 Process Engine: Running
 Process Starts:

Portal

- AxonIvyExpress
- PortalKit
- PortalStyle
- PortalTemplate
- SelfServiceBpm**

[Back to top](#)

The Axon.ivy Engine is running in Demo Mode. This is because you did not install a valid license yet nor did you configure a system database. Note, that everything that you do with the Axon.ivy Engine running in Demo Mode is lost when you shut down the engine. However, you can use the engine also in Demo Mode and tryout the pre-installed Portal application by clicking on the **Portal Home** link. To login use one of the predefined demo users: **demo**, **guest** or **admin**. The passwords of the demo users are equal to the user names (E.g. **demo** for the **demo** user). Login as **demo** user and try to create a **TODO** task for the guest user using the **Axon.ivy Selfservice** process:

AXON iVY
digitalize your business

demo

Processes [Show all processes](#)

- Axon.ivy Express
- Axon.ivy Selfservice**

[Add new process](#) | [Delete processes](#)

Statistics [Show all charts](#)

No tasks to display statistics.

Tasks [Show full task list](#)

Name or ID

Sort by priority | Sort by expiry time

No tasks.

Configure the **TODO** task as follows:

AXON iVY
digitalize your business

demo

Self Service Workflow (#159)

Request Case Information

Self Service Ad-hoc Workflow

Define and launch a Workflow at your fingertip: Select a flow pattern, add some tasks and start the Workflow.

Case

Subject * **Setup Axon.ivy Engine**

Description **Please setup a new Axon.ivy Engine on our prod server!**

Pattern ToDo Approval Question-Answer Ad-hoc [Details](#)

Workflow

CREATOR demo 15.09.2017 11:51

TODO x

guest 18.09.2017 11:51

+ Task

Start Workflow

Start the workflow by pressing the **Start Workflow** button. After that, logout, then login again as **guest** user. On the task list, you now see the new **TODO** task that you created before. Try to work on the task by clicking on the arrow located on the right side of the task:

Go ahead and play around with the Selfservice process. Try out different types of tasks.

In Axon.ivy a task is a piece of work (a part of a process) that is assigned to a user or role. A user to whom a task is assigned or who has the role to which a task is assigned can work on the task. When a user works on a task the task disappears from the task list of other users who might also be able to work on the task. This means only one user can really work on a task at the same time. In a process, it is possible to define parallel tasks. Therefore, it is possible that two or more users work parallel on different tasks of the same process instance. In Axon.ivy a process instance is called a case.

Configure the Engine

Now, let's configure the Axon.ivy Engine with a license and system database.

To start with that you must first request a valid Axon.ivy Engine license. Either you get a license for your productive system through one of our sales personal or contact our support for time limited tryout licenses. If you do not have a license you can skip this section and continue with the next section.

Moreover, you need to have a supported database server up and running with a database user that has the rights to create new databases. The configuration and creation of the system database differs a little bit depending on the database system you use. We will use a PostgreSQL database server.

Shutdown the Axon.ivy Engine first by stopping its service:

```
systemctl stop axonivy-engine-7x.service
```

Let's install the license. You can do this by simple copy the license *.lic file into the *configuration* folder:

```
cp ~/license.lic /etc/axonivy-engine-7x
```

To configure the system database, use the **config-db** command of the **EngineConfigCli** tool. Replace **yourdatabasserver** with the name of the host where your PostgreSQL server is running. Replace **dbuser** and **password** with the credentials of a database user that has the rights to create a new database on the database server.

```
cd /usr/lib/axonivy-engine-7x/bin
./EngineConfigCli config-db org.postgresql.Driver \
jdbc:postgresql://yourdatabasserver:5432/AxonIvySystemDatabase \
dbuser password
```

Now, let's create the system database with the **create-db** command:

```
./EngineConfigCli create-db
```

The system database is used by Axon.ivy Engine to store configurations, users, roles, process instances, tasks and process data.

Next, define an administrator by modifying the “ivy.yaml” file of the `/etc/axonivy-engine-7x` directory:

```
# sample ivy.yaml which defines a single administrative account:
Administrators:
  admin:
    Password: 1234
    Email: sudo@acme.com
```

Administrators can administrate the Axon.ivy Engine. For example, they can add or remove users, assign user to roles, enable or disable applications, etc.

Therefore, you need at least one administrator so that you can later administrate the Axon.ivy Engine.

The Email address of administrators are used to send mail notifications if license problems occur.

Optionally, disable the AJP protocol connector of the Axon.ivy Engine internal web server in the “ivy.webserver.yaml” file of the `/etc/axonivy-engine-7x` directory:

```
# sample ivy.yaml which disables AJP, as no front-end webserver is in charge:
Connector:
  AJP:
    Enabled: false
```

The Axon.ivy Engine internal web server provides connectors for the following protocols:

HTTP Protocol used by web browser to communicate with a web server. This protocol is not secure since the communication is not encrypted. Axon.ivy Engine uses port 8080 by default.

HTTPS Like HTTP but secure. It uses TLS to encrypt the communication between the web browser and server. Axon.ivy Engine uses port 8443 by default.

AJP This protocol is used to communicate with an Microsoft IIS or Apache httpd frontend server. AJP=Apache Jakarta Protocol. Axon.ivy Engine uses port 8009 by default.

Have a look at the “ivy.webserver.yaml” to see what other parts of the webserver and its connectors can be adjusted for your needs.

Now, start the Axon.ivy Engine again:

```
systemctl start axonivy-engine-7x.service
```

Note, that the HTTP port of the Axon.ivy Engine may have changed. If you did change the http settings! So open again a web browser and navigate to `http://yourservername:yourportnumber/ivy`. Note, that the header with the demo mode message is gone. You now have a production ready Axon.ivy Engine.

Deploy an Axon.ivy project to the Engine

Let's deploy an Axon.ivy project to the Axon.ivy Engine. We use demos that are shipped with the Axon.ivy Designer.

```
cd /var/lib/axonivy-engine-7x/deploy
wget https://developer.axonivy.com/permalink/latest/axonivy-designer-linux.zip -O /tmp/designer.zip
sudo -u ivy unzip -d IvyDemoApp -j /tmp/designer.zip applications/samples/WorkflowDemos.iar
rm /tmp/designer.zip
```

Here we deploy a new application by dropping packed (IAR) projects into it. However it is also possible to deploy full applications as ZIP or unpacked projects.

You can monitor the deployment with:








```
tail -f IvyDemoApp/*.iar.deploymentLog
```

As soon as the deployment is finished the iar-files will be postfixed with `.deployed`. In case of an error the postfix is `.notDeployed`. E.g. `WorkflowDemos.iar.deployed`.

An Axon.ivy Engine can manage multiple applications. Each application has its own user and task management. A user of one application can only work in that application and not in another application. A task of one application will never be visible in another application. Therefore, applications can be used to build multi tenancy or stages (DEV, Q&A, PROD) without to install multiple Axon.ivy Engines.

Refresh the main page of the Axon.ivy Engine. There is now a new application called **IvyDemoApp** with a section called **WorkflowDemos**. Under it links are available to start processes.

WorkflowDemos

Version:	 WorkflowDemos\$1	
Process Engine:	 Running	
Case Map Starts:	 CaseMap/Lending.icm (Business Data)	
Process Starts:	 5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	 5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	 5.2: Concurrent Modification (Business Data)	This use case handles the problematic of concurrent modification of the same business data values.
	 5.3 Migrate BusinessData format	Starts the migration of the internal Business Data JSON format. This is an administrative tasks, that should not be made accessible to workflow users.

Congratulations you have installed and configured your first Axon.ivy Engine and also deployed your first Axon.ivy processes.

Linux (with console tools only)

In this section, you will learn how to install and configure an Axon.ivy Engine on a Linux server only using the console. On the Linux machine no windowing system has to be installed. However, to test your configuration a client machine with a web browser is needed.

Prepare your Machine

Before starting with the installation of Axon.ivy Engine prepare your Linux machine with the necessary tools and software needed for the installation (`wget`, `unzip`, Java 8 runtime). Most distribution may have pre-installed these tools but especially certain Docker images may have not. On Debian (e.g. Ubuntu, etc.) based system use:

```
apt update
apt install sudo wget unzip openjdk-8-jre-headless
```

Install the Engine

We suggest that you install the Axon.ivy Engine into a new folder called `/opt/ivy/engine`. Create the directory and change the owner to your current user:

```
cd /opt
sudo mkdir ivy
sudo chown myuser:myuser ivy
```

Replace **myuser** with the name of your current user.

Instead of using your current user we suggest that on a productive system you use a special user called **ivy**. First, create a new user and group called **ivy**. Then, change the owner of the folder `ivy` to the user **ivy**. After that, login as user `ivy` and work with the new user.

```
sudo mkdir ivy
adduser ivy
...
sudo chown ivy:ivy ivy
ls -al
...
drwxr-xr-x 3 ivy ivy      4096 Sep 15 11:26 ivy
...
login ivy
...
```

Download the latest engine:

```
cd ivy
mkdir engine
cd engine
wget https://developer.axonivy.com/permalink/latest/axonivy-engine.zip -O engine.zip
```

To install Axon.ivy Engine simply unzip the downloaded file `AxonIvyEngine*.zip` into a new `AxonIvyEngine*` folder:

```
unzip engine.zip -d latest
rm engine.zip
cd latest
```

The most important folders in the Axon.ivy Engine installation folder are:

- the `bin` folder which contains all the executables.
- the `configuration` folder which contains the configuration files.
- the `deploy` folder which is used to deploy Axon.ivy projects.

Start the Engine

Start the Axon.ivy Engine by navigation to the `bin` folder and executing the `AxonIvyEngine` binary:

```
cd bin
./AxonIvyEngine
```

This will start the Axon.ivy Engine as a user process. On the last lines of the output a URL is displayed:

```
[100%] Service ProcessModelVersion Portal/AxonIvyExpress$1 started [0ms]
Go to http://yourservername:8080/ivy to see the info page of Axon.ivy Engine.
Axon.ivy Engine is running and ready to serve. [9375ms]
```

Type 'shutdown' and confirm with ENTER to stop the running engine instance

Copy this URL. On your client machine open a web browser and navigate to that URL. This will display the Axon.ivy Engine main page.

Normally, you want to run Axon.ivy Engine as a daemon process and not as a user process. You can register and manage the Axon.ivy Engine daemon using `systemd`.

Use the Engine

The main page of the Axon.ivy Engine looks like this:

The Axon.ivy Engine is running in Demo Mode. This is because you did not install a valid license yet nor did you configure a system database. Note, that everything that you do with the Axon.ivy Engine running in Demo Mode is lost when you shut down the engine. However, you can use the engine also in Demo Mode and tryout the pre-installed Portal application by clicking on the **Portal Home** link. To login use one of the predefined demo users: **demo**, **guest** or **admin**. The passwords of the demo users are equal to the user names (E.g. **demo** for the **demo** user). Login as **demo** user and try to create a **TODO** task for the guest user using the **Axon.ivy Selfservice** process:

Configure the **TODO** task as follows:

The screenshot shows the AXON iVY interface for configuring a workflow. The header includes the AXON iVY logo and a search bar with 'demo' selected. The main content area is titled 'Self Service Workflow (#159)' and contains two sections: 'Case' and 'Workflow'.

Case Section:

- Subject:** Setup Axon.ivy Engine (highlighted with a red box)
- Description:** Please setup a new Axon.ivy Engine on our prod server! (highlighted with a red box)
- Pattern:** Radio buttons for 'ToDo' (selected), 'Approval', 'Question-Answer', and 'Ad-hoc'. A 'Details' link is also present.

Workflow Section:

- CREATOR:** demo, 15.09.2017 11:51
- Task:** guest (highlighted with a red box), 18.09.2017 11:51
- A green '+ Task' button is visible.

At the bottom of the workflow section, there is a green 'Start Workflow' button with a checkmark icon.

Start the workflow by pressing the **Start Workflow** button. After that, logout, then login again as **guest** user. On the task list, you now see the new **TODO** task that you created before. Try to work on the task by clicking on the arrow located on the right side of the task:

Go ahead and play around with the Selfservice process. Try out different types of tasks.

In Axon.ivy a task is a piece of work (a part of a process) that is assigned to a user or role. A user to whom a task is assigned or who has the role to which a task is assigned can work on the task. When a user works on a task the task disappears from the task list of other users who might also be able to work on the task. This means only one user can really work on a task at the same time. In a process, it is possible to define parallel tasks. Therefore, it is possible that two or more users work parallel on different tasks of the same process instance. In Axon.ivy a process instance is called a case.

Configure the Engine

Now, let's configure the Axon.ivy Engine with a license and system database.

To start with that you must first request a valid Axon.ivy Engine license. Either you get a license for your productive system through one of our sales personal or contact our support for time limited tryout licenses. If you do not have a license you can skip this section and continue with the next section.

Moreover, you need to have a supported database server up and running with a database user that has the rights to create new databases. The configuration and creation of the system database differs a little bit depending on the database system you use. We will use a PostgreSQL database server.

Shutdown the Axon.ivy Engine first by typing **shutdown** and **Y**:

```
...
Go to http://ivy1:8080/ivy to see the info page of Axon.ivy Engine.
Axon.ivy Engine is running and ready to serve. [11596ms]
Type 'shutdown' and confirm with ENTER to stop the running engine instance
shutdown
Should 'Axon.ivy Engine' be stopped? ([Y]es / [N]o): Y
```

```
Stopping Axon.ivy Engine ...
[ 0%] Stopping Server
...
```

Let's install the license. You can do this by simple copy the license **.lic* file into the *configuration* folder:

```
cp ~/license.lic /opt/ivy/engine/latest/configuration
```

To configure the system database, use the **config-db** command of the **EngineConfigCli** tool. Replace **yourdatabasserver** with the name of the host where your PostgreSQL server is running. Replace **dbuser** and **password** with the credentials of a database user that has the rights to create a new database on the database server.

```
./EngineConfigCli config-db org.postgresql.Driver \
jdbc:postgresql://yourdatabasserver:5432/AxonIvySystemDatabase \
dbuser password
```

Now, let's create the system database with the **create-db** command:

```
./EngineConfigCli create-db
```

The system database is used by Axon.ivy Engine to store configurations, users, roles, process instances, tasks and process data.

Next, define an administrator by modifying the “ivy.yaml” file of the *configuration* directory:

```
# sample ivy.yaml which defines a single administrative account:
Administrators:
  admin:
    Password: 1234
    Email: sudo@acme.com
```

Administrators can administrate the Axon.ivy Engine. For example, they can add or remove users, assign user to roles, enable or disable applications, etc.

Therefore, you need at least one administrator so that you can later administrate the Axon.ivy Engine.

The Email address of administrators are used to send mail notifications if license problems occur.

Lastly, disable the AJP protocol connector of the Axon.ivy Engine internal web server in the “ivy.webserver.yaml” file of the *configuration* directory:

```
# sample ivy.yaml which disables AJP, as no front-end webserver is in charge:
Connector:
  AJP:
    Enabled: false
```

The Axon.ivy Engine internal web server provides connectors for the following protocols:

- HTTP** Protocol used by web browser to communicate with a web server. This protocol is not secure since the communication is not encrypted. Axon.ivy Engine uses port 8080 by default.
- HTTPS** Like HTTP but secure. It uses TLS to encrypt the communication between the web browser and server. Axon.ivy Engine uses port 8443 by default.
- AJP** This protocol is used to communicate with an Microsoft IIS or Apache httpd frontend server. AJP=Apache Jakarta Protocol. Axon.ivy Engine uses port 8009 by default.

Have a look at the “ivy.webserver.yaml” to see what other parts of the webserver and its connectors can be adjusted for your needs.

Now, start the Axon.ivy Engine again as background process.

```
nohup ./AxonIvyEngine &
```

Note, that the HTTP port of the Axon.ivy Engine may have changed. If you did change the http settings! So open again a web browser and navigate to `http://yourservername:yourportnumber/ivy`. Note, that the header with the demo mode message is gone. You now have a production ready Axon.ivy Engine.

Deploy an Axon.ivy project to the Engine

Let's deploy an Axon.ivy project to the Axon.ivy Engine. We use demos that are shipped with the Axon.ivy Designer.

```
cd /opt/ivy/engine/deploy/Portal
wget https://developer.axonivy.com/permalink/latest/axonivy-designer-linux.zip -O designer.zip
unzip -j designer.zip applications/samples/WorkflowDemos.iar applications/samples/Connectivity.iar
rm designer.zip
```

An Axon.ivy Project is normally stored in a folder. However, you can also deploy *.iar files. An *.iar file is more or less a packed (zipped) Axon.ivy Project folder.

You can monitor the deployment with:

```
tail -f WorkflowDemos.iar.deploymentLog
```

As soon as the deployment is finished the iar-files will be postfixed with .deployed. In case of an error the postfix is .notDeployed. E.g. *WorkflowDemos.iar.deployed*.

An Axon.ivy Engine can manage multiple applications. Each application has its own user and task management. A user of one application can only work in that application and not in another application. A task of one application will never be visible in another application. Therefore, applications can be used to build multi tenancy or stages (DEV, Q&A, PROD) without to install multiple Axon.ivy Engines.



Warning

There is a default **Portal** application which should only be used for demo purpose. Create always a new application for the production environment, because the shipped **Portal** application has a special lifecycle and no real migration path.

Refresh the main page of the Axon.ivy Engine. There is now a new section called **WorkflowDemos** available with new links available to start processes.

WorkflowDemos

Version:	WorkflowDemos\$1	
Process Engine:	Running	
Case Map Starts:	CaseMap/Lending.icm	
Process Starts:	5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	5.1: Browse Dossiers (Business Data)	Business Data Store Demo. Manage dossier objects in the Business Data Repository
	5.2: Concurrent Modification (Business Data)	This use case handles the problematic of concurrent modification of the same business data values.
	5.3 Migrate BusinessData format	Starts the migration of the internal Business Data JSON format. This is an administrative tasks, that should not be made accessible to workflow users.

Congratulations you have installed and configured your first Axon.ivy Engine and also deployed your first Axon.ivy project. If you are interested in how to do the same but with UI tools read the previous section. In the next chapter, you will learn the main concept of the Axon.ivy Engine and the details of how to configure, administrate and monitor it.

Docker

We provide an image for demo and testing purpose on Docker Hub: <https://hub.docker.com/r/axonivy/axonivy-engine/>

This image can be started with the following command:

```
docker run -p 8080:8080 axonivy/axonivy-engine
```

After startup, the engine can be accessed under the following url: <http://localhost:8080/ivy>

On GitHub you can find some examples how to use this image: <https://github.com/ivy-samples/docker-samples>



Warning

We strongly recommend to build an own image for production systems. Because the container should be always up to date with the newest security patches of the underlying system.

Take a closer look at the following Dockerfile if you need to build an own image:

```
FROM openjdk:8-jre-slim-stretch
LABEL maintainer="Reto Weiss <reto.weiss@axonivy.com>"

ARG IVY_ENGINE_DOWNLOAD_URL
ARG IVY_PACKAGE_NAME=axonivy-engine-7x
ARG IVY_HOME=/usr/lib/axonivy-engine

RUN apt-get update && \
    apt-get install -y wget && \
    \
    useradd --uid 1000 --user-group --no-create-home ivy && \
    \
    wget ${IVY_ENGINE_DOWNLOAD_URL} -O /tmp/${IVY_PACKAGE_NAME}.deb && \
    dpkg -i /tmp/${IVY_PACKAGE_NAME}.deb && \
    ln -s /usr/lib/${IVY_PACKAGE_NAME} ${IVY_HOME} && \
    ln -s /var/lib/${IVY_PACKAGE_NAME} /var/lib/axonivy-engine && \
    ln -s /etc/${IVY_PACKAGE_NAME} /etc/axonivy-engine && \
    rm -f /tmp/${IVY_PACKAGE_NAME}.deb && \
    \
    rm -rf ${IVY_HOME}/system/applications/System/EngineConfigUi && \
    rm -rf /usr/share/doc/${IVY_PACKAGE_NAME} && \
    rm -rf /var/lib/apt/lists/*

ADD --chown=ivy:ivy ./docker-entrypoint.sh ${IVY_HOME}/bin/docker-entrypoint.sh
RUN chmod u+x ${IVY_HOME}/bin/docker-entrypoint.sh

WORKDIR ${IVY_HOME}
USER 1000
EXPOSE 8080
ENTRYPOINT ["/usr/lib/axonivy-engine/bin/docker-entrypoint.sh"]
```

The following snippet shows the `docker-entrypoint.sh`:

```
#!/bin/bash
set -e

amountOfLicenceFiles=$(find /etc/axonivy-engine/*.lic -maxdepth 1 -type f|wc -l)
if [ $amountOfLicenceFiles -gt 1 ]; then
    bin/EngineConfigCli wait-for-db-server
    bin/EngineConfigCli create-db
else
    # only demo licence available
    echo "if you don't want to run in demo mode, install a licence file under /etc/axonivy-engine/*.lic"
```

```
fi  
exec bin/AxonIvyEngine
```

Chapter 3. Installation

Upgrading from an older version



Warning

Upgrading from Xpert.ivy Server 3.x to Axon.ivy Engine 5.x and later is not supported.

If you upgrade to a new update release within the same Long Term Support (LTS) or Leading Edge (LE) version (e.g. 7.0.3 to 7.0.4) then follow the steps in the section Preparation and Upgrade. A project migration and system database conversion are normally not needed if not explicitly communicated otherwise in the *Migration Notes*. Therefore, you can skip section Project Migration and steps 2 and 4 in the section Upgrade.

Preparation



Warning

Before upgrading of an Axon.ivy Engine read the *Migration Notes* document of the new version. This document will tell you exactly what has changed since the last version and will list any additional steps to be undertaken, which might not be described here.

1. Install the new Axon.ivy Engine version to a new installation directory (See section Install Axon.ivy Engine).
2. Read the *Migration Notes* document of the new version.
3. If necessary (according to the *Migration Notes*), request a new licence (see section Installing a Licence).
4. Back up the system database and the application file directories of the old installation.
5. Copy the file *serverconfig.xml* plus all YAML and properties files from the *configuration* directory of the old installation directory to the *configuration* directory of the new installation.
6. Unless a new licence is required (see 3.) you should also copy the old licence file to the new installation.
7. Modifications on any other configuration files located in the *configuration*, *elasticsearch/config*, *webapps/ivy/WEB-INF* or *webapps/ivy/META-INF* directories of the old installation may be ported over to the corresponding files in the new installation, if required. To see what has been changed, we recommend the usage of some diff tool to compare the individual config files of old and new installation.
8. If you have installed additional extension plugins into the *dropins* directory then copy them to the *dropins* directory of the new installation directory if they are compatible with the new Engine version. For those which are not compatible try to get new compatible versions and install them.

Project Migration

Project migration is only necessary if mentioned in the *Migration Notes*. If migration is required, all projects deployed to process model versions in state PREPARED, RELEASED, DEPRECATED and ARCHIVED must be converted. The following steps are necessary for every process model version:

1. Copy the project from the process model version file directory on the engine to a local directory on your developer machine.
2. Import the project into your Designer workspace.
3. Migrate the project according to the *Migration Notes* of the Designer. Usually this is achieved by invoking the *project conversion* action on each project (see Designer Guide for more information). Some manual adaptations may be necessary.

4. Test the migrated project in the Designer.

All migrated projects must be redeployed to the new, upgraded engine version (see next section).

Upgrade

1. Stop the engine of the old version (See section Start / Stop Engine).
2. Either convert the system database with the “Engine Configuration UI” (See “System Database”). Or set *autoConvert* property to *true* in the “ivy.yaml” of the *configuration* directory.
3. Start the engine of the new version (see section Start / Stop Engine).
4. Redeploy all converted/migrated Axon.ivy projects using the Axon.ivy Engine Administration (see section Deploying a Project).
5. You may now delete the old engine installation directory, **unless** the following warning applies to your installation:



Warning

Please note that the new, upgraded engine installation will still refer to the application file directories that were used by the old installation. As a consequence, you must never delete the directory of an old installation, if it contains application file directories (you can check the file directory by displaying the application information inside the Axon.ivy Engine Administration). If the application file directories of your installation are stored elsewhere, then the deletion of the old engine installation will not cause any problems.

Standard Edition Installation

It is recommended to read the Introduction chapter before installing an Axon.ivy Engine. The following list shows the necessary steps that are required to install and run an Axon.ivy Engine:

1. Gather all the information you need:
 - The server platform the engine will be installed on.
 - The database system used to host the system database.
 - Order a licence file for your installation. You need to know the host name of the machine you want to install the Axon.ivy Engine on. More information about the licence can be found in the section Install a Licence of the Installation chapter.
 - If an integration with a web server is planned, then get all the necessary configuration information of the web server.
 - If an integration with an external security system is planned, then get all the necessary configuration information of the external security system (e.g. Active Directory or Novell eDirectory).
 - Ivy uses a bundled Elasticsearch server to search through Business Data. If the use of an external Elasticsearch server is planned, then get the necessary configuration information for it. When running an Axon.ivy Engine Enterprise Edition the use of an external Elasticsearch server is mandatory. See the Elasticsearch section in the “ivy.yaml” of the *configuration* directory.
2. Install all required operating systems, web servers and database systems.
3. Install the Axon.ivy Engine
4. Install your licence file
5. Configure the Axon.ivy Engine
6. Start the Axon.ivy Engine and test if it is running.

If everything is fine so far, you can perform the following optional configuration steps:

1. Run the Axon.ivy Engine as Service which runs automatically after a reboot.
2. Integrate Axon.ivy Engine into web servers if necessary.
3. Deploy workflow applications.

Demo Mode

Axon.ivy Engine offers a demo mode for demonstration purposes. The demo mode allows you to install and start the Axon.ivy Engine without configuration and without a productive licence. To install and start an Axon.ivy Engine in demo mode simply execute the steps 3 and 6 from the list above.



Warning

The Axon.ivy Engine uses a memory database as system database in demo mode. This means that everything you configure and all cases that are created by any sessions in demo mode are lost when you shut down the engine.



Tip

In demo mode you can login to the Engine Administration using with the predefined user **admin** and password **admin**.

Enterprise Edition Installation

The installation process of an Axon.ivy Engine Enterprise Edition node is very similar to the standard installation process. To save time you can copy the configuration from the first node you have installed to other nodes. See the next chapters to learn how to install the first node, and how to proceed to install further nodes either on different machines or on the same machine.

Once you've installed all Axon.ivy Engine Enterprise Edition nodes you may want to integrate them into a web server that will act as single frontend. The web server can be configured to work as a load balancer that distributes the incoming requests evenly to the Axon.ivy Engine Enterprise Edition nodes. Consult the chapter Web Server Integration for more information.

Installation of the first engine node

Follow the standard installation process to install the first Axon.ivy Engine Enterprise Edition node.

For an Axon.ivy Engine Enterprise Edition installation an external Elasticsearch server installation is mandatory. See Elasticsearch installation for more information.

At point 4 you must make sure that you install an Axon.ivy Enterprise Edition licence.

At point 5 an additional Cluster configuration tab will be displayed in the “Cluster”. Inside this tab use the *Add local node* button to add the new node to the list of nodes of the Axon.ivy Engine Enterprise Edition.

Installation of another engine node on a different machine

To install further Axon.ivy Engine Enterprise Edition nodes on other machines proceed as follows:

1. Install the Axon.ivy Engine
2. Copy the *configuration* directory inside the installation directory of the first Axon.ivy Engine Enterprise Edition node to the installation directory of the currently installing node. Overwriting all existing files.
3. Replace the licence file from the first Axon.ivy Engine Enterprise Edition node with the Axon.ivy Enterprise Edition licence for this node in the *configuration* directory.

4. Start the “Engine Configuration UI” program. The system database and administrators and web server tab should display the values you have configured on the first node. Change to the Cluster tab and use the *Add local node* button to add the node to the list of nodes of the Axon.ivy Engine Enterprise Edition. Save your changes.
5. Start the Axon.ivy Engine Enterprise Edition node and test if it is running.

Installation of another engine node on the same machine

To install further engine nodes on the same machine where a node is already installed proceed as follows:

1. Install the Axon.ivy Engine
2. Copy the *configuration* directory inside the installation directory of the first engine node to the installation directory of the currently installing node. Overwrite all existing files.
3. Replace the licence file from the first engine node with the Axon.ivy Enterprise Edition licence for this node in the *configuration* directory.



Note

Every cluster node needs its own licence file even if nodes run on the same machine.

4. Start the “Engine Configuration UI” program. The system database and administrators tab should display the values you have configured for the first node.

Change to the WebServer tab and specify different port numbers than those you have specified for the other nodes on this machine.

Change to the Cluster tab and use the *Add local node* button to add the node to the list of nodes of the Axon.ivy Engine Enterprise Edition. Save your changes.

5. Start the Axon.ivy Engine Enterprise Edition node and test if it is running.

Install Axon.ivy Engine

To install the Axon.ivy Engine extract the correct zip file for your platform to the directory where you want to install the Axon.ivy Engine.

For Debian based operating systems there is a convenient installer package available.

The following platforms are supported:

CPU	Architecture	Operation System	Installation File
Intel	x64	Debian (Ubuntu/ Mint)	axonivy-engine-V_X.Y.Z.deb
Intel	x64	Windows	AxonIvyEngineX.Y.Z_Windows_x64.zip
Intel	x64	Linux and Windows*	AxonIvyEngineX.Y.Z_All_x64.zip
Intel	x64	Linux and Windows*	AxonIvyEngineX.Y.Z_Slim_All_x64.zip

Table 3.1. Supported Axon.ivy Engine Platforms

* The 'All' and 'Slim_All' engines are delivered with launchers for Linux and Windows, but without a JRE. To use the slim engine set up the *IVY_JAVA_HOME* environment variable pointing to a supported x64 Oracle JRE, or the *JAVA_HOME* environment variable pointing to a supported x64 Oracle JDK. The Slim engine comes without projects like the Portal.



Note

Note, that the installation procedure implies sufficient administration and access rights on the system. For example the access to drive C: on a Windows Server 2008 system is very restrictive that you might install the programs on drive D: instead.

Installed Files and Directories

After the installation the following files and folders are located in the Axon.ivy Engine installation folder:

Folder or File Name	Description
<i>bin/</i>	Contains programs to start and configure the Axon.ivy Engine
<i>clientlib/</i>	Contains libraries that are deployed to the client machines
<i>signed/linux/</i>	Linux specific libraries
<i>signed/linux_native/</i>	Native Linux libraries
<i>signed/windows/</i>	Windows specific libraries
<i>signed/windows_native/</i>	Native Windows libraries
<i>configuration/</i>	Contains the Axon.ivy Engine configuration data
<i>defaults/</i>	Documentation and templates for ivy.yaml files
<i>demo.lic</i>	Demo licence file
“ivy.yaml”	Main configuration file of the Axon.ivy Engine. Configures environments such as the system database, e-mail servers, administrators and more.
“ivy.cache.properties”	System database cache configurations
<i>keystore.jks</i>	Keystore with the default signature of the Axon.ivy Engine (for https/ssl)
<i>truststore.jks</i>	Empty truststore can be used to add trusted server certificate for SSL connection clients
“log4jconfig.xml”	Logging configuration
<i>servercontrolcenter.configuration</i>	Control Center configuration
<i>doc/</i>	
<i>html/</i>	Axon.ivy Engine Guide as HTML documentations
<i>pdf/</i>	Axon.ivy Engine Guide as PDF document
<i>newAndNoteworthy/</i>	Contains new and noteworthy features of the latest Axon.ivy Engine and Designer releases
<i>migrationNotes/</i>	Contains migration notes of the latest Axon.ivy Engine and Designer releases
<i>dropins/</i>	Third party extension libraries that contribute to the Axon.ivy runtime
<i>elasticsearch/</i>	Bundled Elasticsearch server
<i>jre/</i>	Java Runtime Environment for Axon.ivy Engine
<i>logs/</i>	Contains the log files
<i>misc/</i>	
<i>apache</i>	Files to integrate Axon.ivy into an Apache web server
<i>iis</i>	Files to integrate Axon.ivy into a Microsoft Internet Information Server (IIS)
<i>visualvm</i>	The Axon.ivy VisualVM plugin file
<i>system/</i>	The OSGi system
<i>applications/</i>	The system provided applications
<i>configuration/</i>	The OSGi configuration
<i>features/</i>	Installed OSGi features

Folder or File Name	Description
<i>lib/boot/</i>	OSGi boot classpath libraries
<i>plugins/</i>	Installed OSGi plugins. Basically all default or automatically installed java libraries of the Axon.ivy Engine
<i>projects/</i>	Contains deployable Axon.ivy projects
<i>webapps/</i>	
<i>ivy/</i>	Contains the Axon.ivy Engine web interface
<i>ivy/info/</i>	Contains the info web pages
<i>ivy/WEB-INF/</i>	Contains the “web.xml” file
<i>ivy/META-INF/</i>	Contains the “context.xml” file
<i>ivy/wf/</i>	Contains the workflow web interface
<i>work/</i>	Contains temporary files that are created and used by the Axon.ivy Engine
<i>NewAndNoteworthy.html</i>	Overview / entry point for list of new and noteworthy features in this release
<i>MigrationNotes.html</i>	Overview / entry point for migration of last to current release
<i>Readme.html</i>	Important information about this engine release
<i>ReleaseNotes.txt</i>	Release notes with bug fixes and new features

Table 3.2. Installed Files and Directories

Windows Programs

The *bin* folder of a windows installation contains the following native dynamic link libraries and executable files:

File	Description
<i>Example.ilc</i>	Example of an ivy launch control file. For more information see section “Windows Program Launcher Configuration” .
<i>JavaWindowsServiceHandler.dll</i>	Library that contains native methods to register, unregister, configure, start and stop windows services
<i>JVMLauncher.dll</i>	Library containing code to launch the Java virtual machine.
<i>NTEventLogAppender.dll</i>	Library that implements native methods to log into the windows event log (32 Bit only).
<i>ControlCenter.exe</i>	Program that allows to configure, start and stop the Axon.ivy Engine. It also permits to configure the Windows services. For more information see section “ Control Center”.
<i>ControlCenterC.exe</i>	Same as <i>ControlCenter.exe</i> but additionally logs any output to a console window.
<i>AxonIvyEngine.exe</i>	Starts the Axon.ivy Engine. For more information see section “AxonIvyEngine”.
<i>AxonIvyEngineC.exe</i>	Same as <i>AxonIvyEngine.exe</i> but additionally logs any output to a console window.
<i>AxonIvyEngineConfig.exe</i>	Program to configure the Axon.ivy Engine. For more information see section “Engine Configuration UI”.
<i>AxonIvyEngineConfigC.exe</i>	Same as <i>AxonIvyEngineConfig.exe</i> but additionally logs any output to a console window.
<i>AxonIvyEngineService.exe</i>	Executable of the Windows service. For more information see section “Windows Service”.

Table 3.3. Windows Programs

Linux Programs

The *bin* folder of a Linux installation contains the following script files:

File	Description
<i>AxonIvyEngine</i>	Starts the Axon.ivy Engine. For more information see section “AxonIvyEngine”.
“AxonIvyEngine.conf”	Java virtual machine configuration (Xms, Xmx, JMX, ...) for the Engine.
<i>AxonIvyEngineConfig</i>	Program to configure the Axon.ivy Engine. For more information see section “Engine Configuration UI”.
<i>AxonIvyEngine.service</i>	Template systemd script of the Linux service. It will be copied to <i>/etc/systemd/system/</i> by running <i>InstallService.sh</i> .
<i>control.conf</i>	Java virtual machine configuration (Xms, Xmx, JMX, ...) for the control tools (ControlCenter & AxonIvyEngineConfig)
<i>ControlCenter</i>	Program that allows to configure, start and stop the Axon.ivy Engine. For more information see section “Control Center”.
<i>InstallService.sh</i>	Script to install the Axon.ivy Engine as a daemon. For more information see section “Linux Service (systemd)”.
<i>launcher.sh</i>	Helper script to launch a Java program.

Table 3.4. Linux Programs

Installing a Licence

By default a demo licence is installed that allows you to run the Axon.ivy Engine in demo mode. You have to install a licence in order to run Axon.ivy Engine in a production environment.



Note

The licence file contains the name of the host where the engine is installed on. The licence will only work if the name of the machine exactly matches the name stored in the licence file.

To install a licence file follow the steps below:

1. Copy the licence file **.lic* to the directory *configuration/*.
2. Change the extension of your old licence files to anything, but **.lic* (e.g. from *foo_bar_another_licence.lic* to *foo_bar_another_licence.lic.old*).



Tip

You may leave *demo.lic* in the *configuration* folder, because this licence is taken only if no other licence files are found.

System Database

The Axon.ivy Engine system database is used by the server to store configuration, security, content and workflow information. See chapter Configuration to find out how you can create and configure Axon.ivy Engine system databases. Axon.ivy Engine supports the following database systems to host the system database:

- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- Postgre SQL

Password Encryption

Passwords are stored encrypted in the system database using state of the art encryption algorithms. More information can be found in the chapter System Database Encryption.

Character set and collation

All characters in databases are encoded with a specific charset (e.g. utf8, latin1, cp1257). Lastly it defines which kind of characters can be stored at all.

The collation is a set of rules that defines how the database management system compares and orders the data (e.g. utf8_unicode_ci, latin2_general_ci). Common abbreviations in the name of the collations are the following:

- ci = case insensitive
- cs = case sensitive
- ai = accent insensitive
- as = accent sensitive

As well as the character set the collation can be defined mostly on several levels: server, database, table or column. Everything about this subject is very dependent on the actual database management system.

Support case insensitive searches

If a case insensitive search is required, it must be guaranteed that the affected column has a case insensitive collation.

- 1. Check character set & collation of the column
- 2. Change character set & collation if necessary

Look at the specific chapters for your database below.

MySQL

Information

MySQL is an Open Source database. For more information go to www.mysql.com. You can download the latest mysql JDBC driver (Connector/J) from www.mysql.com.

Configuration

The following table explains the fields you have to configure on *System Database* tab in the Axon.ivy Engine Configuration program:

Field Name	Value(s)	Description
Database	MySQL	The database system to use.
Driver	MySQL	The database JDBC driver to use.
Host	localhost, testdbserv, ...	The name of the host where the database system is running.
Port	3306, 3307, 3308, ...	The IP port where the database system is listening for requests.
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the Axon.ivy Engine system database.

Field Name	Value(s)	Description
User Name	root, admin, AxonIvy, ...	The name of the database user who is used to connect to the database system.
Password	****	The password of the database user who is used to connect to the database system.

Table 3.5. MySQL Configuration

Creation

The following table explains the additional creation parameters you have to configure on *System Database* tab in the Axon.ivy Engine Configuration program, if you want to create a new system database on a MySQL database system:

Field Name	Value(s)	Description
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the database to create
Engine Type	InnoDB	The type of the MySQL database engine to use. At the moment only InnoDB is possible.

Table 3.6. MySQL Creation Parameter

Driver

The following table shows information about the JDBC driver that Axon.ivy Engine uses to connect to MySQL database systems:

JDBC Driver Name	com.mysql.jdbc.Driver
JDBC Connection URL Format	jdbc:mysql://<host>[:<port>]/<database name>

Table 3.7. MySQL Driver

Character set & collation

If you want to check the collation of a specified column you can use the following query:

```
SELECT character_set_name, collation_name FROM information_schema.columns WHERE
table_schema = "AXON_IVY_SYSTEM_DATABASE" AND table_name = "iwa_case" AND column_name
= "name"
```

If you need to change the collation look at the official MySQL reference for all supported character sets and collations. The simplest way is to use a case insensitive collation of the current character set. The following code will apply a new collation.

```
ALTER TABLE iwa_case MODIFY name VARCHAR(200) CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

You can find more information about modifying the column in the MySQL reference.

If the Axon.ivy Engine Configuration creates a new database, the following parameters will be automatically applied. If you want to use different charset or collation create an empty database manually with your configuration and then use the Axon.ivy Engine Configuration to create the tables, views and indexes in that database.

Parameter	Value
Charset	utf8

Parameter	Value
Collation	utf8_unicode_ci

Table 3.8. MySQL Default Database Configuration

MariaDB

Information

MariaDB is an Open Source database. For more information go to mariadb.org. MariaDB is a drop-in replacement for MySQL. The documentation for MySQL often applies to MariaDB as well.

Configuration

The following table explains the fields you have to configure on *System Database* tab in the Axon.ivy Engine Configuration program:

Field Name	Value(s)	Description
Database	MariaDB	The database system to use.
Driver	MariaDB	The database JDBC driver to use.
Host	localhost, testdbserv, ...	The name of the host where the database system is running.
Port	3306 , 3307, 3308, ...	The IP port where the database system is listening for requests.
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the Axon.ivy Engine system database.
User Name	root, admin, AxonIvy, ...	The name of the database user who is used to connect to the database system.
Password	****	The password of the database user who is used to connect to the database system.

Table 3.9. MariaDB Configuration

Creation

The following table explains the additional creation parameters you have to configure on *System Database* tab in the Axon.ivy Engine Configuration program, if you want to create a new system database on a MariaDB database system:

Field Name	Value(s)	Description
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the database to create
Engine Type	InnoDB	The type of the MariaDB database engine to use. At the moment only InnoDB is possible.

Table 3.10. MariaDB Creation Parameter

Driver

The following table shows information about the JDBC driver that Axon.ivy Engine uses to connect to MariaDB database systems:

JDBC Driver Name	org.mariadb.jdbc.Driver
JDBC Connection URL Format	jdbc:mariadb://<host>[:<port>]/<database name>

Table 3.11. MariaDB Driver

Character set & collation

If you want to check the collation of a specified column you can use the following query:

```
SELECT character_set_name, collation_name FROM information_schema.columns WHERE
table_schema = "AXONIVYSYSTEMDATABASE" AND table_name = "iwa_case" AND column_name
= "name"
```

If you need to change the collation look at the official MariaDB reference for all supported character sets and collations. The simplest way is to use a case insensitive collation of the current character set. The following code will apply a new collation.

```
ALTER TABLE iwa_case MODIFY name VARCHAR(200) CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

You can find more information about modifying the column in the MariaDB reference.

If the Axon.ivy Engine Configuration creates a new database, the following parameters will be automatically applied. If you want to use different charset or collation create an empty database manually with your configuration and then use the Axon.ivy Engine Configuration to create the tables, views and indexes in that database.

Parameter	Value
Charset	utf8
Collation	utf8_unicode_ci

Table 3.12. MariaDB Default Database Configuration

Oracle

Information

Oracle database is a database management system from the Oracle Corporation. For more information go to www.oracle.com

Configuration

The following table explains the fields you have to configure on the System Database tab in the Axon.ivy Engine Configuration program:

Field Name	Value(s)	Description
Database	Oracle	The database system to use.
Driver	Oracle Thin, Oracle Oci	The database JDBC driver to use. Either Thin or OCI driver.
Host	localhost, testdbserv, ...	The name of the host where the database system is running.
Port	1521 , 1522, 1523, ...	The IP port where the database system is listening for requests.
Oracle Service ID (SID)	oracle, db, ...	The identifier of the oracle service.
User Name	root, admin, AxonIvy, ...	The name of the database user who is used to connect to the database system.

Field Name	Value(s)	Description
Password	****	The password of the database user who is used to connect to the database system.

Table 3.13. Oracle Configuration**Tip**

On all (reused) oracle database connections the maximum number of open cursors is set to 1000, independently from the default setting that may be set on the database itself. Those cursors are needed to cache all prepared statements and also for PL/SQL blocks.

It may turn out that the number of open cursors is exceeded, which is indicated by an error message similar to the following:

```
ch.ivyteam.ivy.persistence.PersistenceException: java.sql.SQLException:
ORA-00604: error occurred at recursive SQL level 1
ORA-01000: maximum open cursors exceeded
```

If this should happen, then you may customize (and increase) the number of open cursors per connection with the Java system property `ch.ivyteam.ivy.persistence.db.oracle.MaxOpenCursors`. It can be set in the “Launch Configuration”.

Creation**Note**

With Oracle database the Axon.ivy Engine Configuration program will not create a new database instance for Axon.ivy Engine instead it will create only the user/schema and the tables into a given tablespace.

Before you can create the system database tables on a Oracle Database you have to do the following steps:

1. You may want to create a new Oracle database where the Axon.ivy Engine System Database is located. This is optional you can use an already existing Oracle database.
2. Create a new user (e.g. AxonIvy). Grant all necessary permissions to the user so that he can create and alter tables, indexes, sequences. Of course the user must be able to insert, update, delete and select data from the tables of the system database. This is optional you can use an already existing Oracle user or let the Axon.ivy Engine create one for you with the Axon.ivy Engine Configuration.
3. You may want to create a new tablespace (e.g. AxonIvy) where the Axon.ivy Engine System Database can store the table data. This is optional you can use an already existing tablespace.

**Warning**

Be sure that the configuration of the database connection uses the new database and the Oracle Service ID reflecting the database you want to create the system database tables in.

The following table explains the additional creation parameters you have to configure on the *System Database* tab in the Axon.ivy Engine Configuration program if you want to create a new system database on Oracle database system:

Field Name	Value(s)	Description
Tablespace	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the tablespace where the system database tables will store their data in.
User	AxonIvy, ...	The name of the user which will be created if she is not already existing.

Field Name	Value(s)	Description
		Tables, indexes and views are created in the schema of this user.
Password	***	The password for the given user.

Table 3.14. Oracle Creation Parameter

Driver

The following table shows information about the JDBC driver Axon.ivy Engine uses to connect to Oracle database systems:

JDBC Driver Name	oracle.jdbc.OracleDriver
JDBC Connection URL Format	jdbc:oracle:thin:@<host>:<port>:<service id>

Table 3.15. Oracle Driver

Character set & collation

The character set is defined with the `CREATE DATABASE` statement which is part of the customer configuration (look at creation). We recommend to use `AL32UTF8` (unicode) or at least `WE8ISO8859P1`, which supports Western European languages. By the use of `AL32UTF8` all text fields like `VARCHAR` or `CHAR` supports unicode characters.

Oracle databases works with session parameters for sorting and comparing data. The concept is called Linguistic Sorting and Matching. In the Oracle Database Manager you can configure the initialization parameters. For case insensitive sorting and comparing you can set `NLS_SORT` to `AMERICAN` (select an appropriate language) and `NLS_COMP` to `LINGUISTIC`. At the moment, you can't override the `NLS_LANGUAGE` and `NLS_TERRITORY` which is set to `AMERICAN`.

Microsoft SQL Server

Information

Microsoft SQL Server is a database system from Microsoft. For more information go to www.microsoft.com. We support two different JDBC Drivers for Microsoft SQL Server. First, the official JDBC Driver from Microsoft that provides support for the latest SQL Server features like high availability with multi subnet fail-over. Second, the matured and long-time used `jTDS` JDBC Driver.

Configuration

The following table explains the fields you have to configure on the *System Database* tab in the Axon.ivy Engine Configuration program:

Field Name	Value(s)	Description
Database	Microsoft SQL Server	The database system to use.
Driver	jTDS or Microsoft SQL Server	The database JDBC driver to use.
Host	localhost, testdbserv, ...	The name of the host where the database system is running.
Port	1433 , 1434, 1435, ...	The IP port where the database system is listening for requests.
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the Axon.ivy Engine system database .
User Name	sa, root, admin, AxonIvy, ...	The name of the database user who is used to connect to the database system.

Field Name	Value(s)	Description
Password	****	The password of the database user who is used to connect to the database system.

Table 3.16. Microsoft SQL Server Configuration



Important

If you want to connect to an existing instance of a MS SQL Server you have to configure an additional connection property that is called `instance/instanceName` containing the name of the corresponding database instance.

Creation

The following table explains the additional creation parameters you have to configure on the *System Database* tab in the Axon.ivy Engine Configuration program if you want to create a new system database on Microsoft SQL Server database system:

Field Name	Value(s)	Description
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the database to create

Table 3.17. Microsoft SQL Server Creation Parameter

Driver

The following table shows information about the JDBC drivers Axon.ivy Engine uses to connect to Microsoft SQL Server database systems:

Driver Name	Connection URL Format	Documentation
net.sourceforge.jtds.jdbc.Driver	<code>jdbc:jtds:sqlserver://<host>[:<port>]/<database name></code>	jTDS JDBC Driver Documentation
com.microsoft.sqlserver.jdbc.SQLServerDriver	<code>jdbc:sqlserver://<host>[:<port>];DatabaseName=<database name></code>	Microsoft SQL Server JDBC Driver Documentation

Table 3.18. Microsoft SQL Server Driver

Character set & collation

MSSQL Server supports different (one byte-) character sets with the COLLATE parameter. If you need a unicode character set you have to modify the text fields to type NVARCHAR, NCHAR or NTEXT. First create a new column of this type, copy data from the old column, drop the old column and rename the new column to the old column.

If you want to check the collation of a specific column you can use the following query:

```
SELECT columns.collation_name FROM information_schema.columns WHERE table_name = 'iwa_case' AND column_name = 'name';
```

Probably you need to change the collation. Look at the official MSSQL reference for all supported character sets and collations. The simplest way is to use a case insensitive collation of the current character set. The following code will apply a new collation.

```
ALTER TABLE iwa_case ALTER COLUMN name VARCHAR(200) COLLATE Latin1_General_CS_AI;
```

If the Axon.ivy Engine Configuration creates a new database, the following parameters will be automatically applied. If you want to use different collation create an empty database manually with your configuration and then use the Axon.ivy Engine Configuration to create the tables, views and indexes in that database.

Parameter	Value
COLLATE	Latin1_General_CI_AI

Table 3.19. Microsoft SQL Server Default Database Configuration

PostgreSQL

Information

PostgreSQL is an Open Source database. For more information go to www.postgresql.org. You can download the latest PostgreSQL JDBC driver from jdbc.postgresql.org

Configuration

The following table explains the fields you have to configure on the *System Database* tab in the Axon.ivy Engine Configuration program:

Field Name	Value(s)	Description
Database	PostgreSQL	The database system to use.
Driver	PostgreSQL	The database JDBC driver to use.
Host	localhost, testdbserv, ...	The name of the host where the database system is running.
Port	5432, 5433, 5434, ...	The IP port where the database system is listening for requests.
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the database to use.
User Name	sa, root, admin, AxonIvy, ...	The name of the database user who is used to connect to the database system.
Password	****	The password of the database user who is used to connect to the database system.

Table 3.20. PostgreSQL Configuration

Creation

The following table explains the additional creation parameters you have to configure on the *System Database* tab in the Axon.ivy Engine Configuration program if you want to create a new system database on PostgreSQL database system:

Field Name	Value(s)	Description
Database Name	AxonIvy, AxonIvyEngine, AxonIvySystemDatabase, ...	The name of the database to create

Table 3.21. PostgreSQL Creation Parameter

Driver

The following table shows information about the JDBC driver Axon.ivy Engine uses to connect to PostgreSQL database systems:

JDBC Driver Name	org.postgresql.Driver
JDBC Connection URL Format	jdbc:postgresql://<host>[:<port>]/<database system>

Table 3.22. PostgreSQL Driver

Character set & collation

The character set can only be defined by the `CREATE DATABASE` statement.

If you want to know the collation of a column you can use the following query:

```
SELECT character_set_name, collation_name FROM information_schema.columns WHERE
table_name = 'iwa_case' and column_name = 'name';
```

Maybe there is no collation defined, so PostgreSQL won't show you any value. Ask the DBMS for the default value:

```
SELECT datname, datcollate FROM pg_database;
```

Probably you need to change the collation. Look at the official PostgreSQL reference for all supported character sets and collations. Look also at [Collation Support](#) and `ALTER TABLE` for more information about collations in PostgreSQL.

```
ALTER TABLE iwa_case ALTER COLUMN name varchar(200) COLLATE "de_DE.utf8";
```

If the Axon.ivy Engine Configuration creates a new database, the following parameters will be automatically applied. If you want to use different charset or collation create an empty database manually with your configuration and then use the Axon.ivy Engine Configuration to create the tables, views and indexes in that database.

Parameter	Value
ENCODING	UTF8

Table 3.23. PostgreSQL Default Database Configuration

Chapter 4. Configuration

Engine Configuration

The Axon.ivy engine is configured by files. Most of them are located in the */configuration* directory of the engine.

Files

The “ivy.yaml” file contains the most important entries that define the environment and runtime behaviour of the Axon.ivy engine.

```
# sample ivy.yaml with some often used entries defined
SystemDb:
  Driver: org.mariadb.jdbc.Driver
  Url: jdbc:mariadb://myDbHost:3306/AxonIvySystemDatabase
  UserName: root
  Password: 1234
EMail:
  Server:
    Host: smtp.gmail.com
    Port: 25
Administrators:
  admin:
    Password: 1234
    Email: info@localhost
  devop:
    Password: "${encrypt:4321}"
    Email: dev@axonivy.com
Frontend:
  HostName: workflow.acme.com
  Port: 80
```

Template files

To craft your own configuration you would typically copy values from our template files, located under *[engineDir]/configuration/defaults* or the “Configuration File Reference” and adjust them according to your needs. The template files outline valid configuration attributes and document possible values. They are constantly improved by us, and are not designed to store your actual configuration.

System Database

An untouched Axon.ivy engine runs in Demo mode. In consequence workflow data is never stored, but kept in an memory database. To run a productive engine an external system database must be connected, where workflow data will be stored.

To define the database of the Axon.ivy engine, the `SystemDb` entries must be set.

```
# sample ivy.yaml that configures a MySQL database as data store
SystemDb:
  Driver: org.mysql.jdbc.Driver
  Url: jdbc:mysql://myOtherMysql:3306/AxonIvySystemDatabase
  UserName: theUser
  Password: myPassword
```

To run the Axon.ivy engine with a System Database a license is required. See “Installing a Licence”.

The schema of the Axon.ivy System Database must exist on the referenced database system. The Engine Config UI and “EngineConfigCli” simplify the creation of the `SystemDb` connection.

Users

Users are kept in a so-called security system which can be defined in “ivy.yaml”. Each application defines in “app.yaml” which security system is used. There are two types of security systems:

- **Internal Security System:** Used to manage the users directly on the Axon.ivy engine. There is only one Internal Security System, which is called Ivy Security System. No further settings are available for this Security System. This is also the default Security System for application which has no security system defined.
- **External Security System:** Used to synchronize users from a name and directory service such as Active Directory. The example below shows a simple connection to an Active Directory. Have a look at the configuration file reference for all supported name and directory services and further settings.

```
# sample ivy.yaml that define an Active Directory as security system
SecuritySystems:
  # Custom defined name of your security system
  ActiveDirectoryOfMyCompany:
    Provider: "Microsoft Active Directory"
    Connection:
      Url: "ldap://activedirectory.axonivy.com:389"
      UserName: "activedirectory_user@axonivy.com"
      Password: "${encrypt:1234}"
    Binding:
      DefaultContext: "DC=axonivy,DC=com"
      ImportUsersOfGroup: "CN=AXON Ivy IT,DC=axonivy,DC=com"
```

```
# app.yaml located in <application-directory>/app.yaml
SecuritySystem: ActiveDirectoryOfMyCompany
```

Email

The Axon.ivy engine sends emails for different purposes:

- Emails that are sent within a process via the mail step.
- New task assignment and daily task summaries to users.
- License expiration reminders to the administrators.

For this you have to configure an email server in “ivy.yaml”:

```
# sample ivy.yaml that configures an email server:
EMail:
  Server:
    Host: mail.axonivy.com
    Port: 25
    MailAddress: noreply@axonivy.com
    User: someuser
    Password: somepassword

  DailyTaskSummary:
    # Time of day when the task summary mails will be sent.
    TriggerTime: "02:00"
```

You can configure task email notification settings for new task assignments and daily task summaries at application level in “app.yaml”:

```
# app.yaml located in <application-directory>/app.yaml
EMailNotification:
  DailySummaryOn: monday, tuesday, wednesday, thursday, friday
  OnNewTasks: true
```

```
Language: de
```

Users are able to customize their notification settings in a workflow ui like the Portal. The content of the task email notifications can be customized by providing “Standard Processes”.

Html Theme

The look and feel of Html Dialogs is defined by its theme. You can change the appearance of any dialog on several scopes:

- Globally for all Html Dialogs: via “web.xml”.
- For a single application: in the “app.yaml”:

```
# app.yaml located in <application-directory>/app.yaml
Properties:
  jsf.primefaces.theme: modena-ivy
```

- Or at session level: via the `IvyPrimefacesThemeResolver`.

Passwords

You may want to encrypt sensitive data like a password in your configuration files. To do this you can enclose any value with “`${encrypt:}`”. The Axon.ivy engine will automatically encrypt and replace that value in file, when the configuration will be loaded. The system database password can be encrypted as follows:

```
# ivy.yaml
SystemDb:
  Password: "${encrypt:myPassword}"
```

There is a smooth “Secrets” integration, which is very useful in container environments such as Docker.

Overriding Configuration

Environment variables

Configuration entries of YAML files can be overridden with environment variables of the operating system. Configuration keys in YAML are hierarchic object trees separated by `:` characters. While the environment variable must be written uppercase and separated by `_` characters. You need also to prefix the environment variable with `IVY_`.

So to overwrite the `SystemDb:Url` of the “ivy.yaml” file, the environment variable `IVY_SYSTEMDB_URL` must be set.

Global application values

The “app.yaml” in the `/configuration` folder can be used to set global application configuration values that are applied to all applications on the engine.

Docker Containers

Container technology empowers you to pull up reproducible, documented and complete isolated infrastructures. Axon.ivy fully supports container environments such as Docker, Kubernetes or OpenShift. You can easily customize the configuration of an Axon.ivy engine by using system environment variables or by providing configuration files like the “ivy.yaml” file.

The following example will override the url of the system database configuration using environment variables:

```
docker run -e "IVY_SYSTEMDB_URL=jdbc:mysql://db:3306/AxonIvySystemDatabase" ...
```

Instead of using environment variables, you can simply provide an “ivy.yaml” file.

```
# ivy.yaml
SystemDb:
  Url: jdbc:mysql://db:3306/AxonIvySystemDatabase
```

```
docker run -v ivy.yaml:/etc/axonivy-engine/ivy.yaml ...
```

For further docker examples have a look at our docker-samples GitHub repository.

Secrets

You can use Docker Secrets to store passwords. Simply create a file in `/run/secrets` which has the same name as the configuration entry. For example, to provide `SystemDb:Password` as secret file you need to create the file `/run/secrets/ivy.SystemDb.Password`

Configuration File Reference

ivy.yaml

[engineDir]/configuration/ivy.yaml

```
#
# -----
# Axon.ivy Engine Configuration
# -----
#
# This file configures the Axon.ivy engine and its external systems.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html
#
# By default the engine is pre-configured to run in demo mode.
# To run an engine in a productive environment at least the system database
# must be configured.
#
# SECRETS / PASSWORDS:
# Any configuration value can be encrypted just by enclosing it with "${encrypt:}".
# * to encrypt the string myPassword write "${encrypt:myPassword}"
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-pa
#
# OVERRIDING:
# Any configuration value provided here can be set in alternative sources.
# * environment variables: of the operating system can set app config entries.
# Their key must be prefixed with 'IVY_'.
# E.g. use 'IVY_SYSTEMDB_URL' to override the jdbc driver url.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-o
#

# == System Database Settings ==
#
# Axon.ivy requires a System Database to store the state of running workflow applications.
#
# Unless you run the engine in Demo mode, a valid System DB driver, url and the user+passw
# that are able to connect to the database are mandatory.
#
# [Restart required]
SystemDb:
  # [MySQL]
  Driver: com.mysql.jdbc.Driver
  Url: jdbc:mysql://localhost:3306/AxonIvySystemDatabase
  # > jdbc:mysql://<host>[:<port>]/<database name>

  # [MariaDB]
```

```

Driver: org.mariadb.jdbc.Driver
Url: jdbc:mariadb://localhost:3306/AxonIvySystemDatabase
# > jdbc:mariadb://<host>[:<port>]/<database name>

# [MicrosoftSQL]
Driver: com.microsoft.sqlserver.jdbc.SQLServerDriver
Url: jdbc:sqlserver://localhost:1433;DatabaseName=AxonIvySystemDatabase
# > jdbc:sqlserver://<host>[:<port>];DatabaseName=<database name>

# [PostgreSQL]
Driver: org.postgresql.Driver
Url: jdbc:postgresql://localhost:5432/AxonIvySystemDatabase
# > jdbc:postgresql://<host>[:<port>]/<database system>

# The name of the user to connect to system database. E.g. root, sa, admin, ivy, AxonIvy
UserName: root

# The password of the user to connect to the system database.
Password: "${encrypt:1234}"

# If set to true the system database is automatically converted to the latest version du
Autoconvert: false

# Defines how long ivy should wait (in seconds) at startup for the availability of the c
BootTimeout: 60

# Additional driver specific connection properties.
DriverProperties:
  # [MySQL] Very likely to set if not ssl connection is used, to prevent warn logs
  useSSL: false
  # [MicrosoftSQL] Instance name of the MSSQL Server
  instanceName: SqlServer

# == Deployment Setting ==
#
Deployment:

  # Directory where the server watches for files to deploy.
  # https://dev.axonivy.com/doc/latest/EngineGuideHtml/administration.html#administration-
  #
  # You may want to use a UNC path to specify a remote network location.
  Directory: deploy

# == Data Settings ==
#
Data:

  # Folder where applications are stored, unless otherwise defined in the deployment.
  # If you change this path, proceed as follows...
  # 1. Stop the engine
  # 2. Change this path and move the existing applications to this new directory
  # 3. Start the engine
  # Absolute paths and relative paths are supported
  # [Restart required] for existing apps

```



```

AppDirectory: applications

# Root folder where application files are stored.
# A change in this setting will NOT automatically move existing application files to the
# A change will require to manually move existing files to the new directory.
# Absolute and relative (to the engine root directory) paths are supported.
# If not set the files will be stored underneath each application's file directory.
# [Restart required] for existing apps
FilesDirectory:

# Directory where the server writes temporary working files to.
# [Restart required]
WorkDirectory: work

# == Elasticsearch Settings ==
#
# Axon.ivy uses an Elasticsearch instance to provide a fast query interface against BusinessData.
# The bundled instance is started on demand, in a separate JVM, when an API request needs
#
# You can operate Axon.ivy with the bundled Elasticsearch server or with your own external
#
# [Restart required] except for Username and Password of ExternalServer
Elasticsearch:

# The bundled Elasticsearch server...
# - is started in a separate JVM when a feature requires BusinessData access.
# - reachable only on 'localhost' but the access is unprotected.
# - JVM arguments used to start the bundled Elasticsearch server can be
#   configured in the 'elasticsearch/config/jvm.options' file.
BundledServer:
  # The path to the directory where the bundled Elasticsearch server stores data.
  # It is recommend to configure a data directory that is located outside of the Engine
  # installation directory to ease the Engine migration to newer versions.
  DataPath: elasticsearch/data
  # The name of the cluster of the bundled Elasticsearch server.
  # Must not be defined as it is managed by the Axon.ivy Engine.
  ClusterName:

# Configure the URL of your own Elasticsearch server if you want to use it instead of the
#
# To install your own Elastic search installation follow these steps
# https://www.elastic.co/guide/en/elasticsearch/reference/current/setup.html
#
# Currently Axon.ivy supports Elasticsearch server versions in the 5.5.x range.
# If your Elasticsearch server is running on another host,
# the access to that instance has to be protected.
# You can achieve that with a front-end webserver like NGINX that enforces basic authentication.
ExternalServer:
  Url:
  Username:
  Password: "${encrypt:}"
  # Defines how long ivy should wait (in seconds) for the availability of the external Elasticsearch
  BootTimeout: 60

# For every business data type an Elasticsearch index will be created. E.g. for type ch.

```

```
Index:
  # The name prefix of the index to use to store business data.
  # If multiple ivy Engines use the same Elasticsearch server instance, you need to char
  NamePrefix: ivy.businessdata

# Configures the Elasticsearch client. The client is the ivy engine which communicates v
Client:
  # Maximum seconds to wait until a connection to Elastisearch can be established.
  ConnectTimeout: 10

  # Maximum seconds to wait for data sent by Elastisearch.
  # Raise this value if large datasets are expected.
  ReadTimeout: 30

# == Email Settings ==
#
Email:
  Server:
    Host: localhost
    Port: -1

    # Email address that will be used for emails sent by the server (e.g. task notificatio
    MailAddress: noreply@ivyserver.local
    User: guest
    Password: "${encrypt:}"

  #EncryptionMethod: NONE
  SSL:
    KeyAlias:
    UseKey: false

  DailyTaskSummary:
    # Time of day when the task summary mails will be sent.
    # Format is hh:mm. e.g. "02:00" or "14:15"
    TriggerTime: "00:00"

# == Show Error Messages To End Users Settings ==
#
# When an error occurs while processing a user request an error screen is displayed to the
#
# The displayed error page can be customized for your needs:
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-file
#
Errors:
  # Whether stacktraces, detailed error reports, etc. should be shown to end users.
  #
  # By default (false) we only show a unique 'Error Id'. This 'Error Id' can be used to fi
  #
  # For security reasons normal users should not see technical implementation details.
  # But in development or pre-production environments it might be save to show the full er
  # details directly to the end user.
  ShowDetailsToEndUser: false
```

```

# == Persistence Setting ==
#
Persistence:
  JPA:
    # Persist ivyScript auto initialized fields with NULL values. Affects types are...
    # - ch.ivyteam.ivy.scripting.objects.Date
    # - ch.ivyteam.ivy.scripting.objects.DateTime
    # - ch.ivyteam.ivy.scripting.objects.Time
    # If this option is disabled auto initialized values are stored as before Axon.ivy 6.4
    defaultInitializedAsNull: true

# == Process Element Firing Statistic Settings ==
#
ProcessEngine:
  FiringStatistic:

    # If activated, a process element statistic is written periodically to the log-directo
    Active: false

    # Interval in seconds the 'process element statistic' is written to the log directory
    Interval: 300

# == SSL Client Settings ==
#
SSL:
  Client:
    # A key store is used to read client keys (certificates).
    # This is only required if a remote server requests a client certificate in order to a
    KeyStore:
      UseCustom: false
      KeyPassword: "${encrypt:changeit}"
      Algorithm: SunX509
      File: configuration/keystore.jks
      Password: "${encrypt:changeit}"
      Provider:
      Type: jks

    # A trust store is used to specify trusted server certificates or certificates of cert
    # An SSL client authenticates a server by using the certificates in a trust store.
    TrustStore:
      # The system trust store of the Java Runtime Environment (JRE) contains well known c
      UseSystem: true

      # The custom trust store contains certificates that are self signed or signed by an
      UseCustom: false
      Algorithm: PKIX
      File: configuration/truststore.jks
      Password: "${encrypt:changeit}"
      Provider:
      Type: jks

    # Full qualified class name of a trust manager class that is used to validate server
    # This manager is only considered if neither a custom nor a system trust store is us

```

```

ManagerClass:

# == Failure Behaviour ==
#
SystemTask:
  # Defines the behaviour in case a system task fails.
  # Valid behaviours are...
  # * FAIL_TASK_DO_RETRY
  # * FAIL_TASK_DO_NOT_RETRY
  # * DESTROY_TASK
  # * DESTROY_CASE
  Failure.Behaviour: FAIL_TASK_DO_RETRY

  # Interval in seconds between executions of the search job for system tasks.
  # The job searches system tasks that were not executed because of failures.
  SearchJob.Interval: 900

# == Thread Pools Settings ==
#
ThreadPool:
  # Executes process engine background operations like Database, Webservice calls, etc.
  BackgroundOperationExecutor:
    # Minimum number of threads
    CorePoolSize: 5
    # Maximum number of threads
    MaximumPoolSize: 200

  # Executes unscheduled jobs
  ImmediateJobExecutor:
    # Minimum number of threads
    CorePoolSize: 5
    # Maximum number of threads
    MaximumPoolSize: 50

  # Executes scheduled jobs
  ScheduledJobExecutor:
    # Minimum number of threads
    CorePoolSize: 5

# == Update Checker Settings ==
#
# When newer Axon.ivy versions are available a message will be displayed on the Axon.ivy E
# The update message contains information about the new versions and where those can be do
#
# While checking for new versions the following statistic information are sent to the upda
# These information are only used to improve the product:
# - Engine (version, up time)
# - Configuration (number of: cluster nodes, users, licenced users, applications, process
# - Licence information (number, organisation, individual)
# - Operating system information (name, version, architecture, number of processors)
# - System database (product name and version, driver, identification number)
# - Java memory information (maximum heap memory, maximum non heap memory)

```

```

# - JVM (Java virtual machine) information (version, vendor, name)
# - Host information (host name, SHA-256 hashes of IP address and MAC address to identify
#
UpdateChecker:
  # Whether Update notification messages are shown and statistic information are sent to
  Enabled: true

# == Admin Users ==
#
# List of administrators which will be created during engine startup
# Password and Email will be updated if the administrator already exist
# Email is used to send info mails like license expiration
#
# Default administrator in demo mode is called admin with password admin
# [Restart required]
Administrators:

  # Example admin user with name myAdmin and password mySecret
  myAdmin:
    Password: "${encrypt:mySecret}"
    Email: info@localhost

# == Security Systems ==
#
# List of Security Systems.
# A security system defines how users and roles are managed.
# Security systems that are configured here can be used by applications.
# !! If you change a security system then all users that are no longer defined by the change
# !! SecuritySystem changes are immediately reloaded and a user synchronization is executed
# !! Switching from Microsoft Active Directory or Novell eDirectory to Axon.ivy Security System
# !! Tasks assigned to the deleted users are moved to the UNASSIGNED state and has to be manually
#
SecuritySystems:

  # Example security system with name mySecuritySystem
  mySecuritySystem:
    # [Axon.ivy Security System]
    # The Axon.ivy Security System manages the user and roles in the system database.
    # No additional configuration is needed.
    Provider: "ivy Security System"

    # [Microsoft Active Directory]
    # The Microsoft Active Directory security system uses LDAP to import users and user roles
    # You should also configure at least the properties Url, UserName, Password and Default
    Provider: "Microsoft Active Directory"

    # ["Novell eDirectory"]
    # The Novell eDirectory security system uses LDAP to import users and user role relationships
    # You should also configure at least the properties Url, UserName, Password and Default
    Provider: "Novell eDirectory"

  Connection:
    # Url to the naming and directory service

```

```

Url: ldap://localhost:389

# How to authenticate to the naming and directory service
# none = no authentication (default if Username/Password NOT configured)
# simple = user name and password is used (default if Username/Password is configured)
AuthenticationKind: simple

# User name to authenticate to the naming and directory service (java.naming.security
# Valid formats are...
# - LDAP Distinguished Name (RFC 4514) like cn=Administrator,dc=axonivy,dc=com
# - Active Directory user name like Administrator@axonivy.com
UserName:

# Password to authenticate to the naming and directory service (java.naming.security
Password: "${encrypt:}"

# Use a connection pool to store established LDAP connections
UseLdapConnectionPool: false

# Here you can configure additional environment properties for the LDAP context.
Environment:
# How to handle LDAP aliases. Possible values are... always, never, finding, search
# https://docs.oracle.com/javase/jndi/tutorial/ldap/misc/aliases.html
"java.naming.ldap.derefAliases": always

# Specifying the security protocol. If this property is unspecified, the behaviour
# https://docs.oracle.com/javase/jndi/tutorial/ldap/security/ssl.html
"java.naming.security.protocol":

# Specifying how referrals encountered by the service provider are to be processed
# https://docs.oracle.com/javase/jndi/tutorial/ldap/referral/index.html
"java.naming.referral": follow

Binding:
# Default Context to import from.
# The security system only sees and can import objects below the default context.
# Normally, you want to see and import all users of a security system then set the d
# If you want to import only users from a certain department or location, then you c
# See also EverybodyUserGroupName and UserFilter to control/filter the users that ar
# Format = LDAP Distinguished Name (RFC 4514) like dc=axonivy,dc=com or ou=ivyteam,dc=
DefaultContext:

# If configured, then the security system imports only the users that are members of
# See also DefaultContext and UserFilter to control/filter the users that are import
# Format = LDAP Distinguished Name (RFC 4514) of a user group like cn=AxonIvyUser,ou=
ImportUsersOfGroup:

# The security system only imports users that match the given filter.
# See also DefaultContext and EverybodyUserGroupName to control/filter the users tha
# Format = LDAP Search Filter (RFC 4515)
# [Microsoft Active Directory]
UserFilter: "(&(objectClass=user)(!(objectClass=computer)))"
# [Novell eDirectory]
UserFilter: "objectClass=inetOrgPerson"

UserAttribute:

```

```

# The LDAP attribute that stores the name of a user
# [Microsoft Active Directory]
Name: sAMAccountName
# [Novell eDirectory]
Name: uid

# The LDAP attribute that stores the full name of a user
# [Microsoft Active Directory]
FullName: displayName
# [Novell eDirectory]
FullName: fullName

# The LDAP attribute that stores the mail address of a user
EMail: mail

# The LDAP attribute that stores the language of a user
Language:

# Here you can specify a list of additional LDAP attributes that are imported and av
Properties:
  # Maps a user property to an LDAP attribute
  # In the example below 'phoneNumber' is the name of the user property.
  # The value of the property is imported from the LDAP attribute 'phone' of the use
  phoneNumber: phone

Membership:
# The LDAP attribute that stores the user groups a user is member of
# [Microsoft Active Directory]
UserMemberOfAttribute: memberOf
# [Novell eDirectory]
UserMemberOfAttribute: groupMembership

# Should the security system use the LDAP attribute configured in UserMemberOfAttrib
# Sometimes this LDAP attribute is not available because of security concerns.
# If you set this to false, then the security system will import the user role membe
# [Microsoft Active Directory]
UseUserMemberOfForUserRoleMembership: true
# [Novell eDirectory]
UseUserMemberOfForUserRoleMembership: false

# The LDAP attribute that stores the user groups a user group is member of
# [Microsoft Active Directory]
UserGroupMemberOfAttribute: memberOf
# [Novell eDirectory]
UserGroupMemberOfAttribute: groupMembership

# The LDAP attribute that stores the members (user, user groups) of a user group
# [Microsoft Active Directory]
UserGroupMembersAttribute: member
# [Novell eDirectory]
UserGroupMembersAttribute: uniqueMember

# Does the security system has to traverse nested groups (groups that are members of
# Some external security systems provide all users on the member attribute of a user
# [Microsoft Active Directory]
TraverseNestedGroups: true
# [Novell eDirectory]

```

```
TraverseNestedGroups: false
```

```
# The number of objects the security system can read in one LDAP request
PageSize: 500
```

```
# Time of day when the security system will synchronize the users.
# Format is hh:mm. e.g. "02:00" or "14:15"
UpdateTime: "00:00"
```

app.yaml

[engineDir]/configuration/app.yaml

```
#
# -----
# Axon.Ivy Application Configuration
# -----
#
# This files defines the configuration for its application.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-file
#
# By default applications are pre-configured to run without any dependencies.
# However in productive enviroments applications often interact with many
# external system such a Mail Servers (SMTP) or Directory services (LDAP).
#
# The 'defaults/app.yaml' serves as template that can be copied into
# an application directory as 'app.yaml' file.
# However 'app.yaml' can be deployed as part of the application projects.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/administration.html#administration-de
#
# SECRETS / PASSWORDS:
# Any configuration value can be encrypted just by enclosing it with "${encrypt:}".
# * to encrypt the string myPassword write "${encrypt:myPassword}"
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-pa
#
# OVERRIDING:
# Any configuration value provided here can be set in alternative sources.
# * environment variables: of the operating system can set app config entries.
#   Their key must be prefixed with 'IVY_APPLICATIONS_MYAPPNAME_'.
#   E.g. use 'IVY_APPLICATIONS_MYAPPNAME_SECURITYSYSTEM' to override the security system.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-c
# * ivy.yaml: can contain app specific entries, by placing them under the 'Applications' r
#   Applications:
#     myAppName:
#       SecuritySystem: mySecuritySystem
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-f
#
# == Data Settings ==
#
Data:
# Application folder where application files are stored. It overrides the root file fold
# A change in this setting will NOT automatically move existing application files to the
```



```

# A change will require to manually move existing files to the new directory.
# Absolute and relative (to the engine root directory) paths are supported.
# If not set the files will be stored in an application specific directory underneath the
# [Restart required] for existing apps
FilesDirectory:

# == Security System ==
#
# A security system manages users and roles and must be defined in ivy.yaml with a name.
# Here you can reference those security system by its name. If no security system is defined
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-user
# !! If you change the security system of an application then all users that are no longer
# !! Tasks assigned to the deleted users are moved to the UNASSIGNED state and has to be manually
#
SecuritySystem:

# == Environment ==
#
# Environments can be defined in ivy projects. Here you can activate a specific environment
#
ActiveEnvironment: Default

# == EMail Notification Settings ==
#
# These email notification settings will be applied to all users of an application.
# Users still have the option to customize their e-mail notification settings for themselves
#
EMailNotification:
  # Whether users should receive a mail when a new task is assigned. Possible values are:
  OnNewTasks: false

  # On which days of the week the users should receive a daily task summary.
  # Possible values are: never, always, monday, tuesday, wednesday, thursday, friday, saturday
  # Any combination of weekdays is allowed.
  # In ivy.yaml you can configure when the email is sent EMail:DailyTaskSummary:TriggerTime
  DailySummaryOn: never

  # Language of the emails. You can specify a locale. e.g. de, de_CH, de_AT, de_DE, en, en_US
  Language: en

# == Standard Processes ==
#
# Standard processes are a set of predefined processes, which you can customize in your ivy project
# To enable these custom processes, the library id of the ivy project must be specified here
# The library id is <group-id>:<project-id> from the ivy project deployment definition.
# e.g the library id of the portal template is "ch.ivyteam.ivy.project.portal:portalTemplate"
#
StandardProcess:

# https://dev.axonivy.com/doc/latest/EngineGuideHtml/administration.html#administration-

```

```

DefaultPages:

# https://dev.axonivy.com/doc/latest/EngineGuideHtml/administration.html#administration-
MailNotification:

# == Properties ==
#
# Application properties can be queried by ivy projects and allows ivy developers to make
#
Properties:

# JSF Primeface Theme that is used by HTML Dialogs.
# Available themes:
# ivy, modena-ivy, afterdark, afternoon, afterwork, aristo, black-tie blitzer, bluesky,
# delta, dot-luv, ggplant, excite-bike, flick, glass-x, home, hot-sneaks, humanity, le-
# redmond, rocket, sam, smoothness, south-street, start, sunny, swanky-purse, trontasti
jsf.primefaces.theme: modena-ivy

# == Global Variables ==
#
# Global variables are defined in ivy projects.
# All of those can be overridden independently of the environment.
#
GlobalVariables:
  myGlobalVariable: value

# == Databases ==
#
# Databases are defined in ivy projects with a name.
# Connection details from those databases can be overridden independently of the environme
#
Databases:

# This is an example configuration for the database with the name myDb.
myDb:
  Url: "jdbc:mysql://localhost:3306/myDbName"
  Driver: com.mysql.jdbc.Driver
  UserName: admin
  Password: "${encrypt:1234}"
  MaxConnections: 5

# Properties are merged with higher priority with those from the project.
Properties:
  name: value

# == RestClients ==
#
# Rest Clients are defined in ivy projects with a name.
# Any configuration from those clients can be overridden independently of the environment k
#

```

```
RestClients:

# This is an example configuration for the rest client with the name myRestClient.
myRestClient:
  Url: "http://localhost:8080"

# If defined, all features from the project will be completely replaced.
Features:
  - ch.ivyteam.ivy.rest.client.mapper.JsonFeature
  - ch.ivyteam.ivy.rest.client.authentication.HttpBasicAuthenticationFeature

# Properties are merged with higher priority with those from the project.
Properties:
  username: admin
  password: "${encrypt:1234}"
  name: value

# == WebserviceClients ==
#
# Web Service Clients are defined in ivy projects with a name.
# Any configuration from those clients can be overridden independently of the environment
#
WebserviceClients:

# This is an example configuration for the soap web service client with the name myWebService
myWebService:

# If defined, endpoint urls will be completely replaced per port type with those from
Endpoints:

# name of the port type, which is defined in the project.
myPortType:
  - "http://localhost:8088"
  - "http://webservice/api/soap"

# If defined, all features from the project will be completely replaced.
Features:
  - ch.ivyteam.ivy.webservice.exec.cxf.feature.HttpBasicAuthenticationFeature
  - ch.ivyteam.ivy.webservice.exec.cxf.feature.ProxyFeature

# Properties are merged with higher priority with those from the project.
Properties:
  username: admin
  password: "${encrypt:1234}"
  name: value

# Authentication property for the legacy axis stack
# Possible values for axis 1: NONE, HTTP_BASIC
# Possible values for axis 2: NONE, HTTP_BASIC, HTTP_DIGEST, NTLM
authType: NONE
```

ivy.webservice.yaml

[engineDir]/configuration/defaults/ivy.webservice.yaml

```

#
# -----
# Axon.ivy Web Server Configuration
# -----
#
# This file is a template to configure the internal Web Server of the Axon.ivy engine.
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html
#
# Copy contents of this template to 'configuration/ivy.yaml' before adjusting them to your
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-file
#
# By default this configuration enables all available features
# of the Axon.ivy engine so that all capabilities that might are used
# by a workflow project are accessible.
#
#
# OVERRIDING:
# Any configuration value of this file can be set in alternative sources.
# * environment variables: of the operating system can set app config entries.
#   Their key must be prefixed with 'IVY_'.
#   E.g. use 'IVY_FRONTEND_PORT' to override the front-end webserver port.
#   https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-file
#

# == Front-end Web Server (Reverse Proxy, IIS, Apache, Load balancer, ...) Settings ==
#
# Links generated by Axon.ivy often contain absolute links to the ivy server (e.g. for mail)
# If your Axon.ivy engine is only accessible for clients trough a front-end webserver,
# its host, port and protocol of it must be specified.
Frontend:
  # Hostname of the accessible web server
  HostName: localhost

  # Port of the accessible web server
  Port: 443

  # Protocol of the accessible web server
  Protocol: https

# == REST Service Settings ==
#
# Configures the RESTful services provided.
# [Restart required]
REST.Servlet:
  # Controls the REST servlet interface. If disabled no REST resources will be accessible.
  # Calls to remote REST services are still possible.
  Enabled: true

  # Provides the general CSRF protection via 'X-Requested-By' header for REST services.
  CSRF.Protection: true

  # Provide the REST resources for the mobile app under '{application}/api/workflow'.
  MobileWorkflow.API: true

```

```
# Allows the service developer to get diagnostic information about request processing by
# Those diagnostic/tracing information are returned in response headers (X-Jersey-Tracing)
# On productive environments this feature should not be turned on.
# Valid values are either "OFF", "ON_DEMAND" or "ALL"
Tracing: "OFF"

# == Miscellaneous Settings ==

# Session identifier will be renewed on login to prevent the 'Session Fixation' attack.
Session.RenewIdOnLogin: true

# Name of the Ivy servlet context. Use a simple name without any special characters (e.g.
# [Restart required]
WebServer.IvyContextName: ivy

# Disable it if you don't use the Mobile Offline Dialog feature.
# [Restart required]
OfflineDialog.Enabled: true

# == Web Server Connector Settings ==
# https://tomcat.apache.org/tomcat-8.5-doc/config/http.html
Connector:
# [Restart required]
HTTP:
  Enabled: true
  Port: 8080
  AcceptCount: 100
  Address:
  AllowTrace: false
  BufferSize: 2048
  CompressableMimeType: text/html,text/xml,text/plain
  Compression: off
  ConnectionLinger: -1
  ConnectionTimeout: 60000
  DisableUploadTimeout: true
  EmptySessionPath: false
  EnableLookups: false
  MaxHttpRequestSize: 8192
  MaxKeepAliveRequests: 100
  MaxPostSize: 2097152
  MaxSavePostSize: 4096
  MaxSpareThreads: 50
  MaxThreads: 200
  MinSpareThreads: 4
  NoCompressionUserAgents:
  ProxyName:
  ProxyPort:
  RedirectPort: 8443
  RestrictedUserAgents:
  Server:
  SocketBuffer: 9000
  Strategy: lf
  TcpNoDelay: true
  ThreadPriority: 5
```

```

URIEncoding: UTF-8
UseBodyEncodingForURI: false
UseIPVHosts: false
XpoweredBy: false

# [Restart required]
HTTPS:
  Enabled: true
  Port: 8443
  AcceptCount: 1000
  Address:
  Algorithm:
  AllowTrace: false
  BufferSize: 2048
  Ciphers:
  ClientAuth: false
  CompressableMimeType: text/html,text/xml,text/plain
  Compression: off
  ConnectionLinger: -1
  ConnectionTimeout: 60000
  DisableUploadTimeout: true
  EmptySessionPath: false
  EnableLookups: false
  KeyAlias:
  KeystoreFile: configuration/keystore.jks
  KeystorePass:
  KeystoreType:
  MaxHTTPHeaderSize: 8192
  MaxKeepAliveRequests: 100
  MaxPostSize: 2097152
  MaxSavePostSize: 4096
  MaxSpareThreads: 50
  MaxThreads: 200
  MinSpareThreads: 4
  NoCompressionUserAgents:
  ProxyName:
  ProxyPort:
  RedirectPort: 8443
  RestrictedUserAgents:
  Server:
  SocketBuffer: 9000
  SslProtocol: TLS
  Strategy: lf
  TcpNoDelay: true
  ThreadPriority: 5
  TruststoreFile:
  TruststorePass:
  TruststoreType:
  URIEncoding: UTF-8
  UseBodyEncodingForURI: false
  UseIPVHosts: false
  XpoweredBy: false

# [Restart required]
AJP:
  Enabled: true
  Port: 8009
  Address:

```

```

AllowTrace: false
BackLog: 100
BufferSize: 2048
ConnectionTimeout: 60000
EmptySessionPath: false
EnableLookups: false
MaxPostSize: 2097152
MaxSavePostSize: 4096
MaxSpareThreads: 50
MaxThreads: 200
MinSpareThreads: 4
PacketSize: 8192
ProxyName:
ProxyPort:
RedirectPort: 8443
TcpNoDelay: true
ThreadPriority: 5
TomcatAuthentication: false
URIEncoding: UTF-8
UseBodyEncodingForURI: false
UseIPVHosts: false
XpoweredBy: false

```

ivy.cache.properties

[engineDir]/configuration/ivy.cache.properties

```

#
# -----
# Axon.ivy System Database Cache
# -----
#
# This file configures how data, loaded from the internal system database, are cached in t
# https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html
#
# This file rarely has to be adjusted if a concrete performance issue has been identified.
#
#
# OVERRIDING:
# Any configuration value provided here can be set in alternative sources.
# * environment variables: of the operating system can set cache entries.
#   Their key must be prefixed with 'IVY_SYSTEMDB_CACHE_'.
#   E.g. use 'IVY_SYSTEMDB_CACHE_CH_IVYTEAM_IVY_CASEMAP_INTERNAL_DATA_CASEMAPBUSINESSCASE
#   https://dev.axonivy.com/doc/latest/EngineGuideHtml/configuration.html#configuration-c
#
#
# == System Database Cache Settings ==
#
# ch.ivyteam.ivy.casemap.internal.data.CaseMapBusinessCaseData.CountLimit=1000
# ch.ivyteam.ivy.casemap.internal.data.CaseMapBusinessCaseData.UsageLimit=57600
# ch.ivyteam.ivy.casemap.internal.data.CaseMapEventData.CountLimit=1000
# ch.ivyteam.ivy.casemap.internal.data.CaseMapEventData.UsageLimit=57600
# ch.ivyteam.ivy.cm.internal.data.BinaryContentData.CountLimit=30000
# ch.ivyteam.ivy.cm.internal.data.BinaryContentData.UsageLimit=360000
# ch.ivyteam.ivy.cm.internal.data.ContentObjectData.CountLimit=10000

```

```
# ch.ivyteam.ivy.cm.internal.data.ContentObjectData.UsageLimit=360000
# ch.ivyteam.ivy.cm.internal.data.ContentObjectValueData.CountLimit=30000
# ch.ivyteam.ivy.cm.internal.data.ContentObjectValueData.UsageLimit=360000
# ch.ivyteam.ivy.cm.internal.data.StringContentData.CountLimit=30000
# ch.ivyteam.ivy.cm.internal.data.StringContentData.UsageLimit=360000
# ch.ivyteam.ivy.cm.internal.data.TextContentData.CountLimit=30000
# ch.ivyteam.ivy.cm.internal.data.TextContentData.UsageLimit=360000
# ch.ivyteam.ivy.security.internal.data.AccessControlData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.AccessControlData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.RichDialogUserContextData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.RichDialogUserContextData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.RolePropertyData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.RolePropertyData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.UserAbsenceData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.UserAbsenceData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.UserData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.UserData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.UserLocationData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.UserLocationData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.UserPropertyData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.UserPropertyData.UsageLimit=57600
# ch.ivyteam.ivy.security.internal.data.UserSubstituteData.CountLimit=1000
# ch.ivyteam.ivy.security.internal.data.UserSubstituteData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.AdditionalPropertyData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.AdditionalPropertyData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.BusinessCaseDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.BusinessCaseDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.CaseData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.CaseData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.EventLogCaseHistoryData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.EventLogCaseHistoryData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.EventLogData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.EventLogData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.EventLogDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.EventLogDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.EventLogStatusData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.EventLogStatusData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.EventLogTaskHistoryData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.EventLogTaskHistoryData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.IntermediateEventElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.NoteData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.NoteData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.PageArchiveData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.PageArchiveData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.PageElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.PageElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.SignedTaskData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.SignedTaskData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.SignalEventData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.SignalEventData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.SignalEventDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.SignalEventDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.StartElementData.CountLimit=1000
```



```
# ch.ivyteam.ivy.workflow.internal.data.StartElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.StartEventElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.StartEventElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.StartSignalEventElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.StartSignalEventElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.StartTaskDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.StartTaskDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskBoundarySignalEventReceiverData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskBoundarySignalEventReceiverData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskDataData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskDataData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskEndData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskEndData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskLocationData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskLocationData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskStartData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskStartData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.TaskSwitchEventData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.TaskSwitchEventData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.WebServiceProcessData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.WebServiceProcessData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.WebServiceProcStartElementData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.WebServiceProcStartElementData.UsageLimit=57600
# ch.ivyteam.ivy.workflow.internal.data.WorkflowEventData.CountLimit=1000
# ch.ivyteam.ivy.workflow.internal.data.WorkflowEventData.UsageLimit=57600
```

log4jconfig.xml

[engineDir]/configuration/log4jconfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<!--
=====
Axon.ivy Logging Configuration
=====

This is the logging configuration file for Axon.ivy Engine.
It defines which log messages are logged (category/priority) and where the logs are written.

Logging in Axon.ivy Engine is based on a 3rd party library called Log4J.
http://logging.apache.org/log4j

-->
<log4j:configuration debug="false" xmlns:log4j="http://jakarta.apache.org/log4j/">

  <!-- appender that writes log messages to 'logs/ivy.log' -->
  <appender name="FileLog" class="org.apache.log4j.DailyRollingFileAppender">
    <param name="Threshold" value="INFO"/>
    <param name="File" value="${user.dir}/logs/ivy.log"/>
    <param name="DatePattern" value="'. 'yyyy-MM-dd"/>
    <layout class="ch.ivyteam.log.layout.IvyLog4jLayout">
      <param name="DateFormat" value="yyyy-MM-dd HH:mm:ss.SSS"/>
    </layout>
  </appender>
</log4j:configuration>
```

```

</appender>

<!-- appender that writes log messages with priority WARN or higher to the console -->
<appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
  <param name="Threshold" value="WARN"/>
  <layout class="ch.ivyteam.log.layout.IvyLog4jLayout">
    <param name="DateFormat" value="HH:mm:ss.SSS"/>
    <param name="ContextPrinting" value="false"/>
    <param name="FixedCategoryLength" value="40"/>
  </layout>
</appender>

<!-- appender that writes configuration changes to 'logs/config.log' -->
<appender name="ConfigLog" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="File" value="\${user.dir}/logs/config.log"/>
  <param name="DatePattern" value="'. 'yyyy-MM-dd"/>
  <layout class="ch.ivyteam.log.layout.IvyLog4jLayout">
    <param name="DateFormat" value="yyyy-MM-dd HH:mm:ss.SSS"/>
    <param name="ContextPrinting" value="false"/>
  </layout>
</appender>

<!-- appender that writes log messages to 'logs/runtime.log' -->
<appender name="RuntimeLog" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="File" value="\${user.dir}/logs/runtime.log"/>
  <param name="DatePattern" value="'. 'yyyy-MM-dd"/>
  <layout class="ch.ivyteam.log.layout.IvyLog4jLayout">
    <param name="DateFormat" value="yyyy-MM-dd HH:mm:ss.SSS"/>
  </layout>
</appender>

<!-- prevent "ClientAbortException: java.io.IOException: Broken pipe" from filling the l
<category name="org.apache.myfaces.application.ResourceHandlerImpl" class="ch.ivyteam.lo
  <priority value="FATAL"/>
</category>

<!-- disable deprecated integer API warnings -->
<!--
<category name="ch.ivyteam.ivy.persistence.restricted.TableKeyCompatibilityConvertor" cl
  <priority value="ERROR"/>
</category>
-->

<!--
  Enables web service client SOAP message logging for a certain application and process
  Replace {application} and {process_model} in the logger name below with the name of th
-->
<!--
<category name="runtimelog.{application}.{process_model}.web_service" class="ch.ivyteam.
  <priority value="DEBUG"/>
  <appender-ref ref="RuntimeLog"/>
</category>
-->

<!--
  Enables Rest client message logging for a certain application and process model.
  Replace {application} and {process_model} in the logger name below with the name of th
-->

```

```

<!--
<category name="runtime.log.{application}.{process_model}.rest_client" class="ch.ivyteam.
  <priority value="DEBUG"/>
  <appender-ref ref="RuntimeLog"/>
</category>
-->

<!--
  Config Monitoring: Writes an audit log that allows to track configuration changes over
  These logs are not passed to the root logger (additivity="false")
-->
<logger name="ch.ivyteam.ivy.audit.config" additivity="false">
  <level value="INFO"/>
  <appender-ref ref="ConfigLog"/>
</logger>

<!-- every log message with priority INFO or higher is passed to the file and console ap
<root>
  <level value="INFO" />
  <appender-ref ref="FileLog"/>
  <appender-ref ref="ConsoleAppender"/>
</root>

</log4j:configuration>

```

web.xml

[engineDir]/webapps/ivy/WEB-INF/web.xml

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
=====
  Configures the embedded Tomcat Webserver of Axon.ivy
=====

Please keep the web.xml file on the designer and engine synchronous
to have the same settings on designer and engine,
because this file is not deployed from the designer to the engine.

See apache tomcat documentation for more information about this configuration:
http://tomcat.apache.org/tomcat-8.5-doc/config/

After a change in the web.xml a restart of Axon.ivy is required
to apply the new configuration.

-->
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="false">

<!-- ===== Html Dialog Configuration ===== -->

<!--
THEME:
To set the primefaces theme (default is 'modena-ivy', was 'ivy' with 5.1)

```

```

remove the comment markers from around the context-param below
See available themes: http://primefaces.org/themes.html
-->
<!--
<context-param>
  <param-name>primefaces.THEME</param-name>
  <param-value>#{ivyPrimefacesThemeResolver.getTheme('modena-ivy')}</param-value>
</context-param>
-->

<!--
#{ivyThemeResolver.getThemes()} returns a list of all by default available themes.
If additional customer specific themes are installed they can be configured as comma sepa
#{ivyThemeResolver.getThemes()} will then additionally also return the configured custome
-->
<!--
<context-param>
  <param-name>primefaces.customer.themes</param-name>
  <param-value></param-value>
</context-param>
-->

<!-- ===== Error pages ===== -->
<!--
  Custom error pages can be added with error-page elements bellow.
  The referenced error-page must be placed in the folder 'webapps/ivy'.

  The pre-configured default error page is:

<error-page>
  <location>/ivy-error-page.xhtml </location>
</error-page>

  By adding the <exception-type> tag to the <error-page> configuration
  it is also possible to configure a specific error page for status codes
  or kind of exceptions:

<error-page>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/custom-exception-error-page.xhtml</location>
</error-page>
<error-page>
  <error-code>404</error-code>
  <location>/custom-404-error-page.xhtml</location>
</error-page>

Implementation:
Use the 'ErrorPageMBean' to retrieve information about the thrown exception and the envi
https://developer.axonivy.com/doc/latest/PublicAPI/ch/ivyteam/ivy/webserver/ErrorPageMBean
-->
<!--
<error-page>
  <error-code>404</error-code>
  <location>/custom-404-error-page.xhtml</location>
</error-page>
-->

```

```

<!-- ===== Default Session Configuration ===== -->
<session-config>
  <!--
  session-timeout: [default=30]

  Defines the amount of time in minutes after which an inactive user session will be closed.
  Closing sessions means that server side state (e.g. Html Dialog instance) is flushed.
  -->
  <session-timeout>30</session-timeout>

  <!--
  cookie-config/secure: [default=false]

  Enable the secure flag when accessing the Webserver over HTTPS (strongly recommended).
  When enabled the session cookie is only transmitted over HTTPS and not over HTTP.
  -->
  <!--
  <cookie-config>
    <secure>true</secure>
  </cookie-config>
  -->
</session-config>

<!-- ===== Security Headers ===== -->
<!--
<!-- Some commonly recommended HTTP Security Headers are configured here -->
<!-- for the /ivy web application. -->
<!-- These Security Headers are added on the HTTP Responses -->
<!-- to the Client Browser. -->
<!-- But not all Security Headers are supported by all Web browsers. -->
<!-- See: https://tomcat.apache.org/tomcat-8.5-doc/config/filter.html -->
<!--
<!-- |=====|=====| -->
<!-- | HEADER          | VALUE          | -->
<!-- |=====|=====| -->
<!-- | X-Frame-Options   | SAMEORIGIN    | -->
<!-- | X-XSS-Protection  | 1; mode=block | -->
<!-- | X-Content-Type-Options | nosniff       | -->
<!-- |=====|=====| -->
<!-- -->
<filter-mapping>
  <filter-name>httpSecurityHeaders</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
<filter>
  <filter-name>httpSecurityHeaders</filter-name>
  <filter-class>org.apache.catalina.filters.HttpHeaderSecurityFilter</filter-class>
  <init-param>
    <param-name>antiClickJackingOption</param-name>
    <param-value>SAMEORIGIN</param-value>
  </init-param>
  <init-param>
    <param-name>xssProtectionEnabled</param-name>
    <param-value>true</param-value>
  </init-param>

```

```

<init-param>
  <param-name>blockContentTypeSniffingEnabled</param-name>
  <param-value>true</param-value>
</init-param>
</filter>
</web-app>

```

context.xml

[engineDir]/webapps/ivy/META-INF/context.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
=====
  Configures Valves and Realms of the embedded Tomcat Webserver
=====

Please keep the context.xml file on the designer and engine in sync
to have the same settings on designer and engine
as this file is not deployed from the designer to the engine

See apache tomcat documentation for more information about context configuration:
https://tomcat.apache.org/tomcat-8.5-doc/config/context.html

-->
<Context antiResourceLocking="false" privileged="true" >

  <!-- ===== Tomcat Valves ===== -->

  <!--
    Limits the access to the ivy application to clients connecting from localhost.
  -->
  <!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|:1|0:0:0:0:0:0:0:1" />
  -->

  <!--
    Creates an access log entry for each request against the ivy application.
  -->
  <!--
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="access_log." suffix=".txt"
    pattern="%h %l %u %t &quot;%r&quot;; %s %b" />
  -->

  <!-- ===== Axon.ivy Valves ===== -->

  <!--
    SingleSignOnValve:

    Enables single sign on of the user given in a request header field.
    The name of the request header field can be configured in the attribute 'userNameHeader'

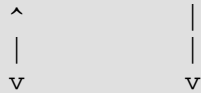
    !! Only use this Valve if you exclusively access Axon.ivy over the WebApplication Firewall
  -->

```

!! Otherwise this will be a security issue.

This Valve is useful if Axon.ivy is protected by a WebApplication Firewall (WAF) with a Identity and Access Management (IAM). Those systems will authenticate and authorize users. The identified user is then sent from the WAF to Axon.ivy using a HTTP request header.

WebBrowser ==> WAF ==> Axon.ivy



IAM ==> Active Directory

<https://developer.axonivy.com/doc/latest/EngineGuideHtml/integration.html#integration.w>

-->

<!--

<Valve className="ch.ivyteam.ivy.webserver.security.SingleSignOnValve" userNameHeader="user"

-->

<!-- ===== Custom Valves ===== -->

<!--

You can configure any third party valve or even your own implementation of a valve. A full valve sample implementation can be found on GitHub:

<https://github.com/ivy-samples/ivy-extension-demos/tree/master/ProcessingValve>

-->

</Context>

Chapter 5. Security

General

This chapter describes how to run an Axon.ivy Engine in a secure way. This is important when providing an ivy engine in a secure intranet environment and especially when making an engine accessible over the internet. Some parts might be done by the IT Operation provider.

There are at least seven important topics:

1. Run the Axon.ivy Engine behind a fully patched front-end server (like IIS, nginx) with restricted accessibility (paths, ports, users, etc.)
2. Only allow access to the URLs of your application / block access to system URLs
3. Don't allow direct access to the Axon.ivy Engine
4. Run the Axon.ivy Engine with a dedicated system user and Database users with limited access rights
5. Run the latest Axon.ivy Engine major version with all updates marked as security relevant.
6. Only serve users over HTTPS (configured on the front-end server (IIS/nginx))
7. Document and/or automate the server setup
8. Ensure that the provider performs daily backups (Database, relevant Engine folders)

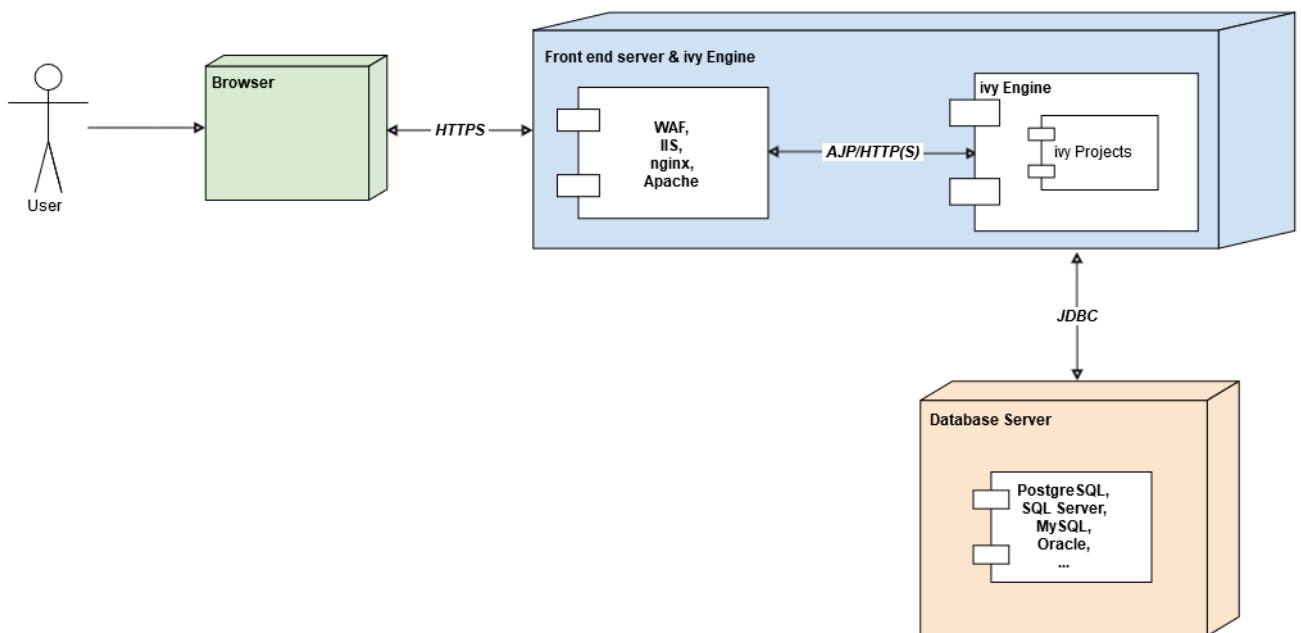


Figure 5.1. Deployment sample (with combined front-end server and ivy engine)

Front-end Server

A front-end server is a server over which a user accesses the Axon.ivy Engine. There are roughly three types of front-end servers:

- Web servers (e.g. Microsoft IIS, nginx, Apache HTTP Server)

- Web Application Firewalls (WAF)
- Custom cloud provider reverse proxies (often nginx based)

Port Configuration

Aim: Only allow communication with the Axon.ivy Engine through the front-end server.

Don't allow direct access to the Axon.ivy Engine ports (e.g. 8080, 8443, 8009) from the outside. Requests should always be performed through a front-end server (e.g. IIS). It's best to open up only the ports in the firewall that are really needed (most of the times (HTTP (80) and/or HTTPS (443)). It's best to automate these port tests for continuous security.

Test: Check if you can reach the ivy Engine over port 8080 (the port defined in the config).

Expected outcome: The Engine should not be reachable.



Warning

When using Ubuntu Linux as a server make sure to enable and configure the firewall (**ufw**) as it's not enabled by default.

Additional Security Headers

Following additional security headers are recommended.

Header name	Description
Strict-Transport-Security	Set this header if the Engine should only be accessed over HTTPS (strongly recommended). For more information, see: Strict-Transport-Security. Can be adjusted on the embedded Tomcat with the pre-configured <code>HttpHeaderSecurityFilter</code> in the "web.xml"
Content-Security-Policy	Set this header if you want to reduce the risk of having an exploitable Cross-site scripting (XSS) vulnerability. With a Content-Security-Policy you can define from which locations external resources can be loaded and if scripts embedded in HTML can be executed. For more information, see: Content Security Policy (CSP). A CSP example with the embedded Tomcat is available here: https://answers.axonivy.com/questions/2982
Referrer-Policy	Set this header if you want to control how or if the referrer is disclosed to other sites. For more information, see: Referrer-Policy

Table 5.1. Additional Security Headers

Path Configuration

Aim: Only allow paths from the front-end server to the Axon.ivy Engine that are required by your ivy projects to work correctly.

Test: Check if you can reach `http(s)://<front-end-server>/ivy/error` (and all other blocked URLs). Also check if you can access your process that should be available to end users. Note that the ivy servlet is not necessarily running under `/ivy/`. It is best to automate these tests for continuous security.

Expected Outcome: Forbidden paths should not be reachable (HTTP Status: 4xx). The process should be reachable for end users.

Example for a JSF-based Application: Filter following URL parts:

URL Part	Also blocks	Used for
/ivy/error		Displaying and administering Errors
/ivy/pro/System		System apps (Admin UI)
/ivy/pagearchive		Page Archive
/ivy/rd	/ivy/rdlib, /ivy/rdjmp	RIA (ULC)
/ivy/ulcload		ULC Load Tests
/ivy/ws/System/	PortalConnector Web Services	PortalConnector Web Services, used by Portal (!)
/ivy/api/		Workflow API, REST APIs
/ivy/wf/		Server Workflow UI
/ivy/info		Process overview page

Table 5.2. URL Parts to Block

Block URLs in IIS

Unfortunately, IIS does not seem to have support for white listing URLs, so we have to use a black list approach using request filtering, in which we block URL segments.



Warning

When using the request filter on IIS the URL parts are generally not allowed. A filter `/ivy/error` means that `/ivy/bla/ivy/error/test` is also not allowed (because it's a part of the URL).

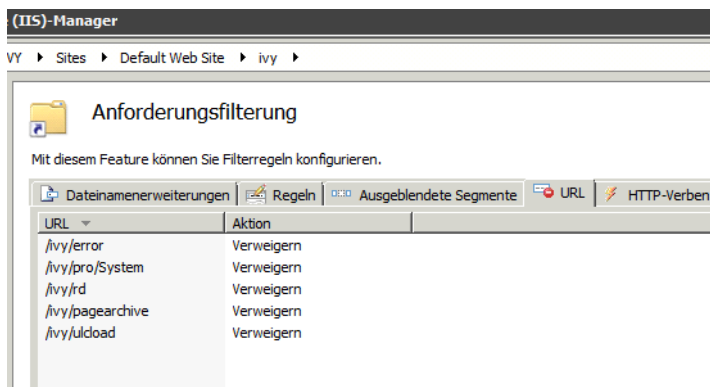


Figure 5.2. IIS Request Filter Config

After changing the configuration restart IIS and check that the URLs are not accessible anymore.

Block URLs in nginx

In the nginx server configuration, URLs can be blocked like this (repeat for the other URLs).

```
location /ivy/rd {
    deny all;
    return 403;
}
```

After changing the configuration restart nginx and check that the URLs are not accessible anymore.

Block URLs on the ivy Engine / Tomcat

Alternatively, URLs can also be blocked directly on the ivy Engine which utilizes and underlying Apache Tomcat for serving HTTP requests. Open the “web.xml” file in *webapps/ivy/WEB-INF/* in a text editor and add the following configuration inside the `<web-app>` tag (!) to block partial access to Rich Dialogs:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
  ...
  <security-constraint>
    <display-name>Restrict access to Richdialog URLs</display-name>
    <web-resource-collection>
      <web-resource-name>Restricted RichDialog</web-resource-name>
      <url-pattern>/rd/*</url-pattern>
    </web-resource-collection>
    <auth-constraint />
  </security-constraint>
  ...
</web-app>
```

After changing the configuration restart the ivy Engine and check that the URLs are not accessible anymore.



Tip

If the URLs are still accessible after blocking them in the web.xml make sure that you didn't include the servlet path (*/ivy*) in the URL pattern.

HTTPS

It is strongly recommended to protect the connection to the server with HTTPS, especially when transferring sensitive data (like passwords). HTTPS connections should be configured on the front-end server. Only current TLS/SSL settings (e.g. no SSLv3 or TLSv1.0) and only up-to-date systems (e.g. no OpenSSL with Heartbleed) should be used. Consult the manual of your front-end server for a secure HTTPS setup.

Remarks

- Current browsers mark login forms with password fields over HTTP as insecure.
- Enable the secure flag on session cookies when using HTTPS (in the “web.xml”).
- Enable HTTP Strict Transport Security (HSTS) (with the `httpHeaderSecurity` filter in the “web.xml” or in the configuration of the front-end server).

Axon.ivy Engine

Disable not required features

If certain features of the Engine are not required by the deployed projects, those features should be disabled.

Optional features

The following features can be disabled if they are not used by the deployed projects:

```
# ivy.webserver.yaml with all optional features disabled
REST:
```

```
# If none of the deployed projects provide REST APIs it is also possible to disable the v
Enabled: false

# REST resources for the mobile app under '{application}/api/workflow'
MobileWorkflow:
  # If the Mobile Workflow REST API is not used on your engine (e.g. by the Axon.ivy Mobil
  API: false

# If the Mobile Offline Dialogs are not used on your engine (most of the time) you can dis
OfflineDialogs:
  Enabled: false
```

Security Features

The following features impact the security and have a good default (depending on what's better for security) and should not be changed.

These feature flags are set in the “ivy.yaml” and “ivy.webserver.yaml”.

- `Errors.ShowDetailsToEndUsers` should always be set to **false**, so that no exception details are shown to end users.
- `WebServer.REST.CSRF.Protection` should always be set to **true**, so that the REST APIs require a Cross-Site Request Forgery (CSRF) token by default.
- `WebServer.RenewSessionIdOnLogin` should always be set to **true**, so that the session id is renewed after Login.

Only grant ivy Permissions where required

Only grant ivy permissions where the users or roles require them. Especially don't grant all permissions to everybody. Favor role-based permissions over user-based ones.

Security issues in the Axon.ivy Engine

If a security issue was found in the Axon.ivy Engine or Designer it is visible in the Release Notes, annotated with an exclamation mark (!):

- **!** - Critical: We strongly recommend to install this hotfix because it fixes a critical security issue!
- ***** - Recommended: We recommend to install this hotfix because it fixes a stability issue.
- **+** - Suggested/Optional: We only recommend to install this hotfix if you are running into the described issue.

Update the engine if your engine version is affected by an issue and it cannot be mitigated by a workaround.

Chapter 6. Integration

Introduction

We recommend to run Axon.ivy Engine behind a web front-end server (Apache httpd, Microsoft IIS, Reverse Proxy, Web Application Firewall, etc.).

In those cases the web front-end server receives all requests from the clients and forwards them to the Axon.ivy Engine which handles them. This allows to integrate the processes and applications that you are running on an Axon.ivy Engine into a company or web portal. Some web front-end server provide Single Sign On (SSO) functionality. The front-end server then is responsible to identify the user (either automatically or by login). After that the user is able to operate on all web sites that are integrated into the web front-end server.

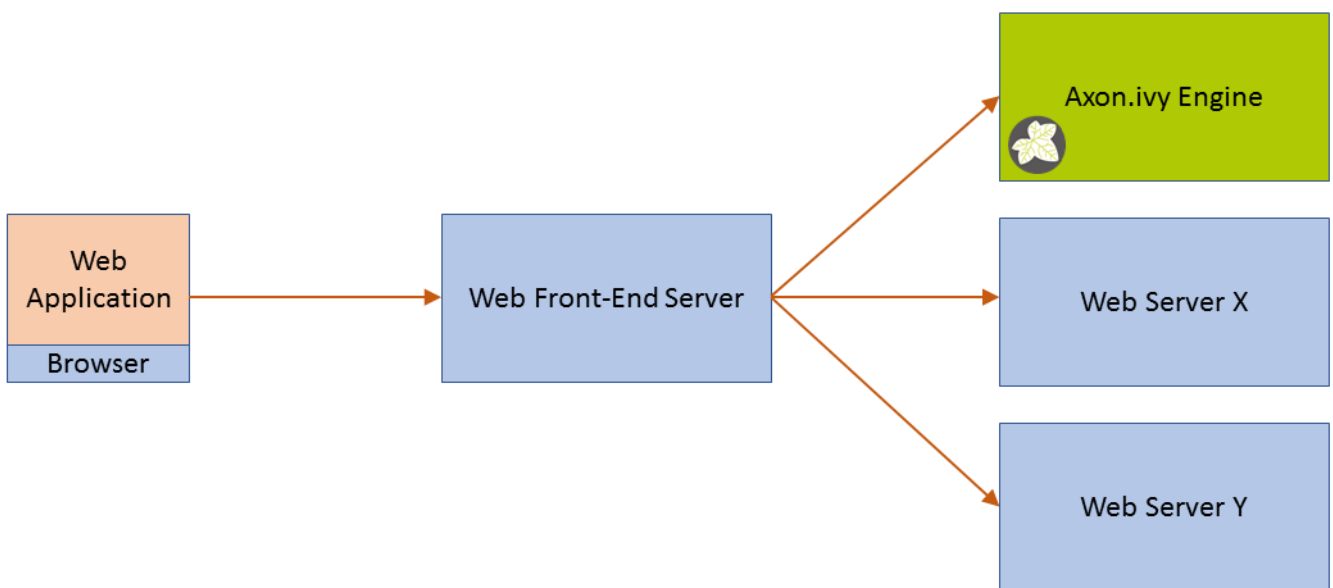


Figure 6.1. Web Front-End Server

The integration for Apache httpd and Microsoft IIS is technically solved by using Tomcat Connectors. More technical details about those connectors can be found on the Apache Tomcat web site.

Integration Directory

All necessary files that you need to integrate an Axon.ivy Engine into a Web Server can be found in the following directories inside the Axon.ivy Engine installation directory:

- Apache HTTP Server for Windows (x64): *misc/apache/*
- Apache HTTP Server for Linux (x64): *misc/apache/*
- IIS for Windows (x64): *misc/iis/*

The directory that matches the platform and webserver where you plan to integrate the Axon.ivy Engine will be called *integration directory* in this chapter.

The integration binaries for Linux are not delivered with the Axon.ivy Engine as it is best practice to use the Tomcat Connector binaries that are provided by the Linux distribution. See “Linux example configuration”

External base URL

Once Axon.ivy is served to clients via a front-end webserver, you must make the front-end webserver known as shown in the “ivy.webserver.yaml”. Axon.ivy will use this configuration to create absolute links that are accessible to clients (e.g. for links in task mails).

```
# sample ivy.yaml that configures an front-end webserver for clients.
# https://acme.com:443/ will be the absolute URL prefix for links generated by Axon.ivy
Frontend:
  HostName: acme.com
  Port: 443
  Protocol: https
```

Apache Integration

An Apache HTTP Server 2.x can be configured as web frontend of an Axon.ivy Engine. The communication between the Apache HTTP Server and the Tomcat from Axon.ivy is possible by using the Apache Tomcat Connector.

Windows example configuration

1. If your Apache HTTP Server is not running on the same host as the Axon.ivy Engine then the integration directory content must be copied to the host where your Apache HTTP Server is running.

- Copy the mod_jk binaries and the sample configuration files from the directory that matches your OS in `[[Axon.ivy Engine install dir]]/misc/apache` to the Apache Host under `C:\Program Files\ivy`

All next steps have to be done on the host where the Apache HTTP Server is running on.

2. Include the copied `jk_module` configuration in the `[[Apache Install Dir]]/conf/httpd.conf`. Add the following lines to do so:

```
# Axon.ivy Engine Integration
Include C:/Program Files/ivy/mod_jk.conf
```

3. Replace all `<path>` strings in the file `C:\Program Files\ivy\mod_jk.conf` so that the file reflects your local paths:

```
# Load mod_jk module
LoadModule      jk_module      c:/program files/ivy/mod_jk-1.2.42-httpd-2.4.so
# Where to find workers.properties
JkWorkersFile   c:/program files/ivy/workers.properties
# Where to put jk shared memory
JkShmFile       c:/program files/ivy/mod_jk.shm
# Where to put jk logs
JkLogFile       c:/program files/ivy/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel      info
# Select the timestamp log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "

# Mount the uri "/ivy/*" to the worker AxonIvyEngine.
JkMount /ivy/* AxonIvyEngine
```

4. If you have configured virtual hosts in your apache configuration you have to map the URI `/ivy/*` in all virtual host you want to integrate Axon.ivy Engine into. This can be done by copying the following line from the `mod_jk.conf` file to the appropriate virtual host definitions:

```
JkMount /ivy/* AxonIvyEngine
```

Copy this to the appropriate virtual host definitions, e.g.:

```
<VirtualHost *:80>
  ServerAdmin webmaster@ivy.soreco.wan
  DocumentRoot "C:/Program Files (x86)/Apache Software Foundation/Apache2.2/docs/ivy.soreco.wan"
  ServerName ivy.soreco.wan
  ServerAlias www.ivy.soreco.wan
  ErrorLog "logs/ivy.soreco.wan-error.log"
  CustomLog "logs/ivy.soreco.wan-access.log" common
  JkMount /ivy/* AxonIvyEngine
</VirtualHost>
```

5. Define the hostname and port, where the Axon.ivy Engine is running. Adjust the content of the file `C:\Program Files\ivy\worker.properties` to do so.

```
worker.list=AxonIvyEngine
worker.AxonIvyEngine.type=ajp13
worker.AxonIvyEngine.port=8009
worker.AxonIvyEngine.host=ivyserver
```

6. Update the external base URL as shown in the “ivy.webserver.yaml”
7. Restart the Apache HTTP Server and the Axon.ivy overview page should be accessible under `http://apacheHostName/ivy`

Linux example configuration

Within this example an Apache HTTP Server is configured so that it can connect to the Tomcat of an Axon.ivy Engine. The configuration step descriptions are generic and can be used under any Linux distribution. But the concrete examples assume that an Ubuntu distribution is installed as Operating System.

1. Install the latest Tomcat Connector (mod_JK) by console.

```
sudo apt install apache2 libapache2-mod-jk
```

2. Enable the new module

```
sudo a2enmod jk
```

3. Update the `worker.properties` file according to the examples in the `[[Axon.ivy Engine install path]]/misc/apache/`. The following example content would connect to an Axon.ivy Engine on the host “ivyserver” under the default AJP port 8009.

/etc/libapache2-mod-jk/worker.properties:

```
worker.list=AxonIvyEngine
worker.AxonIvyEngine.type=ajp13
worker.AxonIvyEngine.port=8009
worker.AxonIvyEngine.host=ivyserver
```

4. Mount the Axon.ivy Engine in the virtual host definition of the Apache HTTP Server. The context URI must match the context of the Axon.ivy Engine.

/etc/apache2/sites-available/default:

```
<VirtualHost *:80>
  ...
  #Mounts the URI /ivy/* to the worker AxonIvyEngine
  JkMount /ivy/* AxonIvyEngine
</VirtualHost>
```



Tip

If the Apache HTTP Server is used as Load Balancer for a clustered Axon.ivy Engine installation, the JK Status Manager can be used to display debugging informations. The Manager is accessible when it is mounted in the virtual host definition configuration.

```
<VirtualHost *:80>
...
#Mounts the URI /jkmanager/* to the JK Status Manager interface.
JkMount /jkmanager/* jkstatus
</VirtualHost>
```

5. Update the external base URL as shown in the “ivy.webserver.yaml”
6. Restart the Apache HTTP Server and the Axon.ivy overview page should be accessible under *http://apacheHostName/ivy*

Change context URI /ivy/

You might like to make the Axon.ivy engine accessible under a custom context URI other than /ivy.

1. Change the context name of Axon.ivy as shown in the “ivy.webserver.yaml”

```
# sample ivy.yaml that configures a different context:
# so Axon.ivy will be accessible trough http://localhost/workflow
WebServer.IvyContextName: workflow
```

2. Change the context name of the Apache HTTP Server by changing the last line of the *mod_jk.conf* configuration file:

```
#JkMount /ivy/* AxonIvyEngine
JkMount /workflow/* AxonIvyEngine
```

3. If you have a virtual host configuration, the **JkMount** command with the new context URI must also be applied to the virtual host definition:

```
<VirtualHost *:80>
...
JkMount /workflow/* AxonIvyEngine
</VirtualHost>
```

Microsoft IIS Integration



Important

To successfully integrate Axon.ivy Engine into Microsoft Internet Information Server (IIS) it is important that you exactly execute all the integration steps described below. If the integration does not work verify each integration step again.

IIS 8 (Windows Server 2012)

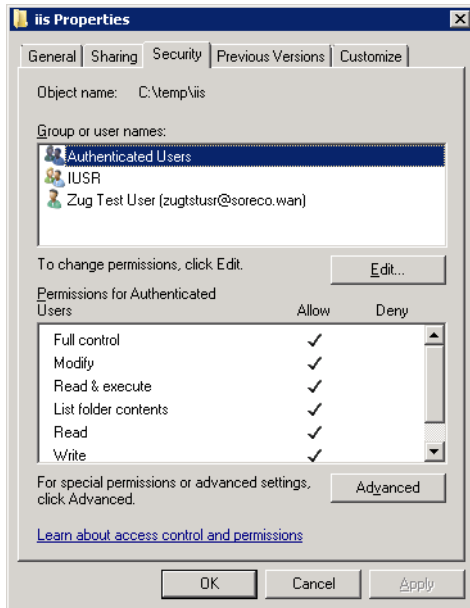


Note

There is a batch script *autoconfig.bat* in the folder *misc\iis* of your engine installation, which installs and configures the IIS automatically on a Windows 2012 Server.

If you are setting up a new IIS Server you can use this script instead of following the instructions below.

1. If your Microsoft Internet Information Server is not running on the same host as the Axon.ivy Engine then copy the integration directory to the host where your IIS is running. All next steps have to be done on the host the IIS is running on.
2. Allow the user groups `Authenticated Users` and `IUSR` to have `Full control` permission on the integration directory.



3. Install Features

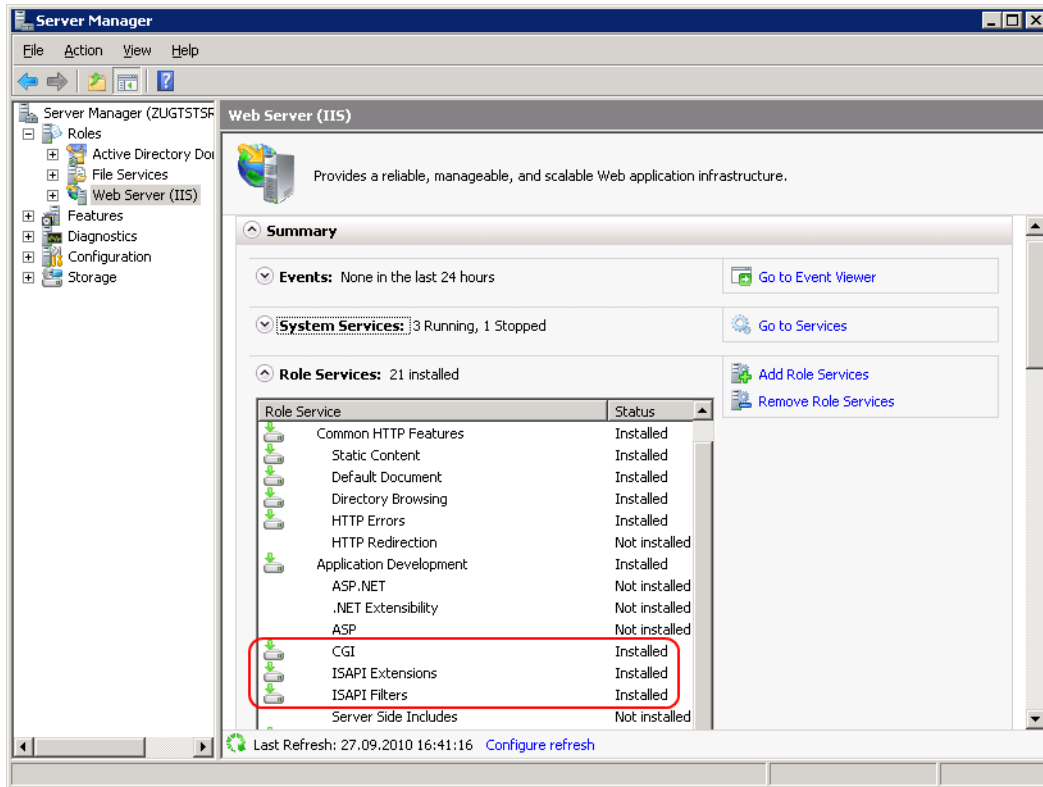


Note

Instead of installing the features manually you can run the following command which ensures that all necessary IIS Features are installed:

```
PKGMGR.EXE /iu:IIS-WebServerRole;IIS-WebServer;IIS-CommonHttpFeatures;IIS-StaticContent;IIS-DefaultDocument;IIS-DirectoryBrowsing;IIS-HttpErrors;IIS-ApplicationDevelopment;IIS-CGI;IIS-ISAPIExtensions;IIS-ISAPIFilter;IIS-HealthAndDiagnostics;IIS-HttpLogging;IIS-RequestMonitor;IIS-Security;IIS-WindowsAuthentication;IIS-RequestFiltering;IIS-Performance;IIS-HttpCompressionStatic;IIS-WebServerManagementTools;IIS-ManagementScriptingTools;IIS-ManagementService
```

Open the Server Manager (*Start > Server Manager*). Select the *Web Server (IIS)*. Validate that under the *Role Services* the services *CGI*, *ISAPI Extensions* and *ISAPI Filters* are installed. If this is not the case select the menu *Add Role Services* to install the missing services.



4. Feature delegation

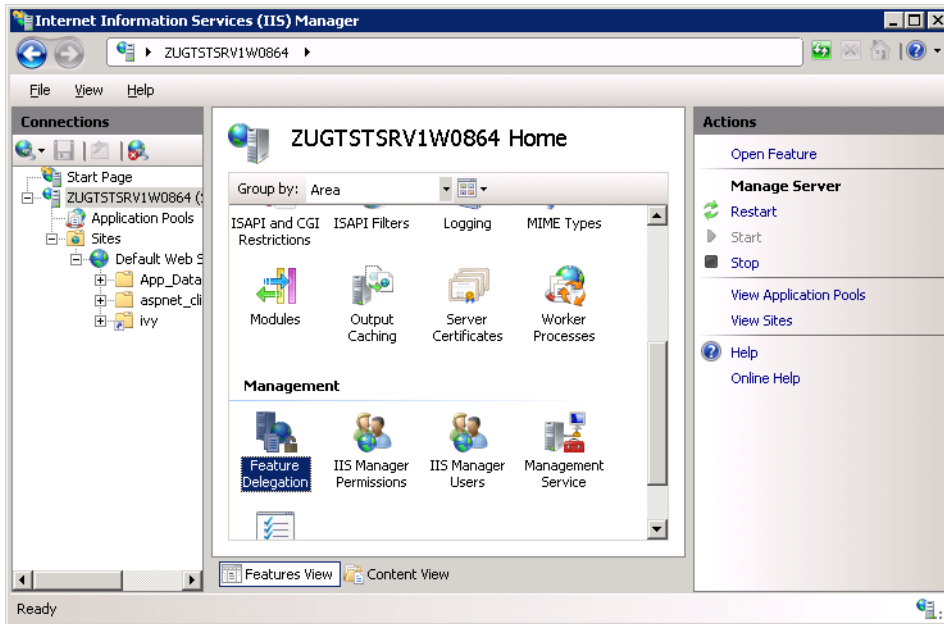


Note

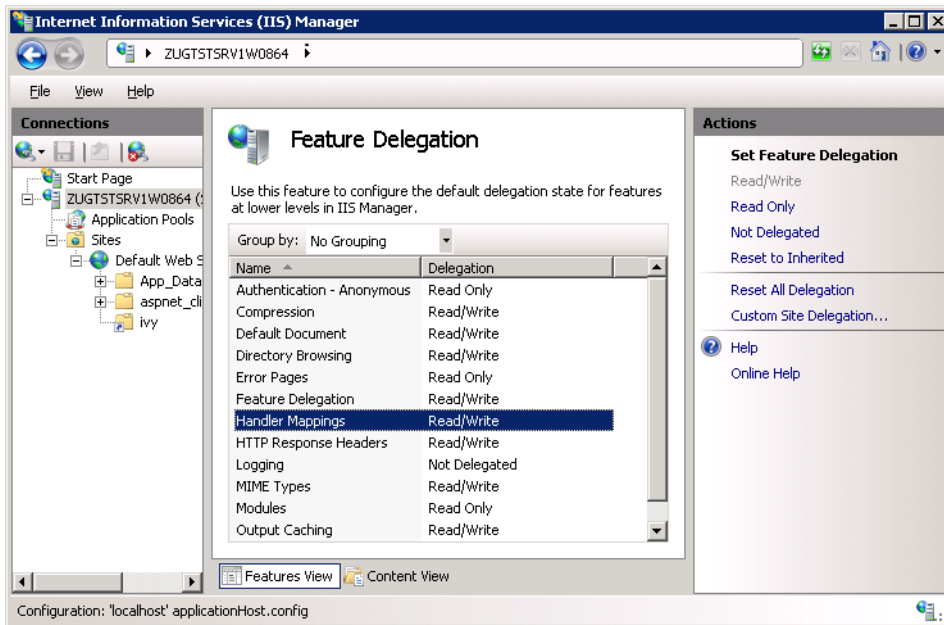
The following command automatically sets the feature delegation:

```
powershell -ExecutionPolicy unrestricted -ImportSystemModules Set-WebConfiguration // System.webServer/handlers -metadata overrideMode -value Allow -PSPath IIS:/
```

Open the Internet Information Services (IIS) Manager (*Start > Internet Information Services (IIS) Manager*). In the Connections pane select the node that represent your machine. In the Feature View open the Feature Delegation entry.



Ensure that the Delegation of the Handler Mappings are set to Read/Write. Use the menu Read/Write on the Actions pane to change the Delegation to Read/Write.



5. Virtual ivy directory



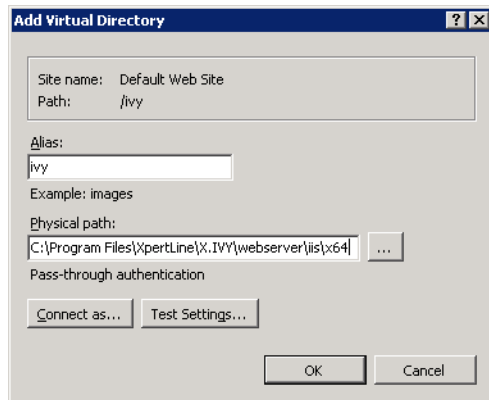
Note

The following commands automatically creates the virtual directory ivy:

```
set path=%path%;%windir%\system32\inetsrv
```

```
appcmd.exe add vdir /app.name:"Default Web Site/" /path:/ivy /physicalPath:<replace this with the path to the integration directory>
```

In the Connections pane navigate to the Web Site you want integrate the Axon.ivy Engine into. Use the context menu `Add Virtual Directory ...` of the Web Site to add a new Virtual Directory. A dialog opens. Configure the `Alias` of the Virtual Directory with `ivy` and the `Physical path` of the Virtual Directory with the path of the integration directory. Click `OK` to close the dialog and create the Virtual Directory:



6. Handler Mapping Permissions

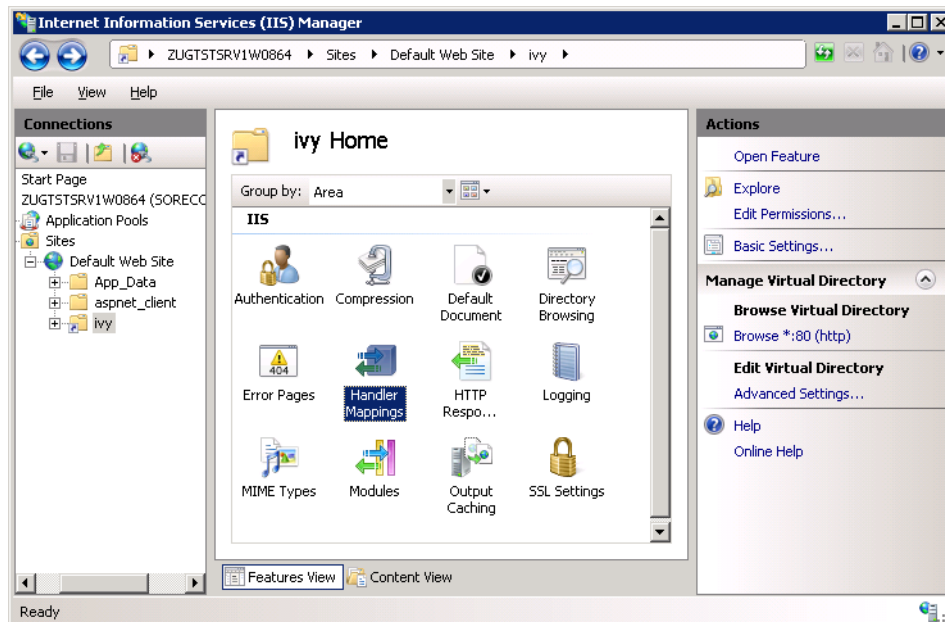


Note

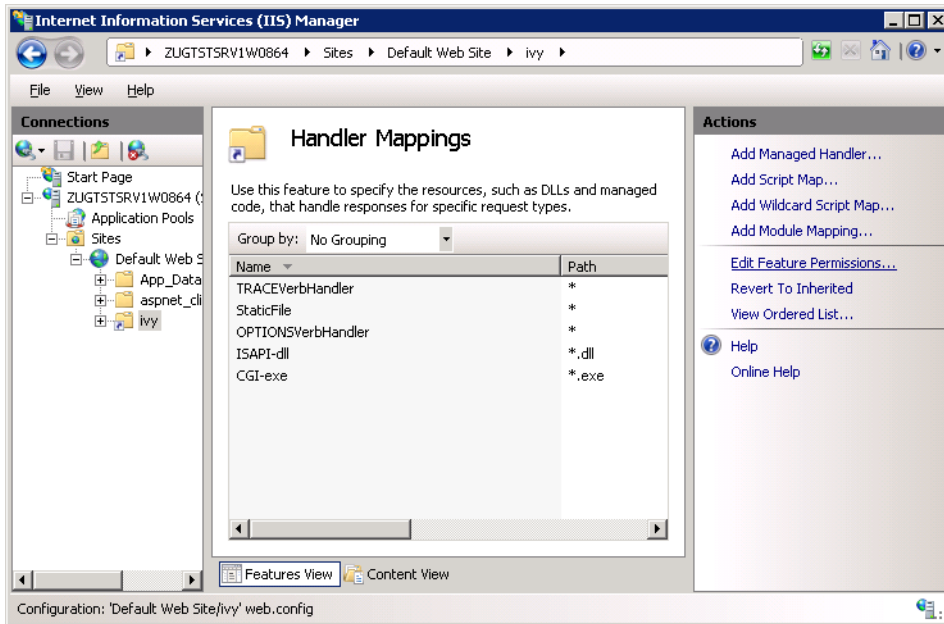
The following command automatically sets the feature permission for the ivy virtual directory:

```
appcmd.exe set config "Default Web Site/ivy" /section:system.webServer/handlers /  
accessPolicy:Read,Write,Execute
```

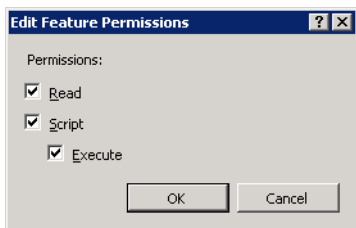
Select the new created Virtual Directory `ivy` in the `Connections` pane and open the `Handler Mappings` entry in the `Feature View`:



In the `Actions` pane select the `Edit Feature Permissions ...` menu:



On the Edit Feature Permission dialog select all three permission and click OK:



7. Configure Error Page



Note

The following command automatically configures that the detailed error page of the Engine is shown:

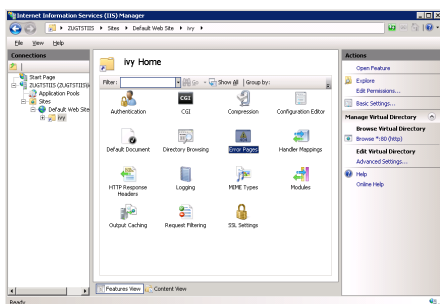
```
appcmd.exe set config "Default Web Site/ivy" /section:system.webServer/httpErrors /errorMode:Detailed
```



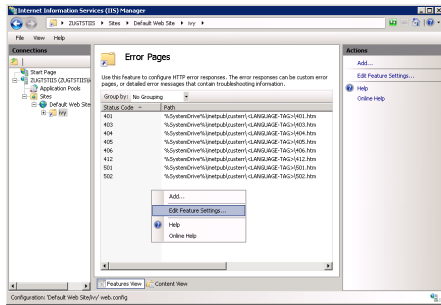
Tip

See the “web.xml” for more information about this configuration.

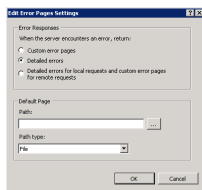
Select the new created Virtual Directory `ivy` in the Connections pane and open the `Error Pages` entry in the Feature View:



Right click and select **Edit Feature Settings...** or select the same from the **Actions** pane (in the right hand side)



Select the **Detailed errors** radio button and click on **OK**



8. Install ISAPI filter

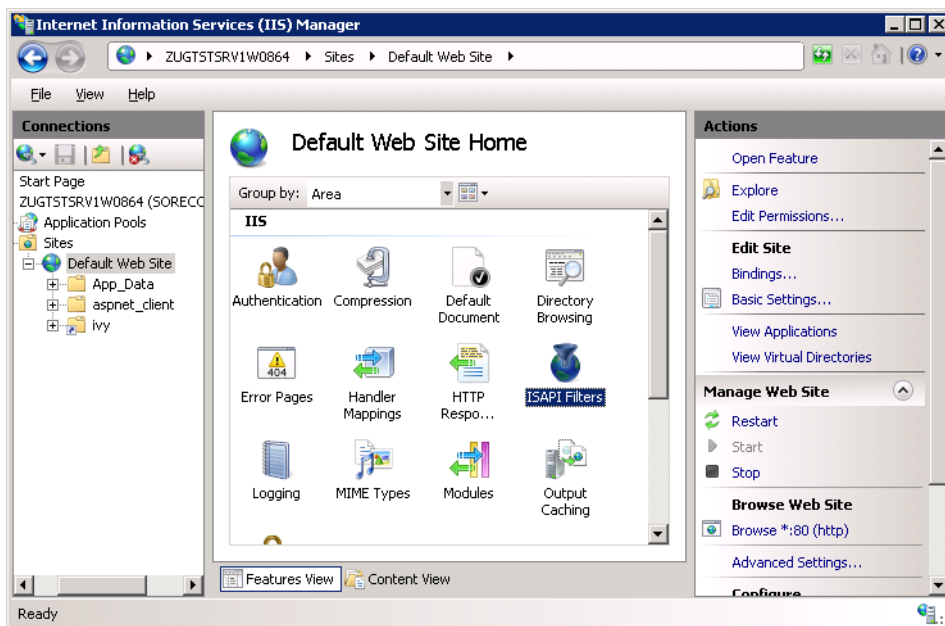


Note

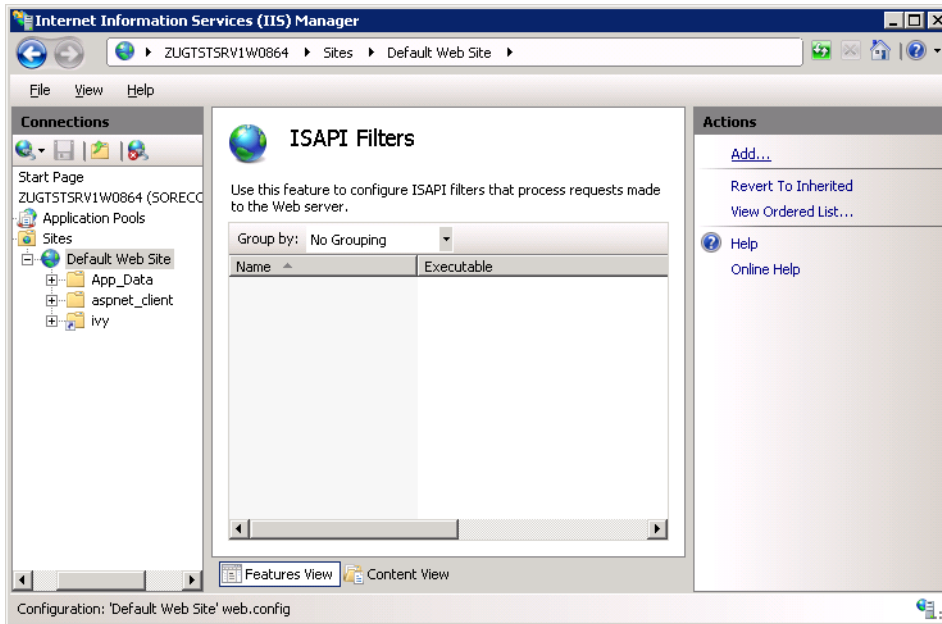
The following command automatically adds the ISAPI Filter:

```
appcmd.exe set config /section:isapiFilters /+[@start,name='Tomcat',path='<replace this with the path to the integration directory>\isapi_redirect-1.2.42.dll']
```

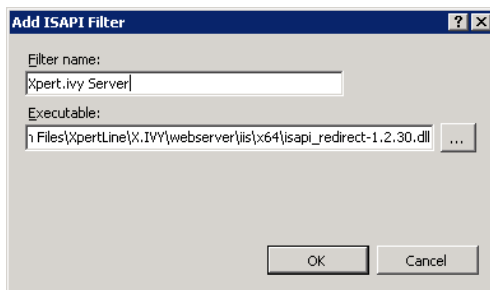
Select the **Web Site** in the **Connections** pane and open the **ISAPI Filters** entry in the **Feature View**:



In the **Actions** pane select the **Add...** menu:



On the Add ISAPI Filter dialog configure the Filter name with Axon.ivy Engine and the Executable with the path of the *isapi_redirect-1.2.42.dll* located in the integration directory. Click OK to add the ISAPI Filter:



9. Change ISAPI filter restriction

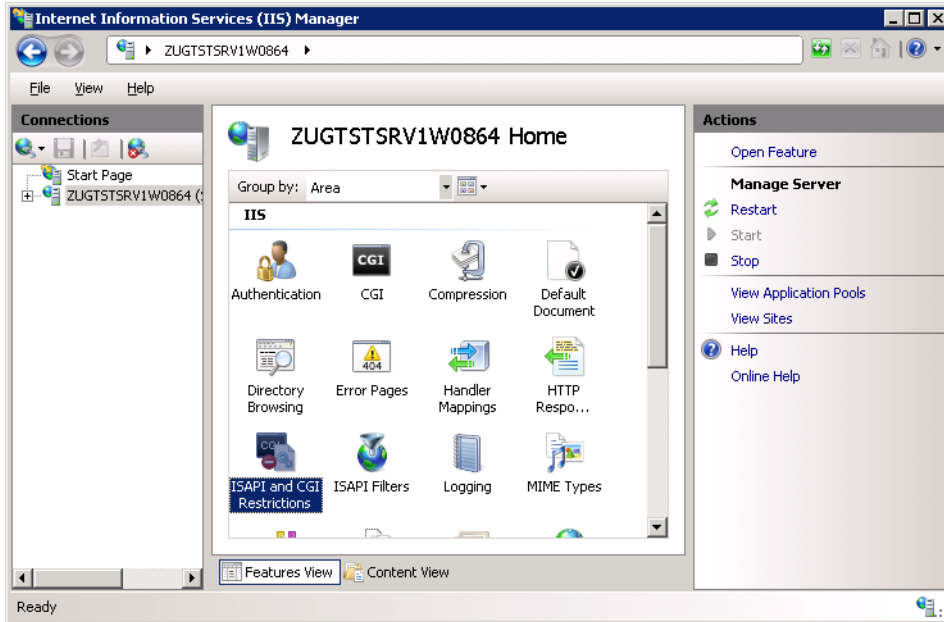


Note

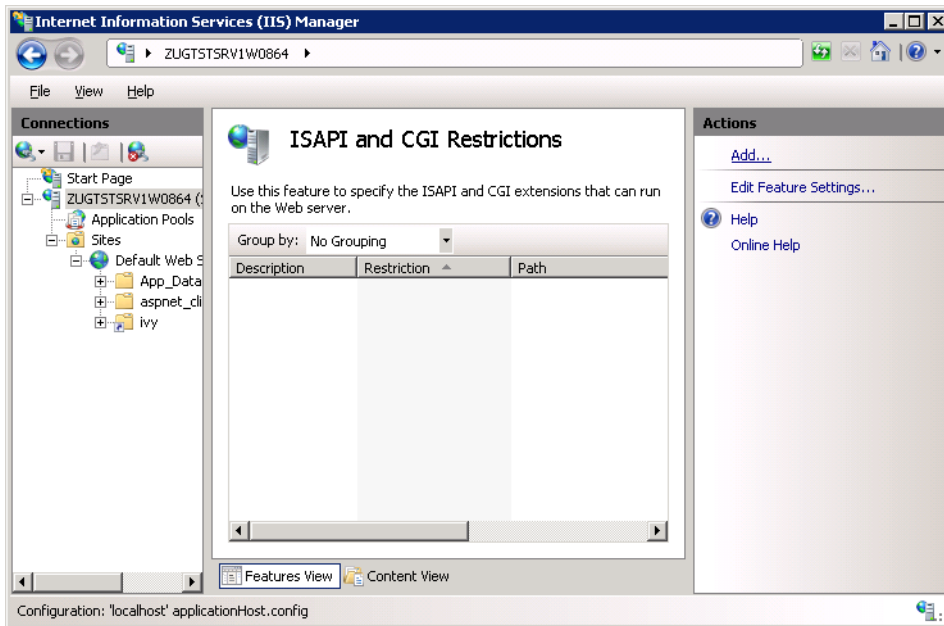
The following command automatically adds the ISAPI Restriction:

```
appcmd.exe set config /section:isapiCgiRestriction /+[@start,description='Tomcat',path='<replace this with the path to the integration directory>\isapi_redirect-1.2.42.dll',allowed='true']
```

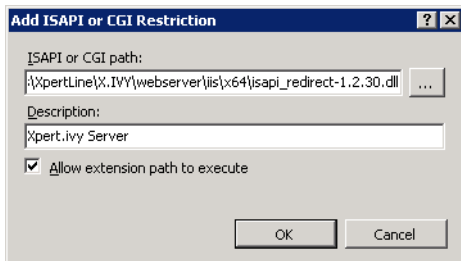
In the Connections pane select the node that represent your machine and open the ISAPI and CGI Restrictions entry in the Features View:



In the Actions pane select the Add . . . menu:



On the Add ISAPI or CGI Restriction dialog configure the ISAPI or CGI path with the path of the *isapi_redirect-1.2.42.dll* located in the integration directory. As Description use *Axon.ivy Engine*. Select the Allow extension path to execute check box. Click OK to add the ISAPI or CGI Restriction:



10. If your Microsoft Internet Information Server is not running on the same host as the Axon.ivy Engine or if you have changed the AJP port of the Axon.ivy Engine then open the file *worker.properties* inside the integration directory in a text editor. Change the following line if you have changed the AJP port to another value than 8009:

```
worker.AxonIvyEngine.port=8009
```

Change the value `localhost` in the following line to the host where your Axon.ivy Engine is running if your Microsoft Internet Information Server is not running on the same host as the Axon.ivy Engine:

```
worker.AxonIvyEngine.host=localhost
```

11. Update the external base URL as shown in the “ivy.webserver.yaml”
12. Check if the integration is working by opening a web browser on the address `http://<your host>/ivy/`

Change context URI /ivy/

You might like to make the Axon.ivy engine accessible under a custom context URI other than /ivy/.

1. Change the context name of Axon.ivy as shown in the “ivy.webserver.yaml”

```
# sample ivy.yaml that configures a different context:
# so Axon.ivy will be accessible trough http://localhost/workflow
WebServer.IvyContextName: workflow
```

2. Change the context name of the Microsoft IIS by changing the last line of the *uriworkemap.properties* configuration file:

```
#/ivy/* AxonIvyEngine
/workflow/*=AxonIvyEngine
```

Access multiple Axon.ivy Engines through one IIS

Multiple Axon.ivy Engine instances can be accessed through a single IIS server. This is especially useful if multiple Axon.ivy versions must be accessible during a migration phase. The following explanation shows a solution for the scenario, where a legacy Xpert.ivy 3.9 Server and an Axon.ivy 5.x Engine must be accessible through a single IIS host.

1. Make the newer Axon.ivy Engine accessible through the IIS as if only one engine would be behind the IIS. For detailed instructions follow “Microsoft IIS Integration”.

In our scenario the integration directory from the Axon.ivy 5.x Engine was used to make the engine instance accessible under `http://localhost/ivy`.

2. The contexts of the Axon.ivy Engines must be unique. By default the context is set to `/ivy/`. If different versions of ivy engines are accessed from the same IIS host, it's useful to change the contexts so that it matches the ivy version. For detailed explanation see “Change context URI /ivy/”

In our scenario the context URI of the Axon.ivy 5.x Engine was changed to `/ivy5/` and the Xpert.ivy 3.9 Server kept his default context `/ivy/`.

3. All Axon.ivy Engines, which are accessed from the same IIS, must listen on a different port for AJP communication. Therefore the AJP port must be changed. This can be configured as shown in the “ivy.webserver.yaml”.

In our scenario the AJP port of the Axon.ivy 7.x Engine was changed to 8010 and the Xpert.ivy 3.9 Server kept his default AJP port 8009.

```
# ivy.yaml with AJP enabled on 8010
AJP:
  Enabled: true
  Port: 8010
```

4. The Axon.ivy Engines must be declared in the *worker.properties* file of the integration directory. It's important that each worker has a unique name and that they are listed in the *worker.list* property.

In our scenario the *worker.properties* looks as follows:

```
worker.XpertIvyServer3x.type=ajp13
worker.XpertIvyServer3x.port=8009
worker.XpertIvyServer3x.host=ivyhostname39

worker.AxonIvyEngine5x.type=ajp13
worker.AxonIvyEngine5x.port=8010
worker.AxonIvyEngine5x.host=ivyhostname50

worker.list=XpertIvyServer3x,AxonIvyEngine5x
```

5. The contexts of the Axon.ivy Engines must be registered in the *uriworkemap.properties* file of the integration directory.

In our scenario we make Axon.ivy 5.x available under `http://localhost/ivy5/` and Xpert.ivy 3.9 under `http://localhost/ivy`. So the *uriworkemap.properties* file looks as follows:

```
/ivy/*=XpertIvyServer3x
/ivy5/*=AxonIvyEngine5x
```

Single Sign On

Axon.ivy Engine supports single sign on in Windows environments. The following preconditions must be fulfilled for single sign on:

- The application on the Axon.ivy Engine must use Active Directory Security System
- The Axon.ivy Engine must be integrated into a Microsoft Internet Information Server (IIS)

IIS 8 (Windows Server 2012)



Note

There is a batch script *autoconfigSSO.bat* in the folder *misc\iis* of your engine installation. This script automatically sets up SSO on a Windows 2012 Server.

If you are setting up a new IIS Server you can use this script instead of following the instructions below.

1. **Install Windows Authentication**

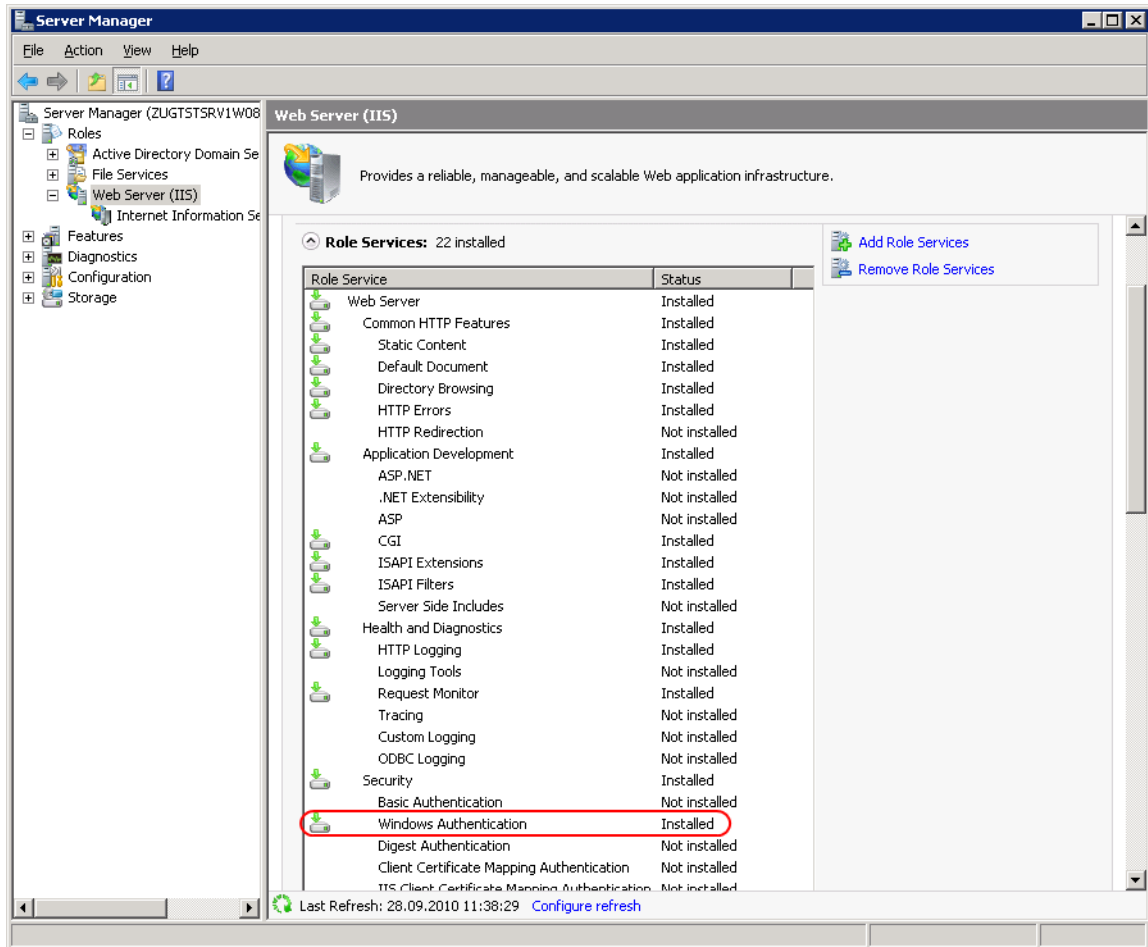


Note

The following command automatically installs the Windows Authentication:

```
PKGMRG.EXE /iu:IIS-WindowsAuthentication
```

Open the Server Manager (*Start > Server Manager*). Select the *Web Server (IIS)*. Validate that under the *Role Services* the service *Windows Authentication* is installed. If this is not the case select the menu *Add Role Services* to install the missing service.



2. Deactivate Anonymous Authentication



Note

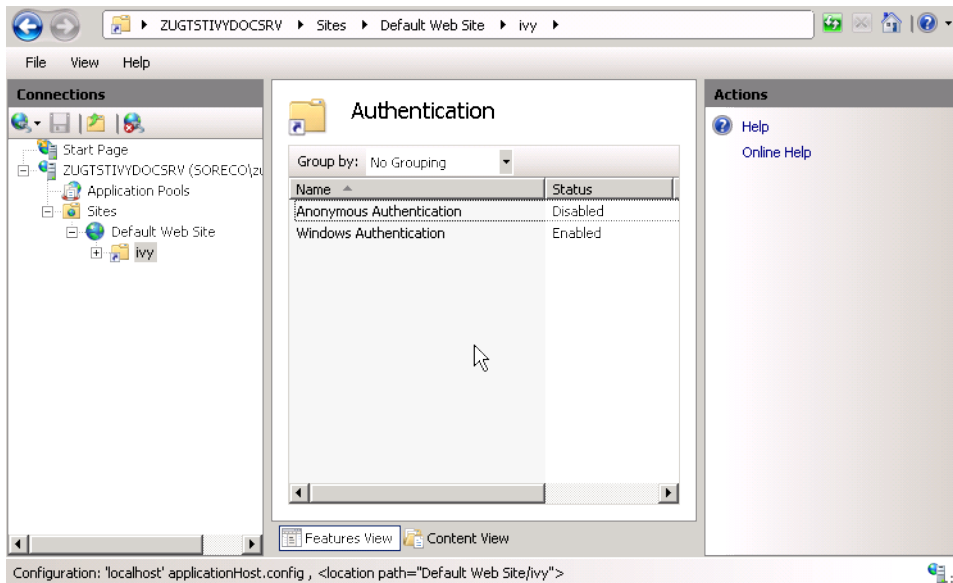
The following command automatically deactivates the Anonymous Authentication:

```
set path=%path%;%windir%\system32\inetsrv
```

```
appcmd.exe set config "Default Web Site/ivy" -section:system.webServer/security/authentication/anonymousAuthentication/enabled:"False" /commit:apphost
```

Open the Internet Information Services (IIS) Manager (*Start > Internet Information Services (IIS) Manager*). In the Connections pane select the *ivy* Virtual Directory node. In the Feature View open the Authentication entry. Select the Windows Authentication and use the menu Enable in the Actions pane to enable Windows Authentication.

Make sure that all other authentication modes such as Anonymous Authentication or Digest Authentication are disabled, otherwise IIS will use those authentication modes and Single Sign On will not work.



3. Activate Windows Authentication



Note

The following command automatically activates the Windows Authentication:

```
appcmd.exe set config "Default Web Site/ivy" -section:system.webServer/security/authentication/windowsAuthentication /enabled:"True" /-providers.[value='Negotiate'] /commit:apphost
```

Remove all providers except NTLM from Windows Authentication, otherwise Single Sign On may not work with the RIA clients.

Troubleshooting

<https://answers.axonivy.com/tags/ajp/>

Basic Authentication

In the following situations Basic Authentication is required:

- to use the Axon.ivy Mobile App
- to provide REST services which require authentication

IIS 8 (Windows Server 2012)



Note

There is a batch script *autoconfigBasicAuth.bat* in the folder *misc\iis* of your engine installation. This script automatically sets up Basic Authentication on a Windows 2012 Server.

If you are setting up a new IIS Server you can use this script instead of following the instructions below.

1. Install Basic Authentication

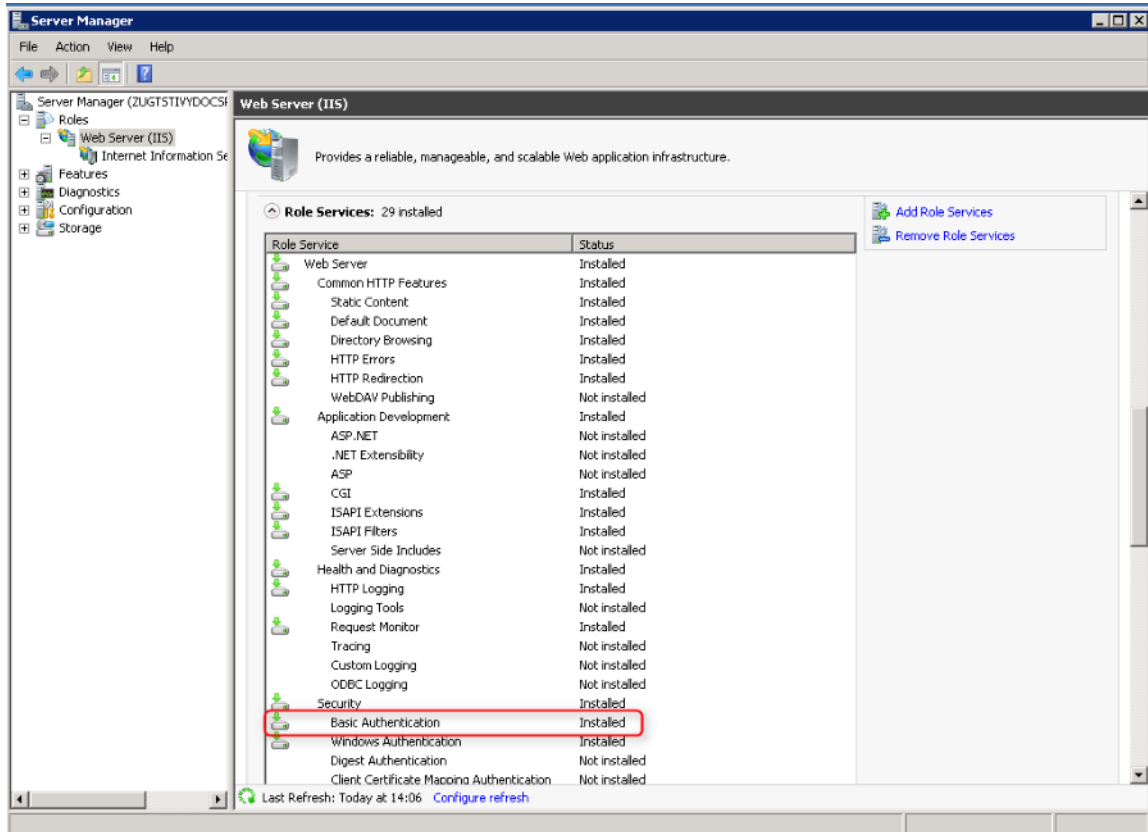


Note

The following command automatically installs Basic Authentication:

PKGMR.EXE /iu:IIS-BasicAuthentication

Open the Server Manager (*Start > Server Manager*). Select the *Web Server (IIS)*. Validate that under the Role Services the service Basic Authentication is installed. If this is not the case select the menu Add Role Services to install the missing service.



2. Activate Basic Authentication



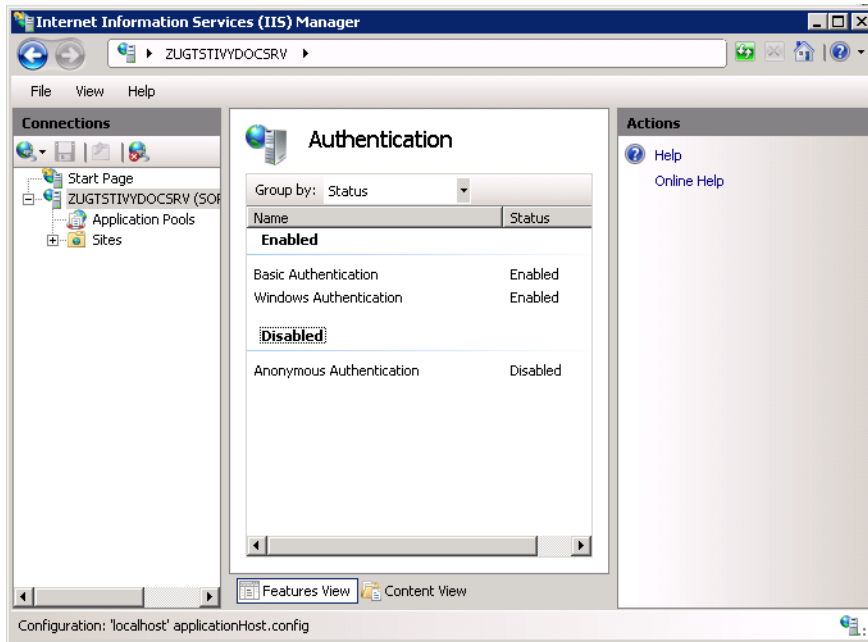
Note

The following command automatically activates the Basic Authentication:

```
set path=%path%;%windir%\system32\inetsrv
```

```
appcmd.exe set config "Default Web Site/ivy" -section:system.webServer/security/authentication/basicAuthentication /enabled:true /commit:apphost
```

Open the Internet Information Services (IIS) Manager (*Start > Internet Information Services (IIS) Manager*). In the Connections pane select the ivy Virtual Directory node. In the Feature View open the Authentication entry. Select the Basic Authentication and use the menu Enable in the Actions pane to enable Basic Authentication.



Error Handling

If the engine is running behind an IIS web server and an error occurs on the Engine IIS shows its own error page and hides the error page coming from the Engine. This is the default IIS behavior.

The Axon.ivy IIS integration script configures the IIS to show the detailed error page of the Engine (see 'Errors' in "ivy.yaml"). IIS can be reset to its default behavior (e.g. because of security reasons) with the following steps:

1. Open the IIS manager
2. Select the virtual directory `ivy` and on its `Features View`, double click on `Error Pages`
3. Right click and select the `Edit Feature Settings...` or select the same from the `Actions` pane (on the right hand side)
4. Select the "Detailed errors for local requests ..." radio button and click OK.

Axon.ivy Cluster Integration

Axon.ivy Engine Enterprise Edition (Cluster) works with sticky sessions. This means that the load balancer must forward all requests from a session to the same cluster node. Of course if a cluster node is no longer available then the request can be sent to another cluster node. Note, that this will cause that the user gets a new session and he loses his current work.

Load Balancing with Tomcat connector (IIS, Apache)

The Tomcat connector can be configured to act as a load balancer for multiple Axon.ivy Engine Enterprise Edition nodes. The load balancer and the cluster nodes can be configured in the `workers.properties` file that is located in the integration directory. An example load balancer configuration can be found in the file `cluster_loadbancer_workers.properties`. In this file one worker is configured called `AxonIvyEngine` that is a load balance worker (`type=lb`). The property `balance_workers` of the `AxonIvyEngine` worker defines the workers between which the load balance worker will balance the load. Here one worker per each Axon.ivy Engine Node should be configured. In the example file three workers are configured `AxonIvyEngineNode1`, `AxonIvyEngineNode2` and `AxonIvyEngineNode3`.

The node workers are similar to a normal standalone worker. You can use the attributes `hostname` and `port` as explained above. Additionally they have two extra attributes called `lbfactor` and `route`. With the `lbfactor` attribute you can influence how the load balancer distributes the load to the workers. The higher the `lbfactor` of a worker relative to the other workers is the more load the worker gets.

The `route` attribute is necessary for realizing sticky sessions. An Axon.ivy Engine Enterprise Edition will only work correctly, if the load balancer sends all request of the same http session to the same node (sticky sessions). To support this requirement, each Axon.ivy Engine Enterprise Edition node will add a special identifier called `jvm route` to the http session identifier. The `jvm route` identifier is calculated from the host name and the Local Cluster Node Identifier. The `route` attribute configured on a node worker must be equal with the `jvm route` of the node:

```
worker.AxonIvyEngineNode1.route=<JVM route identifier of Node 1>
```

```
worker.AxonIvyEngineNode2.route=<JVM route identifier of Node 2>
```

The *JVM route identifier* of a cluster node can be found on the *cluster node detail page* for an Axon.ivy Cluster Node. This information can be retrieved as follows:

1. Using a web browser, navigate to the main page (`http://<host>:<port>/ivy`) of an Axon.ivy Engine installation.
2. Select the *Cluster* link in the page header.
3. In the appearing list of cluster nodes press the name of a cluster node to see it's details.

The screenshot shows the 'Cluster Nodes' page in the Axon.ivy management interface. A table lists two nodes. The first node, 'ZUGPCFS.axonivy.soreco.ch', is highlighted with a red box and a red arrow labeled 'click'. Below the table, the 'Cluster Node Detail single node' view is shown, with the 'JVM Route' field highlighted by a red box and a red arrow.

Name	Host	State	Communication	Is Master	Is Local	Start Time
ZUGPCFS.axonivy.soreco.ch	ZUGPCFS.axonivy.soreco.ch	RUNNING	UP	yes	yes	05.09.14 11:20
ZUGPCFS.axonivy.soreco.ch	ZUGPCFS.axonivy.soreco.ch	RUNNING	UP	no	no	05.09.14 11:20

Cluster Node Detail single node

Name	single node
Host	localhost
IP Address	127.0.0.1
IP Port	-1
Local Identifier	0
JVM Route	zugpcfs_axonivy_soreco_ch
State	RUNNING
Communication	UP
Communication Address	-
Is Master	yes
Master Cluster Node	ZUGPCFS.axonivy.soreco.ch
Is Local	yes
Last Start Timestamp	05.09.14 11:20
Last Stop Timestamp	-
Last Fail Timestamp	-
Axon.Ivy Version	RC-5.1.0
Operating System Name	Linux
Operating System Version	3.11.0-26-generic
Operating System Architecture	amd64
Java Version	1.7.0_55
Java Virtual Machine Name	Java HotSpot(TM) 64-Bit Server VM

Figure 6.2. Axon.ivy Cluster Node Details page

More technical details about load balancing and sticky sessions can be found on the Apache Tomcat web site.

Example

Let's assume that we have an Axon.ivy Engine Enterprise Edition with two Cluster Nodes. Node 1 is installed on host `ivynode1` and the AJP port is configured to 8009. Node 2 is installed on host `ivynode2` and the AJP port is configured to 8010. `ivynode1` is a new machine with a lot of power. `ivynode2` is an old machine and we want that `ivynode1` is working twice as hard as `ivynode2`. The `jvm route` of the nodes are `ivynode1.soreco.ch` and `ivynode2.soreco.ch`.

The `workers.properties` file must then look like this:

```
worker.list=XIvy

# Load Balanced Cluster Worker
worker.AxonIvyEngine.type=lb
worker.AxonIvyEngine.balance_workers=AxonIvyEngineNode1,AxonIvyEngineNode2

# 1st Axon.ivy Engine Cluster Node
worker.AxonIvyEngineNode1.type=ajp13
worker.AxonIvyEngineNode1.port=8009
worker.AxonIvyEngineNode1.host=ivynode1
worker.AxonIvyEngineNode1.route=ivynode1.soreco.ch
worker.AxonIvyEngineNode1.lbfactor=2
```

```
# 2nd Axon.ivy Engine Cluster Node
worker.AxonIvyEngineNode2.type=ajp13
worker.AxonIvyEngineNode2.port=8010
worker.AxonIvyEngineNode2.host=ivynode2
worker.AxonIvyEngineNode2.route=ivynode2.soreco.ch
worker.AxonIvyEngineNode2.lbfactor=1
```

Load Balancing with other Load Balancer Products

As described above the load balancer must ensure that all requests from the same user session is forwarded to the same cluster node. This can be done by configuring the load balancer so that all requests sent by one client IP address is always forwarded to the same cluster node (IP based stickiness). Another possible configuration is to use the Axon.ivy Session Id to provide session stickiness. The session id is provided by Axon.ivy Engine Enterprise Edition with two different methods. Either as HTTP session cookie with the name JSESSIONID or at the end of request URLs as ;jsessionid= parameter. The load balancer must be able to read the session id with both methods otherwise RIA clients will not work correctly.



Warning

Some load balancer provide session stickiness using their own HTTP session cookie. If you use this method then RIA clients will fail to start.

Web Application Firewall

A web application firewall (WAF) or web shield is a firewall which protects web applications against attacks over the HTTP protocol. Combined with an Identity and Access Management (IAM) System it also protects against unauthorized access and supports single sign on (SSO).

Single Sign On

Most WAF or IAM systems allow to configure a way how the user name of the identified user is transmitted to the web applications. With Axon.ivy Engine a typical system landscape will look like this:

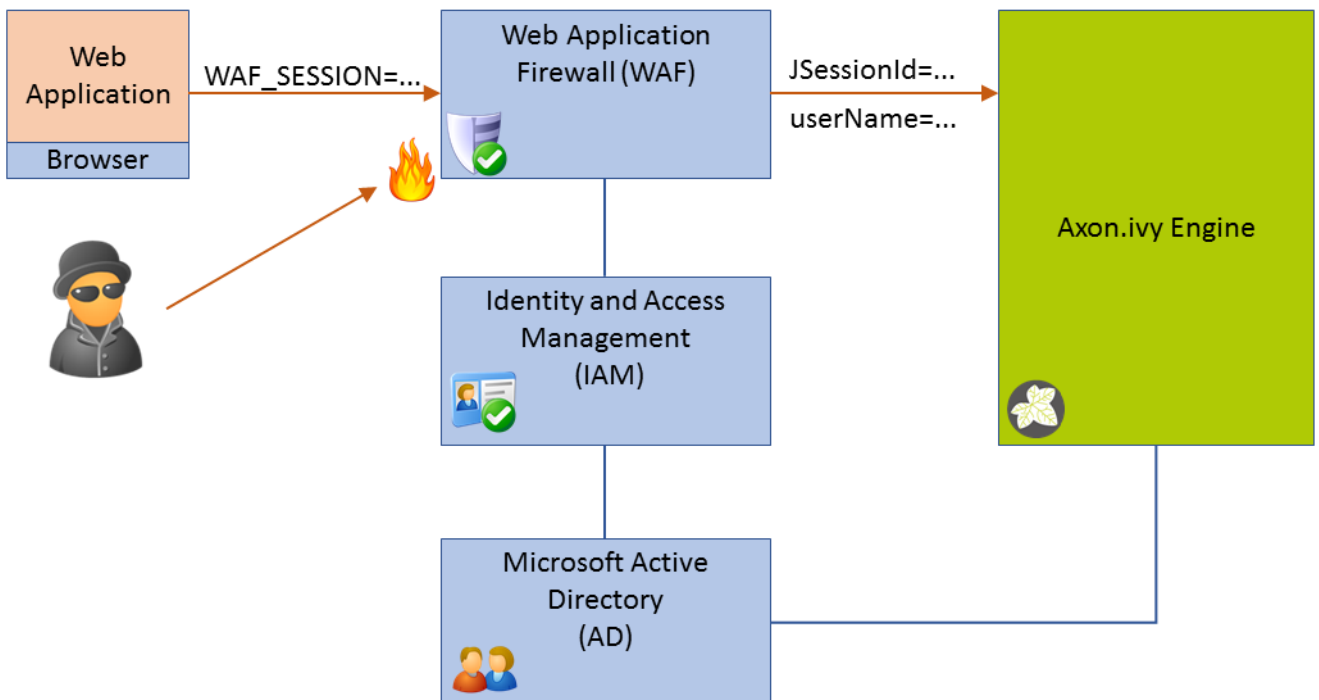


Figure 6.3. Single Sign On Infrastructure using a Web Application Firewall, Identity and Access Management and Active Directory

The only available access point must be the WAF. Any traffic has to be routed over it. The WAF tries to protect the web application behind it (e.g. Axon.ivy Engine) from attacks. The WAF uses the IAM to identify users and to protect certain resources from unauthorized access. The IAM itself may use a directory server like Microsoft Active Directory to know users. The WAF can be configured to provide the name of the identified user either as HTTP header or HTTP cookie to the web application (Axon.ivy Engine).

On the other side Axon.ivy Engine provides a Valve that reads the user name from a HTTP header. If Axon.ivy Engine knows the user it automatically authenticates the user to the current Axon.ivy Engine session. This works best if Axon.ivy Engine also uses a directory server like Microsoft Active Directory to synchronize users. The Valve that reads the user name from a HTTP header is disabled by default. To enable it, open the file “context.xml” in the `[engineDir]/webapps/ivy/META-INF` directory and uncomment the following line:

```
<Valve className="ch.ivyteam.ivy.webserver.security.SingleSignOnValve" userNameHeader="user"
```

The attribute `userNameHeader` can be used to configure the HTTP header that should be read.



Warning

If you activate this Valve you must ensure that the Axon.ivy Engine cannot be accessed directly. All traffic must be routed over the WAF. Otherwise, an attacker could simply send a valid user name as header in a HTTP request and immediately has access bypassing the authentication!

Instead of sending the plain user name in a HTTP header there are multiple other ways and technologies (SAML token, Kerberos, etc.) how the WAF can transmit the current user identity to the web applications. You can support this cases by registering your own Valve in the “context.xml” file. Your value reads the current user identity from the request and puts a user principal object with the user name to it. Axon.ivy Engine will check if a user principal is set on a request and automatically searches the user and authenticates it. The code of your valve can look like this:

```
import java.io.IOException;
import java.security.Principal;

import javax.servlet.ServletException;

import org.apache.catalina.connector.Request;
import org.apache.catalina.connector.Response;
import org.apache.catalina.valves.ValveBase;
import org.apache.commons.lang3.StringUtils;

public class AuthValve extends ValveBase
{
    @Override
    public void invoke(Request request, Response response) throws IOException, ServletException
    {
        String userName = getUserFromRequest(request);
        if (StringUtils.isNotBlank(userName))
        {
            Principal userPrincipal = createUserPrincipalWith(userName);
            request.setUserPrincipal(userPrincipal);
        }
        getNext().invoke(request, response);
    }

    /**
     * Finds out which user was authenticated by an external instance
     * @param request
     * @return user name
     */
    private String getUserFromRequest(Request request)
    {

```

```
// Example implementation:
// Gets the user name from the HTTP Header field User.
// Has to be changed depending on the protocol or product that you are using
String userName = request.getHeader("User");
return userName;
}

private Principal createUserPrincipalWith(String userName)
{
    return new UserPrincipal(userName);
}

private static class UserPrincipal implements Principal
{
    private String userName;

    private UserPrincipal(String userName)
    {
        this.userName = userName;
    }

    @Override
    public String getName()
    {
        return userName;
    }
}
}
```

The method `getUserNameFromRequest` depends on the technology the WAF sends the user identity.

Chapter 7. Administration

Deployment

Bring your processes to life by deploying them on an Axon.ivy engine. Deployment simply means to install an Axon.ivy project on an Axon.ivy engine. Our file based deployment mechanism makes the deployment very easy, just by dropping the file at the right place. This mechanism is perfectly suitable for a CI/CD pipeline and forms the basis for the deployment feature of our maven plugin.

1. Get a prepared ivy project from your developer.
2. Deploy the project by simply dropping the file in the deployment directory.
3. Check the result of the deployment on the server info page.

Prepare

Before deployment, the Axon.ivy project must be available as ivy-archive (IAR) or packed as a zip-archive (ZIP). It is also possible to pack multiple Axon.ivy projects in one single zip-archive. All project dependencies must be resolved, either already installed in the application or part of the deployment.

We recommend to build a zip-archive, which contains all projects of an application.

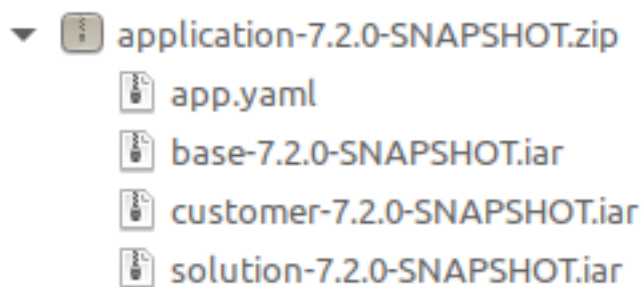


Figure 7.1. Sample full application zip-archive

You are able to configure the application within a full application zip-archive. Also, proper versioning is important during deployment.

Deploying

Drop the file in the deployment directory, the deployment will be started immediately if the Axon.ivy Engine is running. Otherwise, the deployment is executed when the engine is starting.

There are subdirectories in the deployment directory for each application. The project can be copied to the corresponding subdirectory. It is also possible to create a subdirectory manually. In this case a new application will be created. Alternatively, the project can also be placed in the deployment directory itself. It is then deployed into the application with the same name as the filename.

If you want to influence the deployment behavior, you can do this with deployment options.

Check the results

Go to the server info page, which is by default <http://localhost:8080/ivy>. You should see now your new processes available. If you think there is missing something you better check the deployment logs, which can be found in the same directory where you have dropped the deployment file.

File suffix	Description
.deployed	Last deployed file to the engine in case of a successful deployment
.notDeployed	Last not deployed file to the engine in case of error while deployment
.deploymentLog	Contains the log output which is written during the deployment
.deploymentError	Contains the error cause and is only written when the deployment fails

Table 7.1. Deployment marker files

Advanced Deployment

To fully automate your CI/CD pipeline, you may want to further configure your target application of your deployment and also influence the deployment behavior.

Configure Application

If you are deploying a full application zip-archive you can optionally add an “app.yaml” in the root of your zip-archive, which contains the configuration of the application.

```
# sample app.yaml which can be part of the full application zip-archive
SecuritySystem: ActiveDirectoryOfMyCompany
EMailNotification:
  DailySummaryOn: monday, tuesday, wednesday, thursday, friday
  OnNewTasks: true
StandardProcess:
  DefaultPages: ch.ivyteam.ivy.project.portal:portalTemplate
  MailNotification: ch.ivyteam.ivy.project.portal:portalTemplates
```

Versioning

It is highly recommended that you increase the version of your project each time you want to deploy a new version of your project on the engine. This ensures that you will not break currently running cases, and you have the possibility to go back to the previous version if the new version does not work as expected.

Even though overwriting an already deployed process model version with running cases is possible. It is at your own risk and should only be done if you are aware of the possible consequences and ready to accept them.

Deployment Options

With deployment options you can influence the deployment behavior:

```
# Example of a deployment options file using YAML format.
# See http://axonivy.github.io/project-build-plugin/release/deploy-to-engine-mojo.html for
# more information about deployment options parameters.
# All parameters are set to their default values and listed in [brackets].

# Flag indicating if project test users should be deployed to the engine.
deployTestUsers: false          # [false], true

# Defines how project deployment should affect engine configuration.
configuration:
  overwrite: false              # [false], true
  cleanup: DISABLED            # [DISABLED], REMOVE_UNUSED, REMOVE_ALL

# Deployment target settings.
target:
```

```
version: AUTO # [AUTO], RELEASED, (version values, e.g. 2.5 or (2.0,3.0
state: ACTIVE_AND_RELEASED # [ACTIVE_AND_RELEASED], ACTIVE, INACTIVE
fileFormat: AUTO # [AUTO], PACKED, EXPANDED
```

See the Axon.ivy Project Build Plugin deploy documentation for more information about the deployment properties.

There are three distinct locations where you can put your options file:

1. Deployment specific options file - If you want to provide options for a single deployment simply create a file that has the same prefix as the file you want to deploy with a suffix of *.options.yaml*. E.g. if the file you want to deploy is *myProject.iar* then create an options file that is called *myProject.iar.options.yaml*. Note, that after the deployment the *myProject.iar.options.yaml* file will be removed automatically.
2. Global options file - Create a file called *deploy.options.yaml* in the deployment directory of an application. This global options file control all deployments inside the application.
3. Inside the file that you are deploying - Put a *deploy.options.yaml* file inside your project directory, *.iar or *.zip that you want to deploy.

If there are multiple options files available, then only the options file with the highest location priority is considered, other options files will be ignored.

Maven Plugin

The Maven project-build-plugin makes automated continuous deployment to an Axon.ivy Engine possible. The Maven plugin itself uses the file based deployment capability of the Axon.ivy Engine. This means that the deployment folder must be available on the same machine on which the build is executed. You may use Windows-Shares or SMB-Configurations.

An Axon.ivy project can be deployed by invoking Maven with the *deploy-to-engine* goal of the project-build-plugin. To customize the deployment parameters, consult the goal documentation.

Command line deployment

The *deploy-to-engine* goal can be execute on the command line. The following example deploys the project *myProject.iar* to the application 'Portal' of the Engine location under *c:/axonivy/engine*:

```
mvn com.axonivy.ivy.ci:project-build-plugin:7.1.0-SNAPSHOT:deploy-to-engine -Divy.deploy.f
```

Build goal execution

To deploy an ivy-archive (IAR) during it's Maven build lifecycle configured an execution which binds the *deploy-to-engine* goal to a phase in the projects *pom.xml*.

The following POM snippet deploys the current project to the application 'Portal' of the Axon.ivy Engine under *c:/axonivy/engine*.

```
<plugin>
  <groupId>com.axonivy.ivy.ci</groupId>
  <artifactId>project-build-plugin</artifactId>
  <extensions>>true</extensions>
  <executions>
    <execution>
      <id>deploy.to.engine</id>
      <goals><goal>deploy-to-engine</goal></goals>
      <phase>deploy</phase>
      <configuration>
        <deployToEngineApplication>Portal</deployToEngineApplication>
```

```
<deployEngineDirectory>c:/axonivy/engine</deployEngineDirectory>
</configuration>
</execution>
</executions>
</plugin>
```

Further examples are documented on GitHub in the project-build-examples repository.

Standard Processes

With standard processes you can change the default workflow behaviour by simply providing a custom implementation in your ivy project.

For example: Once a user has completed a task, he will be redirected to his personal task list. The default task list is workflow driven and maybe to technical for your end user. With a standard process you could easily provide a branded and use case driven task list that fits perfectly into your domain.

We distinguish between two types of standard processes:

- “Default Pages” are simple pages like the login page or the task list page.
- “Email Notifications” are mail content creation processes which inform user about new task assignments or daily task summaries.

Implementation

To customize a standard process you need to do the following:

1. Implement a process with a predefined process start signature in an ivy project. See the following sub chapters for more information.
2. Deploy the ivy project with the customized standard processes in the application.
3. Finally, the project with the standard processes must be activated in “app.yaml”:

```
# app.yaml located in <application-directory>/app.yaml which activates the portal default
# To enable these custom processes, the library id of the ivy project must be specified
# The library id is <group-id>:<project-id> from the ivy project deployment definition.
StandardProcess:
  DefaultPages: ch.ivyteam.ivy.project.portal:portalTemplate
  MailNotification: ch.ivyteam.ivy.project.portal:portalTemplate
```

Default Pages

To customize default pages, you must implement processes with a predefined process start signature. Checkout the process *Processes/Workflow/Home* in the JsfWorkflowUI which also overrides all types of default pages. The JsfWorkflowUI can be found in *[engineDir]/projects/JsfWorkflowUi.iar*.

The following default pages can be customized:

Default Page	Process Start Signature
Application Home Page as the entry page to the application.	DefaultApplicationHomePage()
Task List with all tasks the current user can work on.	DefaultTaskListPage()

Default Page	Process Start Signature
Process Start List with all processes which the current user can start.	DefaultProcessStartListPage()
End Page which will be displayed to the user after a task or process is completed.	DefaultEndPage(Number endedTaskId)
Login Page which comes up whenever authentication is needed.	DefaultLoginPage(String originalUrl)
Error Page which visualizes error on the front end	<i>no signature: globally defined in "web.xml"</i>

Table 7.2. Default Pages

The screenshot shows a web browser window with the title 'Axon.ivy Workflow'. The address bar displays 'localhost:8080/ivy/faces/instances/jsfworkk'. The page header features the 'AXON ivy' logo with the tagline 'digitalize your business'. A navigation bar contains links for 'Task List', 'Process List', 'Business Cases', and 'Task History'. The main content area is titled 'Task List' and includes the subtitle 'List of current Tasks'. A search bar is located on the right side. Below the search bar is a table with a header 'Name' and three rows of tasks:

	Name
	▶ Approve Peter Parker's bill of expenses
	▶ Accident report of Adam Bien
	▶ Approve John Smith's bill of expenses

At the bottom of the page, it states 'Powered by Axon.ivy Workflow Copyright © 2001 - 2018 AXON IVY AG'.

Figure 7.2. Task List provided by JsfWorkflowUI

Email Notifications

To customize the content of the email notification, you must implement processes with a predefined process start signature. Checkout the processes *Processes/NewTaskMailContent* and *Processes/DailyTaskSummaryMailContent* in the

JsfWorkflowUI which also provides standard processes for email notification. The JsfWorkflowUI can be found in *[engineDir]/projects/JsfWorkflowUi.iar*.

Mail notifications require a configured mail server and enabled notification settings as described in “Email”.

The following email notifications can be customized:

Email Notification	Process Start Signature
New Task page with the new assigned task. This is done everytime a new task is created, an existing task expires or the creator of an existing tasks changes. These events affect the user directly, via his role or his substitution.	MailNotification_NewTask(Number notificationUserId, Number notificationTaskId)
Daily Task Summary page with all open tasks for the user. This notification is executed once a day for each user.	MailNotification_DailyTaskSummary(Number notificationUserId)

Table 7.3. Email Notification

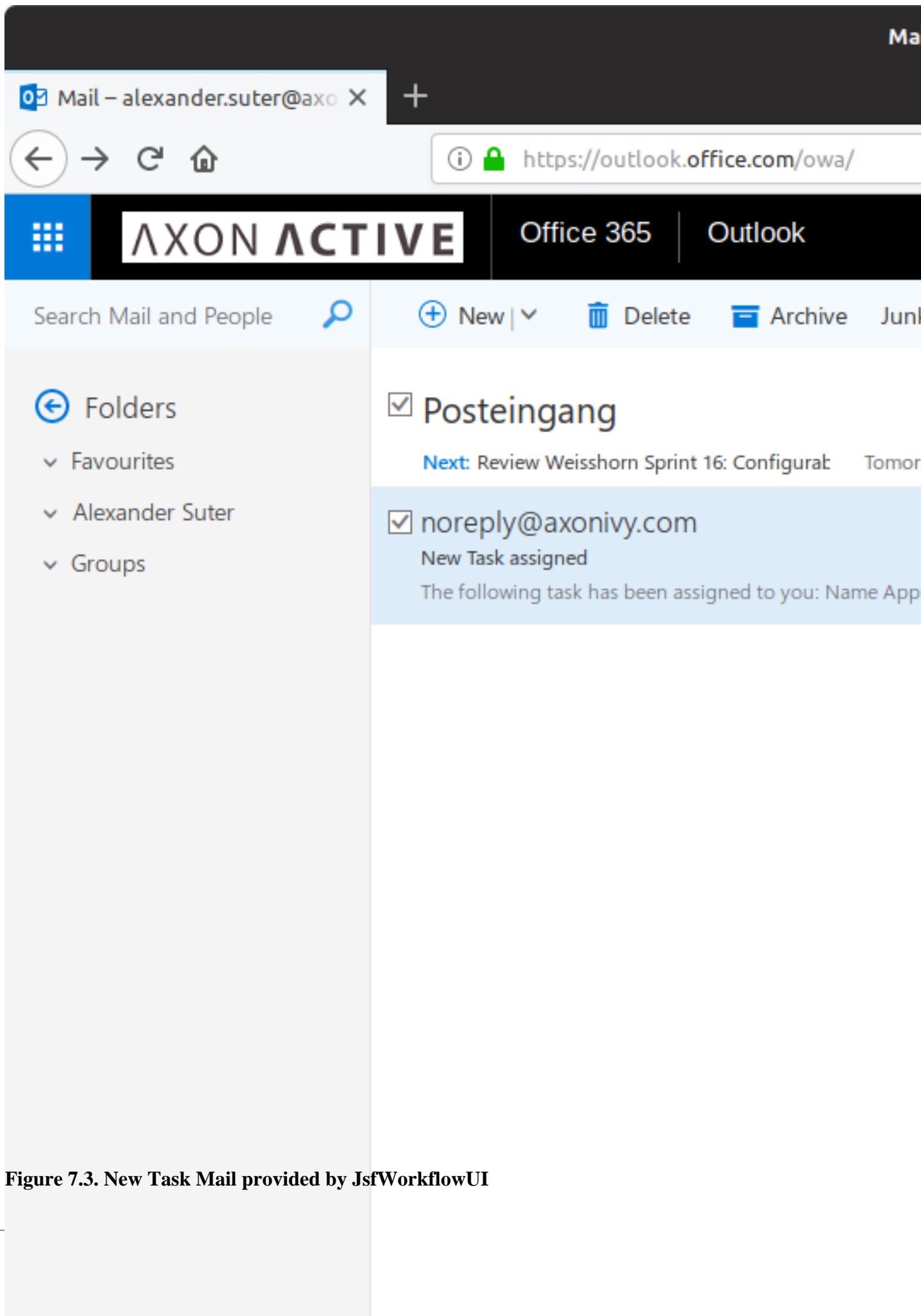


Figure 7.3. New Task Mail provided by JsfWorkflowUI

Implementation hints

- **Subject** : The `<title>` html tag is re-used as email subject
- **Images** : can be referred in the html either from CMS or the file system. These resources will be embedded as mime parts. External images will not be embedded and the links will not be modified at all.
- **Skip** : you can avoid the standard mail sending procedure by custom conditions (e.g. do not send mails to technical users). To do so implement an alternative process flow without displaying any User Dialog or Html Page activity.
- **API**: some API hints to implement email notification processes:
 - get the user you can use `ivy.session.getSecurityContext().findUser(notificationUserId)`
 - get the task you can use `ivy.wf.findTask(notificationTaskId)`
 - get the current open task for a user `ivy.wf.findWorkTasks(...)`

Miscellaneous

GC Optimization

The GC (Garbage Collection) of Java cleans up the unused memory of the JRE. Normally the GC completes in a few milliseconds. If it takes longer (and leads to serious issues for running applications) the optimization below can help to optimize the GC time.

Default GC configuration

By default, the GC strategy is optimized for RIA Applications and an explicit full concurrent GC runs every 10 minutes.



Note

Why a periodical GC is required for RIA Applications?

Normally a GC is triggered in the background when a considerable amount (e.g. 80%) of the available memory is used. Then the GC cleans up all unused memory so that the application can always address new memory as required.

Now comes RIA into play. A RIA application creates each UI widget on server- and client-side to share the UI-state on both sides. To clean this up on both sides the widget must be cleaned first on server-side before it can be cleaned on the client-side. But with the default GC configuration the memory on server-side will not be cleaned until a considerable amount of the available memory is used. But the available memory on server-side is usually considerably higher than on client-side, so this can lead to low memory problems on client side (OutOfMemoryException).

To prevent this situation Axon.ivy Engine triggers a full concurrent GC every 10 minutes. This cleans up the memory on server-side and allows the client-side to clean up its memory too.

Optimization for JSF

In JSF applications you only need a Browser on client side. Therefore, no periodical full concurrent GC is required and you can optimize the GC on low latency.

To change the GC accordingly comment out the following line in the corresponding `ilc-file` or `AxonIvyEngine.conf` for Linux:

```
# * GC optimized for JSF
ivy.garbage.collector.options=-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled
```

System Database Encryption

User passwords are stored encrypted in the system database. Passwords of Axon.ivy users are hashed by using the bcrypt algorithm. Passwords of technical users that are used to communicate with external systems are encrypted using the AES algorithm. The secret key for the AES algorithm is by default created automatically by using a secure random generator. However, it is possible to provide an own secret key as follows:

1. Create your own AES secret key and store it in a key store file by using the Java keytool:

```
keytool -genseckey -alias aes -keyalg AES -keysize 128 -storepass changeit -storetype JCEKS
```

2. Configure the following Java system properties in the launcher configuration:

Java System Property	Description
ch.ivyteam.ivy.persistence.keystore.file	The path to the key store that holds the AES secret key. E.g. <code>keystore.jceks</code> .
ch.ivyteam.ivy.persistence.keystore.password	The password needed to read the key store file. Default value <code>changeit</code> .
ch.ivyteam.ivy.persistence.keystore.alias	The name of the key to read from the key store file. Default value <code>aes</code> .
ch.ivyteam.ivy.persistence.keystore.type	The type of the key store. Default value <code>JCEKS</code> .

Table 7.4. AES Secret Key System Properties



Warning

If you configure to use an own AES secret key after you have already stored technical passwords for external system then those passwords can no longer be decrypted and are useless. You have to reconfigure all those passwords again!

Replacing Java Runtime with newer version

If you want to use profiling tools such as VisualVM it may be required to replace the bundled Java Runtime (JRE) with a newer one. To do so, just perform the following steps:

1. Download the full Java Development Kit (JDK) that suits best for your OS and Ivy-Installation (a 32 bit Ivy Engine cannot be run with a 64 bit runtime). It is important that you download the JDK and not the JRE. You will find the JDK at the Sun/Oracle Java website
2. Install the downloaded JDK on the server (or on any computer with the same OS)
3. Move the whole content of the directory `jre` in the installation folder of your Axon.ivy Engine to some other folder (to have a backup).
4. Copy the whole content of the folder `jre` in the installation directory of the JDK to the `jre` folder in the installation folder of your Axon.ivy Engine (the one you moved in the step above)



Warning

Changing the Java Runtime that is packaged with Axon.ivy Engine is not recommended and should only be done when requested by Axon.ivy support or when required for testing purposes.

Chapter 8. Monitoring

Logging

Axon.ivy uses a library called Log4j from Apache Foundation to log certain events. The logging configuration file is located in the *{Axon.ivy Install Directory}/configuration* directory and is called “log4jconfig.xml”. By default log events are written to the console and to log files. The log files are written to the *{Axon.ivy Install Directory}/logs* directory.

Logging type	Log level
Console log	WARN
File log	INFO
Windows Event Log	FATAL

Table 8.1. Default settings for logging

The log levels are used as follows:

FATAL This level is used to report problems that may cause the engine not to work correctly.

ERROR This level is used to report problems that something has not worked as expected and may cause that the user gets an error message on the UI.

WARN This level is used to report problems that have to be solved because it can lead to errors later.

INFO This level is used to report that something was done. (E.g. for example a database call)

DEBUG This level is used to report internal events. Most of these events are only interesting for developers. However, some of them may also be interesting for troubleshooting.

Feel free to change the logging configuration to your needs.

Log Message Format

A log message looks like the following:

```
10:19:14.173 INFO [ch.ivyteam.ivy.richdialog.exec.internal.panel.RichDialogPanelImpl] [http-8082-4]
[application=0, client=127.0.0.1, task=4, pmv=System$Administration$1, session=4, request=Ulc over HTTP POST
115D746C75FAF428/start1.ivp, executionContext=SYSTEM]
Can not restore UI state. No user is logged in.
```

The first entry of a log message is the exact time it was written (10:19:14.173). Followed by the log level of the message (INFO). Next is the log category ([ch.ivyteam.ivy.richdialog.exec.internal.panel.RichDialogPanelImpl]). Then the name of the thread in which context the log message was written follows ([http-8082-4]). The next section contains a lot of Axon.ivy context information. For example the user session or the process model version that were active when the log message was written. The content of the context information can change depending on the context the log message was written. The following context information exists:

application The identifier of the current application.

client The IP address and maybe the host name of the current web client.

executionContext The security execution context that is used to check permissions. This can be the current session or SYSTEM if security is disabled.

request Information about the current request is written to the log.

requestId	The identifier of the current request. Can be used to filter all messages that are written in the context of the same request.
pmv	The identification string of the current process model version.
processElement	The process element that is currently executed.
rd	The fully qualified name of the current Rich Dialog.
session	The current Axon.ivy session. The identifier of the session and the user name (if a user is logged in).
task	The identifier of the current task.

On the next line the message that was logged follows. In case of errors a java exception stack trace may follow on the next lines.

Runtime Log

On the Axon.ivy Designer certain events of processes are logged to the runtime log view. The process designer itself can write to the runtime log using the `ivy.log` object. On the Axon.ivy Engine all information written to the runtime log is handled by Log4j. It is written to the console, to log files and to the Windows Event Log.

The runtime log entries are written to special log categories which names start with **runtime** followed by the application name, the process model name, and the runtime log category. For example: the category name **runtime**`log.app.hrm.user_code` represents the runtime log of the application called **app**, with the process model called **hrm** and the runtime log category **user_code**.

Example

The following xml snip can be added to the Log4j configuration file so that the runtime log of the process model **hrm** of the application **app** is written to its own log file called `runtime``log.app.hrm.log`:

```
<!-- Defines a log file called runtime
```

Request/Performance Logging

If you want to know the time when a request was received from the Axon.ivy Engine and at what time the request processing of the engine was done, then you use the following log category:

```
ch.ivyteam.ivy.webserver.internal.PerformanceLogValve
```

Configuration Example (configuration/"log4jconfig.xml"):

```
<!-- Configures that the log category
ch.ivyteam.ivy.webserver.internal.PerformanceLogValve has priority DEBUG -->
```

```
<category name="ch.ivyteam.ivy.webserver.internal.PerformanceLogValve"
  class="ch.ivyteam.log.Logger">
  <priority value="DEBUG"/>
</category>
```

The log category logs the entry of a request right after the internal web server has received it. The exit is logged after the request was processed by the web server. In the exit log message you find the duration of the request in microseconds.

The log level of these messages is DEBUG. Change the threshold of the appenders to DEBUG so that log messages with this priority are written to the appender's destination.

Configuration Example (configuration/"log4jconfig.xml"):

```
<appender name="FileLog" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="Threshold" value="DEBUG"/>
  <param name="File" value="${user.dir}/logs/ivy.log"/>
  <param name="DatePattern" value=".'yyyy-MM-dd"/>
  <layout class="org.apache.log4j.IvyLog4jLayout">
    <param name="DateFormat" value="HH:mm:ss.SSS"/>
  </layout>
</appender>
```

If you want to know what the Axon.ivy Engine has done between the entry and exit of the request you can use the context information `requestId` which you can find on every log message. A unique request identifier is assigned to every request. By filtering the log for messages with the same `requestId` you find out what kind of operations Axon.ivy Engine has done during the request.

Example:

```
10:49:40.904 DEBUG [...rformanceLogValve] [http-8081-1] [requestId=43]
  Entry url=http://localhost:8081/ivy/pro/designer/OpenEditor/13224891E742EE17/start4.ivp
  client=0:0:0:0:0:0:1 session=null httpsession=C900A5BC35251533DEB5B36E4316EE98
10:49:41.020 INFO [...nEditor.user_code] [http-8081-1] [application=2147483647,
  client=0:0:0:0:0:0:1, requestId=43, task=1, pmv=designer$OpenEditor$1, processElement=13224891E742EE17-f26-
  t, session=1, request=HTTP GET test.mod/start4.ivp(1.1.0.0), executionContext=1]
  This is my log message
10:49:41.050 INFO [...ner.OpenEditor.db] [Process Extension Thread 1] [application=2147483647,
  client=0:0:0:0:0:0:1, requestId=43, task=1, pmv=designer$OpenEditor$1, processElement=13224891E742EE17-f29-
  bean, session=1, request=HTTP GET test.mod/start4.ivp(1.1.0.0), executionContext=SYSTEM]
  Execute database statement SELECT * FROM IWA_ACCESSCONTROL
10:49:41.050 INFO [...ner.OpenEditor.db] [Process Extension Thread 1] [application=2147483647,
  client=0:0:0:0:0:0:1, requestId=43, task=1, pmv=designer$OpenEditor$1, processElement=13224891E742EE17-f29-
  bean, session=1, request=HTTP GET test.mod/start4.ivp(1.1.0.0), executionContext=SYSTEM]
  Executed database statement successfully in 0 milli seconds
10:49:41.100 DEBUG [...rformanceLogValve] [http-8081-1] [requestId=43]
  Exit url=http://localhost:8081/ivy/pro/designer/OpenEditor/13224891E742EE17/start4.ivp
  client=0:0:0:0:0:0:1 session=1 httpsession=C900A5BC35251533DEB5B36E4316EE98 duration=194181 us
```

In the example above you see the log messages when the request with the id 43 has entered and exited the web server. There was also one user runtime log message written in the same request and one database call that has lasted 0 milliseconds. The whole request needed 19.418 ms to be processed.

Process Element Performance Statistic and Analysis

Configure Process Element Performance Statistic on Axon.ivy Engine

On an Axon.ivy Engine it is possible to dump out performance statistic informations, periodically into a CSV formatted file. This allows to analyse the performance of the engine and to detect long running and performance intensive process elements and processes. The file contains detailed informations of each executed process element since the last dump.

After activation the informations are collected and written to the log-directory of the Axon.ivy Engine installation. The file contains the following name: performance_statistic_####-mm-tt_hh-mm-tt.csv (e.g. performance_statistic_2011-03-15_09-21-05.csv)

Process element performance statistics are not collected by default. They can be enabled in the *ProcessEngine* section of the “ivy.yaml” file.

Analyse the Performance Statistic

All time values are in milliseconds. The execution of some process elements are separated in two categories internal and external.

Internal Category The internal category is used for the execution time in the process engine itself without the external execution.

External Category The external category is used for execution time in external systems. In the table below the process elements are listed which use the external category.

Process Element	Internal Category	External Category
Database Step	Parameter-mapping, caching, output-mapping and ivyScript execution.	The execution of the SQL statement on the database server.
Web Service Call Step	Parameter-mapping, caching, output-mapping and ivyScript execution.	The execution of the Web Service on the web server.
E-Mail Step	Parameter-mapping	The interaction with the Mail-Server.
Program Interface		The execution of the defined Java-Class.

Table 8.2. Process elements with usage of external category

For each executed process element one entry in the view is created. See the table below which information is available.

Name	Description
Entry ID	Entry ID, useful to order the entries by its first execution.
Application	Application of the process element.
Process Model	Process Model of the process element.
PM Version	Process Model Version of the process element.
Process Path	The path to the process.
Element ID	The identifier of the process element.
Element Name	The first line of the process element name (display name).
Element Type	The type of the process element.
Total Time	Total time [ms] of internal and external execution.
Int. Executions	Total internal executions of the process element.
Total Int. Time	Total internal time [ms] of process engine executions.
Min. Int. Time	Minimum internal process engine execution time [ms].
Avg. Int. Time	Average internal process engine execution time [ms].
Max. Int. Time	Maximum internal process engine execution time [ms].
Ext. Executions	Total external execution count.

Name	Description
Total Ext. Time	Total external execution time [ms].
Min. Ext. Time	Minimum external execution time [ms].
Avg. Ext. Time	Average external execution time [ms].
Max. Ext. Time	Maximum external execution time [ms].

Table 8.3. Column Description



Tip

To find a process element by its *Element ID*, use the search dialog *Find process or element* in the Axon.ivy Designer. Use menu *Axon.ivy > Debug > Find process or element*.

Java Management Extensions (JMX)

Java Management Extensions (JMX) is a technology to read and write runtime information from a java processes. This allows monitoring tools to monitor the state the Axon.ivy Engine, e.g. with VisualVM, Java Mission Control or Nagios. A monitoring tool that runs on the same machine and with the same user as the Axon.ivy Engine can connect to Axon.ivy Engine without any additional configuration.

Activate Remote Access

If the Axon.ivy Engine is running under another user or on a remote host than the monitoring tool, then JMX remote access has to be activated. Remote access is protected by a user name and password of an Axon.ivy Engine Administrator, so all Axon.ivy Engine Administrator have access.

On **Windows**: activate remote access by uncommenting the following line in an ivy launch control file **.ilc* (See “Windows Program Launcher Configuration”):

```
ivy.management.port=9003
```

On **Linux**: activate remote access by uncommenting the following line in the “AxonIvyEngine.conf”:

```
#JAVA_OPTS="$JAVA_OPTS -Dcom.sun.management.jmxremote.port=9003 -Dcom.sun.management.jmxre
```

Auto Discovery (JDP)

Some monitoring tools can auto discover running JMX servers in the network. So that the user does not have to know the JMX ip and port.

On **Windows**: Auto discovery is enabled by default if JMX is activated. However, you can disable this feature by uncommenting the following line in the **.ilc* file (See “Windows Program Launcher Configuration”):

```
ivy.management.autodiscovery=false
```

On **Linux**: Auto discovery is enabled in the pre-configured (but uncommented) *JAVA_OPT* of the “AxonIvyEngine.conf”. But you can disable it at any time by changing its value to *false*:

```
-Dcom.sun.management.jmxremote.autodiscovery=false
```

Provided MBeans

The Axon.ivy Engine provides performance and management information by a set of MBeans. These allows to monitor internals of the Axon.ivy Engine. Most monitoring tools provide a user interface to browse the available MBeans. MBeans are mostly shown in a tree which is built with the information provided in the names of MBeans.

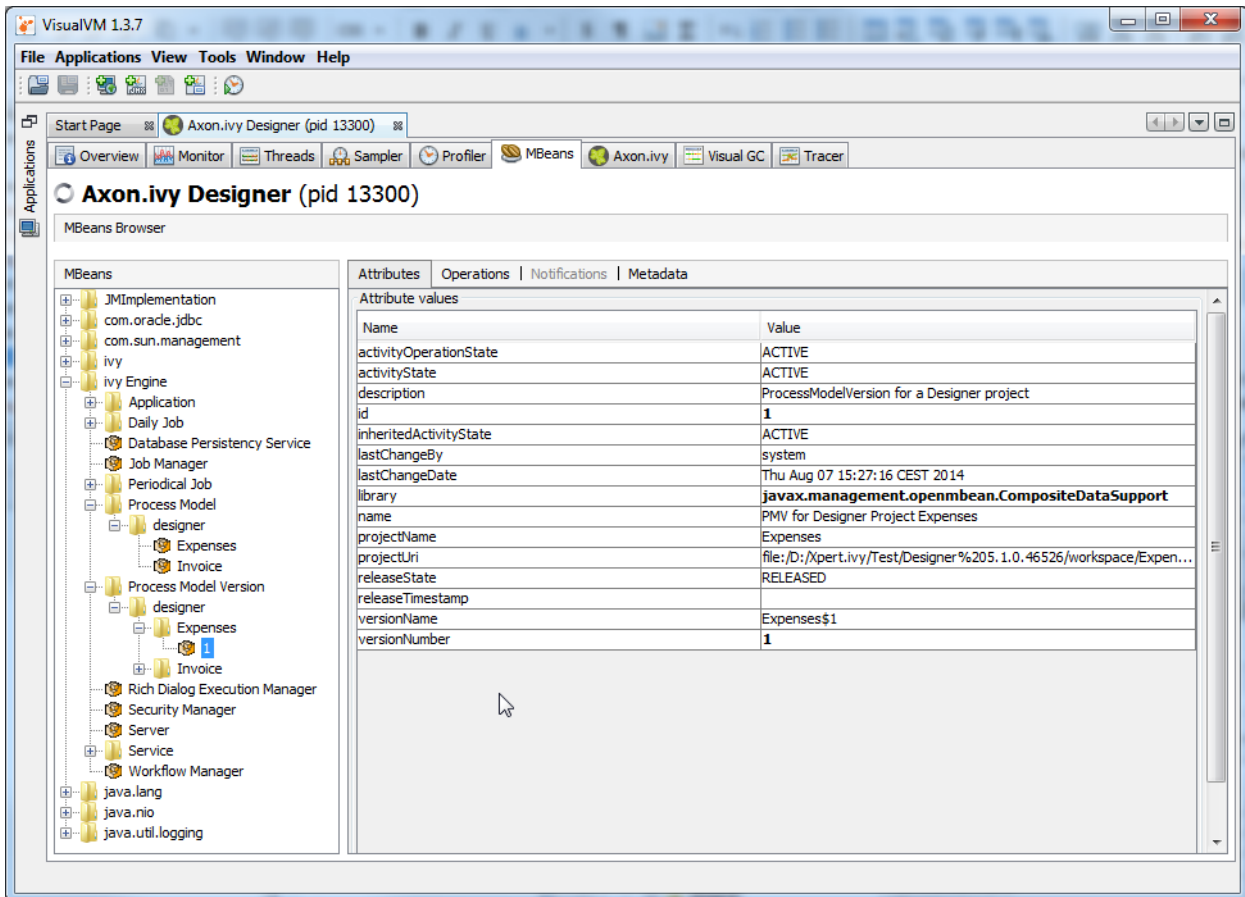


Figure 8.1. MBeans Tree of Axon.ivy shown in MBeans Plugin of VisualVM

The names of MBeans provided by Axon.ivy are structured so that the name contains the Application, Process Model, Process Model Version or Environment where this is reasonable.



Note

Examples of typical Axon.ivy MBean names:

```
ivy Engine:type=External Web Service,application=MyApplication,environment=Default
ivy Engine:type=Job Manager
ivy Engine:type=Process Start Event Bean,application=MyApplication,pm=MyProcessModel
```

The name and description of a MBean can be found in its meta information (see the *Metadata* tab in the MBeans tab of VisualVM). MBeans provide information through attributes and operations. The description of the attributes and operations can also be found in its meta information (see too the tool tips in the *Attributes and Operations* tab of the MBeans tab of VisualVM).



Warning

Manipulating attribute values or calling operations on MBeans will immediately change the configuration of your system and can therefore harm your running applications.

If not mentioned otherwise, a manipulation only affects the currently running engine. The manipulation will not survive a engine restart.

Manipulations that survive a engine restart contain the following text in the description of the attribute or operation: `(Persistent)`.

In addition to the MBeans provided by Axon.ivy some third party libraries included in Axon.ivy provide their own MBeans. One of them is Apache Tomcat that is used as internal web server. Its MBeans provide information about the handling of HTTP requests like request count, errors, execution time, sessions, etc. Moreover, the Java virtual machine also provides some MBeans that provide information about the used memory (Java heap), CPU usage, uptime, etc.

Below a not complete list of provided information:

- External Database (connections, transactions, errors, execution time, etc.)

```
ivy Engine:type=External Database,application=*,environment=*,name=*
```

- Web Service (calls, errors, execution time, etc.)

```
ivy Engine:type=External Web Service,application=*,environment=*,name=*
```

- REST Web Service (calls, errors, execution time, slow calls, etc.)

```
ivy Engine:type=External REST Web Service,application=*,environment=*,name=*
```

- System Database (connections, transactions, errors, execution time, etc.)

```
ivy Engine:type=Database Persistency Service
```

- HTTP Requests (count, errors, execution time, etc.)

```
*:type=GlobalRequestProcessor,name=*
```

- Number of Sessions (HTTP sessions, Axon.ivy sessions, licence relevant sessions, Rich Dialog client sessions, etc.)

```
ivy Engine:type=Security Manager
ivy Engine:type=Rich Dialog Execution Manager
*:type=Manager,context=*,host=*
```

- Background jobs (name, next execution time, etc.)

```
ivy Engine:type=Job Manager
ivy Engine:type=Daily Job,name=*
ivy Engine:type=Periodical Job,name=*
```

- Process Start Event Beans (polls, executions, errors, execution time, etc.)

```
ivy Engine:type=Process Start Event Bean,,application=*,pm=*,pmv=*,name=*
```

- Process Intermediate Event Beans (polls, firings, errors, execution time, etc.)

```
ivy Engine:type=Process Intermediate Event Bean,application=*,pm=*,pmv=*,name=*
```

- Application, Process Model and Process Model Version, Library information (activity state, release state, name, description, etc.)

```
ivy Engine:type=Application,name=*
ivy Engine:type=Process Model,application=*,name=*
ivy Engine:type=Process Model Version,application=*,pm=*,name=*
```

- Cluster, Cluster Nodes and Cluster Communication information (received and sent message, errors, execution time, etc.)

```
ivy Engine:type=Cluster Manager
ivy Engine:type=Cluster Channel
```

- Thread Pool information (core, maximum and current pool size, active threads, queue size)

```
ivy Engine:type=Thread Pool,name=Background Operation Executor
ivy Engine:type=Thread Pool,name=Immediate Job Executor
```

```
ivy Engine:type=Thread Pool, name=Scheduled Job Executor
```

- System Database and CMS Cache

```
ivy Engine type=CacheClassPersistencyService,name=* [clearCache()]
ivy Engine type=CacheClassPersistencyService,name=*,strategy=CacheAll [maxBytesToCache,
ivy Engine type=CacheClassPersistencyService,name=*,strategy=CacheAllRemoveUnused [maxBy
ivy Engine type=CacheClassPersistencyService,name=*,cache=LongBinaries [readHits, readMi
ivy Engine type=CacheClassPersistencyService,name=*,cache=LongCharacters [readHits, read
ivy Engine type=CacheClassPersistencyService,name=*,cache=ObjectsAndAssociations [object
```

- Memory (Java Heap, Perm Gen)

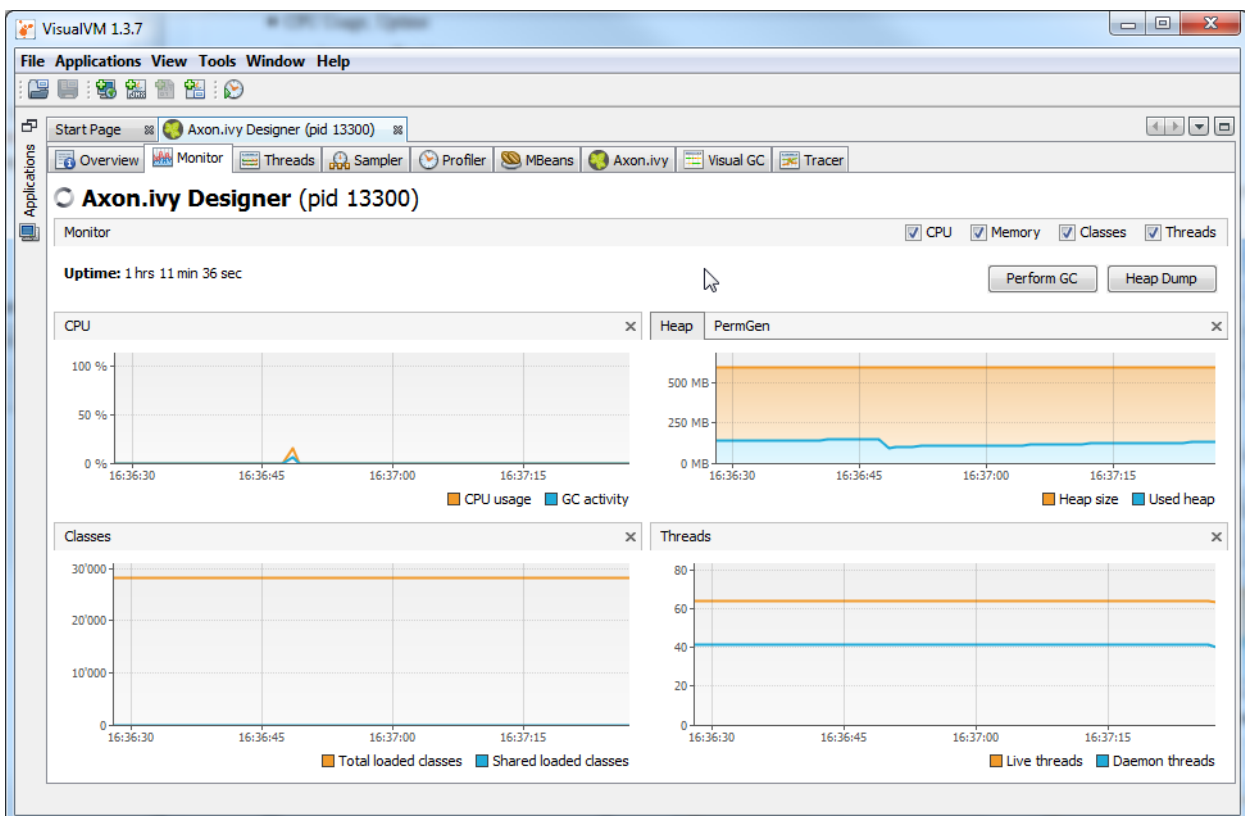
```
java.lang:type=Memory
```

- CPU Usage, Uptime

```
java.lang:type=Runtime
java.lang.type=OperatingSystem
```

VisualVM

We recommend to use VisualVM to monitor Axon.ivy Engine processes. VisualVM allows you to monitor the memory and CPU usage of the Axon.ivy Engine process. It can be used to analyze problems in your Axon.ivy projects like memory leaks or thread dead locks.



VisualVM can connect to all Java processes running on the same host and with the same user. In addition you can use JMX (See section Java Management Extension for more information) to connect VisualVM to processes that run with another user (e.g. as Windows Service) or on remote machines.

VisualVM is available from <https://visualvm.github.io/> or as `jvisualvm` in the `bin` directory of a Oracle JDK (Java Developer Kit).

Axon.ivy Plugin for VisualVM

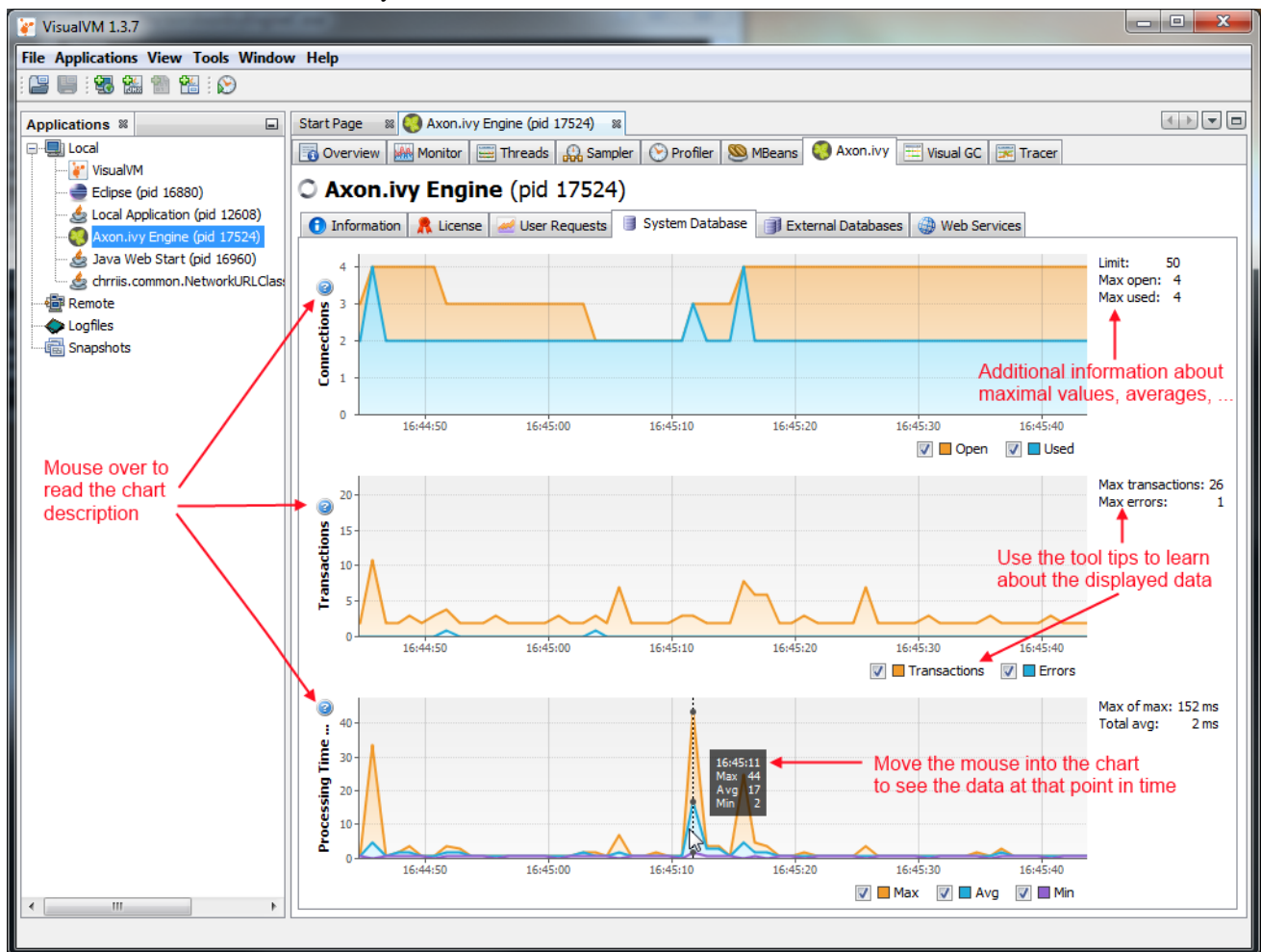
In the delivery of Axon.ivy we provide a dedicated plugin for VisualVM that allows you to monitor some of the technical aspects of an Axon.ivy Engine or Designer. For example you can observe the current transactions on the System Database, whether you violate the licence or how many requests are running at an Axon.ivy Engine at any given time. And in the same tool you can still observe the heap or CPU usage or create thread dumps.



Note

VisualVM is a tool to observe the current state of the monitored engine. It is not intended for long-time observation, recording or even alarming. If you want to do that, make use of the JMX extensions of Axon.ivy. in combination with tools like Nagios or IBM Tivoli.

The plugin itself should be mostly self explanatory. It consists of multiple tabs for the different aspects. Most of the tabs contain a number of charts that always have a similar structure:



Installation

1. Make sure that you have an installation of VisualVM. If you use a standalone version of VisualVM, please make sure that you use at least version 1.3.7.
2. Run VisualVM (in JDK go to the *bin* folder and start **jvisualvm**)
3. Go to the **Tools/Plugins** menu
4. Change to *Downloaded* tab and click on the *Add Plugins...* button

5. In the file chooser that appears, navigate to the subfolder *misc/visualvm* in your engine installation directory and choose the *visualvm-plugin.nbm*.
6. Follow the instructions in the installation wizard.
7. Choose the option to restart VisualVM at the end of the installation wizard.

Chapter 9. Tool Reference

AxonIvyEngine

Axon.ivy Engine program. This program starts an instance of the Axon.ivy Engine.



Tip

An Axon.ivy Engine can also be started as service. More information can be found in the section Engine Service chapter.

Options

The following options are available for the Axon.ivy Engine program:

Option	Description	Mandatory
-start	Starts the engine. Same behaviour as if no options are given. Allows to stop the engine by pressing a key in the console if a console is available.	no
-startdaemon	Starts the engine. Does not allow to stop the engine by pressing a key in the console.	no
-stop	Stops the engine. Only initiate the stop but will not wait until the engine has really stopped.	no
-stopdaemon	Stops the engine. Will wait until the engine has really stopped.	no
-status	Prints the current status of the engine.	no

Table 9.1. AxonIvyEngineConfig Options

Launchers

The following program launchers are available for the Axon.ivy Engine program:

Platform	Console support	Launcher
Windows	no	bin/AxonIvyEngine.exe
Windows	yes	bin/AxonIvyEngineC.exe
Linux	yes	bin/AxonIvyEngine

Table 9.2. AxonIvyEngine Launchers

EngineConfigCli

The console program is used to configure the Axon.ivy Engine. E.g. configure, create or convert the database.

Options

Option	Description
help	Shows which commands and options are possible.

Option	Description
<command> help	Get help to a specific command e.g. <code>EngineConfigCli config-db help</code>

Table 9.3. EngineConfigCli Options

Engine Service

In productive environments it is recommended to run the Axon.ivy Engine as a service.

Windows Service

Axon.ivy Engine Windows Service. This program is the implementation of the Axon.ivy Engine Windows Service. But it can also be used to register, unregister, start and stop Axon.ivy Engine as Windows Service.



Note

You can also register, unregister, start and stop the Axon.ivy Engine Windows Service with the Control Center.

Launchers

The following program launcher is available for the Axon.ivy Engine Service program:

Platform	Launcher
Windows	bin/AxonIvyEngineService.exe

Table 9.4. AxonIvyEngineService Launchers

Options

The following options are available for Axon.ivy Engine Service program:

AxonIvyEngineService [-start|-stop|-register [username password]]-unregister]

Options	Parameters	Description	Mandatory
-start		Starts the Axon.ivy Engine Windows Service	no
-stop		Stops the Axon.ivy Engine Windows Service	no
-register		Registers the Axon.ivy Engine Windows Service within Windows	no
	username	The user name of the user in which context the windows service should run	no
	password	The password of the user in which context the windows service should run	yes, if username is specified
-unregister		Unregisters the Axon.ivy Engine Windows Service from Windows	no

Table 9.5. AxonIvyEngineService Options

Linux Service (systemd)

The install service program helps to install the Axon.ivy Engine as a systemd Linux daemon. To install the service:

1. Change current directory to to the bin/ directory of your Engine: `cd bin/`

2. Run following command as root: `./InstallService.sh`
3. Accept the directory of your engine installation.
4. Set the user and group under which the Engine service should run. Must not be `root`. Typically, a special user with limited access right should be used.
5. Start the Engine service with `systemctl start AxonIvyEngine.service` to check if it works.
6. Check the current status of the service with `systemctl status AxonIvyEngine.service`
7. If you want to start the Engine service on the system start, execute following command: `systemctl enable AxonIvyEngine.service`



Tip

For more information about systemd services consult `man systemd` and `man systemctl`.

Launchers

Platform	Console support	Launcher
Linux	yes	bin/InstallService.sh

Table 9.6. InstallService Launchers

Launch Configuration

Windows Program Launcher Configuration

All windows program launchers can be reconfigured with an additional ivy launch control file (*.ilc). The ivy launch control file must have the same name like the launcher itself but instead of the extension `*.exe` it must use an extension `*.ilc`.



Tip

If you want to reconfigure the `AxonIvyEngine.exe` launcher, then copy the `Example.ilc` file and rename it to `AxonIvyEngine.ilc`.

The ivy launch control file is a text-based property file. The file has the following format:

```
# comment line
property=value
property=value
```

Open the file with a text editor to reconfigure it. Most properties found in the ivy launch control file are used to modify java virtual machine options. The following list shows all available options and explains them:

Property	JVM Option	Description
<code>ivy.heap.max.ratio</code>	yes	The maximum heap size (-Xmx) in percentage of the physical memory of the machine.
<code>ivy.heap.max.size</code>	yes	The maximum heap size (-Xmx) in megabytes.
<code>ivy.heap.start.size</code>	yes	Start heap size (-Xms) in mega bytes.

Property	JVM Option	Description
ivy.heap.free.max.ratio	yes	The maximum free heap memory (-XX:MaxHeapFreeRatio) in percentage of the current heap size.
ivy.heap.free.min.ratio	yes	The minimum free heap memory (-XX:MinHeapFreeRatio) in percentage of the current heap size.
ivy.heap.young.max.size	yes	The maximum young heap size (-XX:MaxNewSize) in megabytes.
ivy.heap.young.min.size	yes	The minimum young heap size (-XX:NewSize) in megabytes.
ivy.heap.eden.survivor.ratio	yes	The survivor heap size as ratio between the eden and the survivor heap size (-XX:SurvivorRatio)
ivy.heap.tenured.young.ratio	yes	The young heap size as ratio between the tenured and the young heap size (-XX:NewRatio).
ivy.jvm.type	yes	The Java virtual machine type to use (ClientHotspotJVM, ServerHotspotJVM).
ivy.dir.aux	no	The directory where the ivyTeam jars are located.
ivy.dir.jre	no	The directory where the java runtime environment is located.
ivy.java.main.class	no	An own Java class to launch instead of ivy engine's main starter class.
ivy.java.main.method	no	Another Java static method to launch on the <i>ivy.java.main.class</i> instead of the default main method. The called method should take the same arguments as a Java main method.
ivy.vm.additional.options	yes	Additional Java virtual machine arguments
ivy.garbage.collector.options	yes	Additional garbage collector arguments. See too GC Optimization.
ivy.windows.service.name	no	The name of the Windows service (only for Windows service launcher).
ivy.application.name	no	The name of the application (only for application launcher).
ivy.application.singleton	no	Is the application a singleton (true, false; only for application launcher).

Table 9.7. Ivy Launch Control Properties

The properties that are bold are the most used properties.

Linux Launcher Configuration

The Java virtual machine (JVM) options for the Engine are configured in the *AxonIvyEngine.conf* file. For all other helper programs the JVM options are configured in the *control.conf* file.



Tip

If you want to configure the memory (-Xmx, -Xms, etc.) or optimize any other JVM options of the Engine then edit the *AxonIvyEngine.conf* file.

AxonIvyEngine.conf

```
#
# ivyTeam Launch Configuration for the Engine
#
# =====
#
#
# Specify Ivy options to pass to the Java VM.
# Don't modify them unless you know what you do.
#
if [ ! "x$JAVA_OPTS" = "x" ]; then
```

```

    echo "JAVA_OPTS already set in environment; will override default settings"
fi

JAVA_OPTS="-Xverify:none -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEr

#
# Garbage collector optimization arguments (for more information see: EngineGuide -> Conf
# Note: Uncomment only one setting of the following two GC optimizations
#
# * GC optimized for JSF
#
#JAVA_OPTS="$JAVA_OPTS -XX:+DisableExplicitGC"
#
#
# * GC settings optimized for RIA with explicit full concurrent gc every 10 minutes (defau
#
#JAVA_OPTS="$JAVA_OPTS -XX:+ExplicitGCInvokesConcurrent -Dsun.rmi.dgc.server.gcInterval=600

#
# Max heap size (-Xmx)
#
#JAVA_OPTS="$JAVA_OPTS -Xmx2048m"

#
# Start heap size (-Xms)
#
#JAVA_OPTS="$JAVA_OPTS -Xms128m"

#
# Headless mode that does not require an X11 environment
#
#JAVA_OPTS="$JAVA_OPTS -Djava.awt.headless=true"

#
# To enable access to the java mangement extension (JMX) server from remote hosts, or from
# You may want to configure another tcp/ip port.
# Access to the management server is protected by default by the java authentication and a
# The file configuration/jaas.config is used as configuration file for JAAS. It must conta
# By default the java discovery protocol (JDP) is enabled.
#
#JAVA_OPTS="$JAVA_OPTS -Dcom.sun.management.jmxremote.port=9003 -Dcom.sun.management.jmxre

#
# Define the local network IP address or hostname of this computer on which the JMX port s
#
#JAVA_OPTS="$JAVA_OPTS -Djava.rmi.server.hostname=<IP of the machine>"

```

Engine Configuration UI

The Engine Configuration UI is a simple user interface that lets operators apply the basic configuration that is necessary to have a productive engine running. This includes especially the installation of a license and the creation of a system database.

Advanced users might prefer to use configurations files (“ivy.yaml”) and the “EngineConfigCli” to roll out and Axon.ivy Engine into production. Configuration files make the installation process faster re-reproducible in various environments such as *dev*, *test* and *prod*.

Usage: Apply configurations in the Tabs. Click *Save* to store the modified data on all tabs. Hit *Discard Changes* to reload the configuration from the filesystem and database.



Legacy Storage Format

The Engine Configuration UI stores the defined configurations in legacy storages such as the System Database (connector-properties, admins).

Since 7.2 users can stick to a file based configuration approach by using the “ivy.yaml”



Note

The changes that you make with the Engine Configuration do not become active unless you restart the engine.

Launchers

The following program launchers are available for the Axon.ivy Engine Configuration:

Platform	Console support	Launcher
Windows	no	bin/AxonIvyEngineConfig.exe
Windows	yes	bin/AxonIvyEngineConfigC.exe
Linux	yes	bin/AxonIvyEngineConfig

Table 9.8. AxonIvyEngineConfig Launchers

Via Control Center

After starting the ControlCenter, select a server entry from the server list on the left side and press the Server button in the configuration area to start the configuration program.

Windows: Start the *ControlCenter.exe* program in the *bin* directory of the Axon.ivy Engine installation directory.

Linux: Start the *ControlCenter* program in the *bin* directory of the Axon.ivy Engine installation directory to start the ControlCenter program.

Options

The following options are available for the Axon.ivy Engine Configuration program:

Option	Description	Mandatory
-headless	Activates the headless mode. Useful if no graphical user interface is installed on the server machine.	no

Table 9.9. AxonIvyEngineConfig Options

Licence

On the *Licence* tab you have to upload a valid licence:

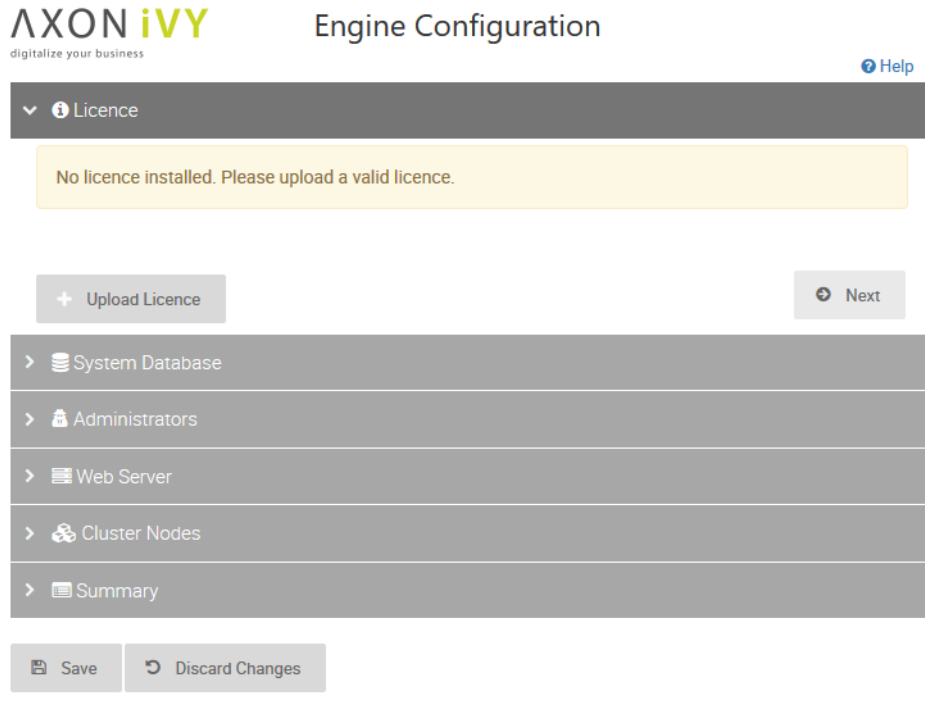


Figure 9.1. Axon.ivy Engine Configuration Licence Tab

Use the *Upload Licence* button to open the file browser and select the licence which should be used.



Note

It is possible to configure the engine without a valid licence, but the engine will always start in the demo mode if you do not have a valid licence and therefore does not use your configuration.

System Database

On the *System Database* tab the Axon.ivy Engine system database can be configured, created and converted:

AXON ivy digitalize your business Engine Configuration [Help](#)

> Licence

▼ System Database

Connection state unknown

Please check the connection to the Database.

Database: MySQL

Driver: mySQL

Host: dbServer

Port: 3306 Default

Database Name: myDatabase

Username: user

Password:

> Administrators

> Web Server

> Cluster Nodes

> Summary

Figure 9.2. Axon.ivy Engine Configuration System Database Tab

First choose the database system and the JDBC driver you want to use. At the moment the Axon.ivy Engine supports the following database systems:

- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- Postgre SQL

The choice of the second step depends on the database system and JDBC driver you have chosen in the first section. Click on the database system links above to find information about how to configure the connection settings. The applied db user needs the following privileges:

- CREATE DATABASE (to create the system database out of the Engine Configuration)

- CREATE, ALTER, DROP Tables, Views, Indexes, Triggers (to update the Axon.ivy Engine)
- INSERT, SELECT, UPDATE, DELETE data

In a third step you can configure additional connection properties. When clicking on the *Additional Properties* button a dialog will show, where you can add, edit or delete the properties. See database system specific chapter (links above) to find information which additional connection properties are available for the database system that you have chosen.

At the top of the page the state of the connection is visible. Use the button on the right to try to connect to the system database.

Create new System Database

If the system database does not exist, use the create button at the bottom to create a new system database. During the creation of a new database the configured connection parameters are used. For some database system additional information is necessary. It must be provided in a pop-up dialog before the new database can be created. See database system specific chapter (links above) to find what additional information is necessary for the chosen database system.



Note

You can previously create an empty database/schema. In this case the server configuration tool will only create the necessary tables into the given database/schema. If the database/schema doesn't exist already, the server configuration tool creates it with a best practice configuration. In this case the applied db user needs the following privileges:

- CREATE, ALTER, DROP Tables, Views, Indexes, Triggers (to update the Axon.ivy Engine)
- INSERT, SELECT, UPDATE, DELETE data

The best practice configurations are documented in chapter System Database.

Convert an old System Database



Warning

We strongly recommend to backup your database before you convert it to a newer version. Be sure that you have enough disk/table space on your database server. Most conversions add new fields to existing database tables which will enlarge the used database space.

If the system database has an older version, use the convert button at the bottom to convert it to the latest version.



Warning

Depending on the conversion steps and your database system it may be necessary to cut all connections to the system database to avoid problems. If you have problems with the conversion, please disconnect all other database management tools, clients or other tools that has a connection to the system database and try again.

System Administrators

On the *Administrators* tab you can configure users that have the right to administrate the Axon.ivy Engine:

AXON ivy digitalize your business

Engine Configuration

[Help](#)

- > Licence
- > System Database
- ▼ Administrators
- > Web Server
- > Cluster Nodes
- > Summary

Name	Fullname	Email	
admin	AXONIVY	support@axonivy.com	

[Add Administrator](#) [Next](#)

[Save](#) [Discard Changes](#)

Figure 9.3. Axon.ivy Engine Configuration Administrator Tab

Defining an email address for the administrators is recommended. Notifications of critical events like licence limits reached are sent to these email addresses.



Warning

This tab is only enabled if you have configured a connection to a valid system database.

Web Server Ports

On the *Web Server* tab you can configure which protocols the internal web server of Axon.ivy Engine should support and on which IP ports the web server is listening:

Figure 9.4. Axon.ivy Engine Configuration WebServer Tab

The following protocols are supported:

Protocol	Description
HTTP	HTTP protocol .
HTTPS	HTTP protocol over secure socket layer (SSL).
AJP	Apache Jakarta Protocol. This protocol is used for the communication of the embedded Servlet Engine with external WebServers like IIS or Apache.

Table 9.10. Web Server Protocols



Warning

This tab is only enabled if you have configured a connection to a valid system database.



Note

In case you disable HTTP port, then the specified port will still be opened by the engine for internal purposes. Even though the engine will refuse connections from remote hosts.

Cluster



Note

This tab is only visible if you have installed an Axon.ivy Enterprise Edition licence.

On the *Cluster* tab you have to configure some information according to the local cluster node:

The screenshot shows the 'Engine Configuration' interface for Axon.ivy. The 'Cluster Nodes' section is expanded, showing a table with the following data:

Host	Local Identifier	IP Port	IP Address
ZUGPCDHU2017.soreco.wa	0	6800	10.0.75.1

Below the table, there is an 'Add local Node' button and a 'Next' button. At the bottom of the interface, there are 'Save' and 'Discard Changes' buttons.

Figure 9.5. Axon.ivy Engine Configuration Cluster Tab

Use the *Add local node* button to add this installation as a new Engine cluster node to the list of cluster nodes in your Axon.ivy Engine Enterprise Edition. You have to configure an *IP Address* and an *IP Port* that will be used by the cluster to communicate with this node.



Note

An Engine cluster node is uniquely identify by the host it is running on and a local identifier. The local identifier is a unique number that identifies nodes running on the same host (machine). Both values are provided by the installed licence. Therefore, every Engine cluster node needs its own licence file.

Admin UI

The Admin UI is the application to manage and deploy applications. Manage its environments and global properties of the Axon.ivy engine.



Legacy Rich Dialog Application

The Admin UI featureset is historic and partly broken:

- System Properties, Application Properties, Security Systems, Active Environments : can still be viewed, but are overwritten by the values defined in config files such as the “ivy.yaml”
- Some features are still only accessible trough the Admin UI: like changing the activity state of Apps, PMs and PMVs.

Since 7.2 users can deploy applications by dropping the project artifacts in the deploy directory. And its application settings can be managed with the “app.yaml” file.

Opening the administration tool

After you have successfully started the Axon.ivy Engine, you can start the engine administration tool. This tool allows you to create and manage all your applications on the engine and in addition to set some further configuration properties.

To do so, launch your preferred web browser and point it at the following the address: `http://ServerName:Port/ivy`.

You should be directed to the Axon.ivy Engine info page:

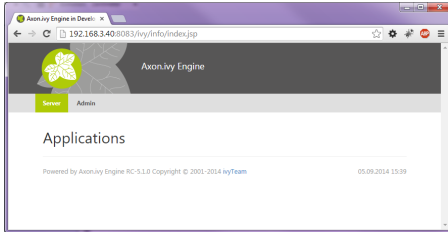


Figure 9.6. Axon.ivy Engine info page

This page displays all your applications and their process models with all contained process starts. When you haven't set up any applications yet, this page will be empty, just like in the screenshot above.

Click the **Admin** link to open the administration tool. Depending on your computer's settings, you may be asked if you'd like to open or save a JNLP file. Choose to open the file directly and wait for the Java Web Start application to load.



Note

Loading of the administration tool may take a few minutes the first time, because some application data needs to be deployed first internally. Please be patient.

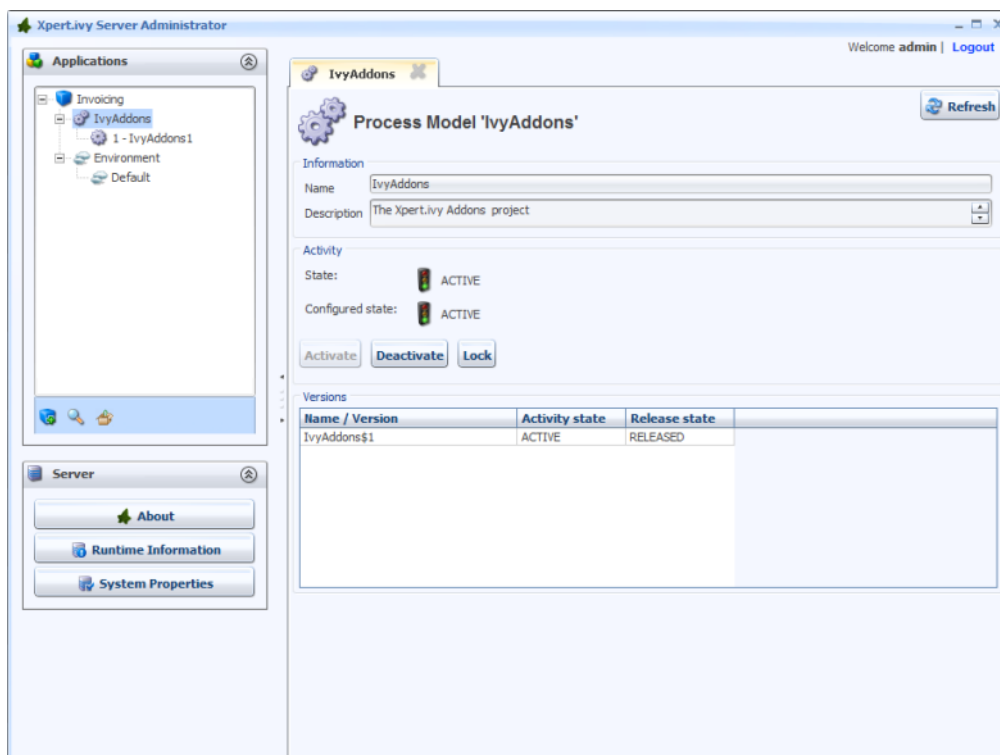


Figure 9.7. Admin Tool

Applications


Applications are environments for process models (projects). Applications are strictly separated from each other and do not share anything. Every application has its own process models (and -versions), roles, users, external databases and so on. Projects may only use libraries that are deployed within the same application.

Create new application



Create by file

Since 7.2 new applications can be created simply by dropping its files into a the deploy directory. There is no need to pre-configure and create an application with the Admin UI. See “Deployment”

To create a new application, press the new application icon () in the Applications section. A dialog will prompt you to enter the following information:

Name	The name of the application to create.
Description	An optional description of the application.
File Directory	The directory on the file system where all the files of this application and its process models will be stored. You can specify an absolute directory (e.g. C:/Data/IvyApps/App1 or /var/ivy/apps/app1) or a directory relative to the engine's installation directory (e.g. apps/App1). The directory may already exist, but it must be empty.



Tip

We recommend to configure an absolute directory outside of the installation directory of the engine because the data in the directory has a much longer life time than the current version of the engine. E.g. if you upgrade your engine you have to choose a new installation directory and may want to delete the installation directory of the old version, which is not possible if the file directory is located inside the installation directory.



Warning

When using an Axon.ivy Engine Enterprise Edition, the file directory must be located on a shared file system which is accessible by all nodes of the cluster with the same file path. This path must be configured here.

Owner	The owner of the application. This field is used for documentation purposes only.
Security System	The security system from which the users are imported. By default this is Axon.ivy which means that all users and roles are completely managed by the Axon.ivy Engine. You can change this to Microsoft Active Directory or Novell eDirectory in which case the users come from the selected directory service. See section Configuring an External Security System for more details about the integration of external security systems.
Create process models	Check the process models, that should automatically be created and deployed into the new application.

Axon.ivy Addons: Contains some Rich Dialog functionality such as some common dialogs simple questions or messages, document factories to create Office documents and many more. All these features can be used freely. Please read more about the Axon.ivy Addons in the documentation of the Axon.ivy Designer.

Axon.ivy Workflow UI (RIA and HTML Version): Both process models contains a default Workflow UI, which provide process and task lists. Furthermore it is possible to browse through history of already finished tasks. Even more, the RIA version provides functionality for administrating all cases and tasks of the application. More information can be found in the Workflow UI guide.

Click **Create** to create the application. If you have chosen a security system other than **Axon.ivy**, a dialog will appear to configure it (see next section).



Warning

There is a default **Portal** application which should only be used for demo purpose. Create always a new application for the production environment, because the shipped **Portal** application has a special lifecycle and no real migration path.

Configuring an External Security System



Configured by files

Since 7.2 the external security system might be configured via “ivy.yaml” and activated in the “app.yaml”. In this case the security system can still be viewed in the Admin UI, but changes have no effect on the actual config.

When you choose an external security system (e.g. *Microsoft Active Directory*) for a new application, you have to configure the connection to the directory server, to specify which users should be synchronized and how the attributes of these users should be mapped.

Users are synchronized as follows:

- Users that exists in the external security system but not in the Axon.ivy Engine are imported to the Axon.ivy Engine if they match the filter criteria.
- Users that exists in the Axon.ivy Engine but not in the external security system or do not match the filter criteria are deleted from the Axon.ivy Engine.
- Axon.ivy user attributes that are mapped to external security system attributes are updated with the latest values from the external security system.

The user synchronisation is executed:

- Once a day (see synchronization time).
- If one click the **Synchronize** button on the *Security System* section of the application configuration screen.
- If a user tries to login himself. This means that a user can login even if he is not yet imported to the Axon.ivy Engine.

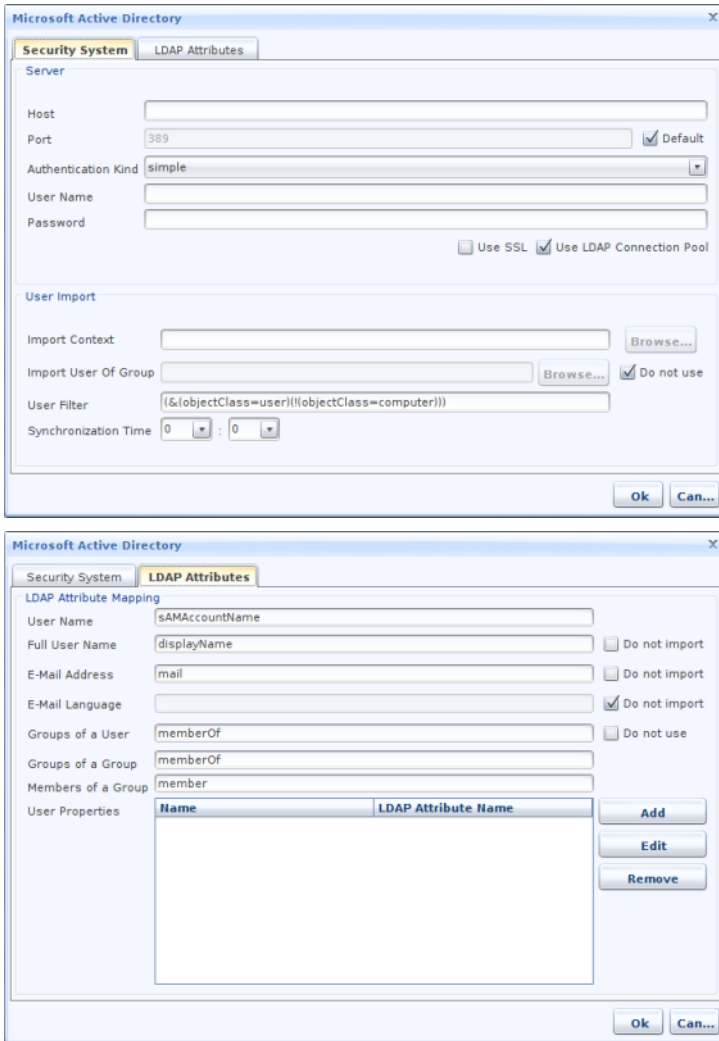


Figure 9.8. External security system configuration dialog

Host / Port	The hostname (or IP address) and port of the directory server
Authentication kind	The authentication method. Use none if your server does not require authentication to connect or simple if authentication by username and password is required
Username / Password	The username and corresponding password to use for connecting to the server. The format of the username can be one of the following (examples): <ul style="list-style-type: none"> <i>connectionuser@fully.qualified.domain.name</i> <i>cn=Connection User, ou=Service Accounts, dc=fully, dc=qualified, domain, dc=name</i>
Use SSL	Activates Secure Sockets Layer (SSL) connection to the Active Directory server.

The SSL connection to the Active Directory Server will only work if the JRE of the Axon.ivy Engine has the certificate of the Active Directory Server in its trust store. The certificate import into the JREs system trust store can be done with a graphical keytool like the KeyStore Explorer or the JRE keytool:

```
bin/keytool.exe -importcert
-file yourCertificateFile
-keystore lib/security/cacerts
```

```
-storepass changeit
-alias myCertificateAlias
```

Ensure that the System trust store is activated within the “System Properties” table. The System Property 'SSL.Client.UseSystemTrustStore' must be set to 'true'.

Use LDAP connection pool	If selected, the server will try to reuse connections and not open a new connection each time.
Import context	<p>The import context is a global filter. This filter is applied to every query made to the external security system. Every user and user group located below this context can be seen. Everything outside the context cannot be seen and will be ignored. In most cases the import context is set to a root object (e.g. for the domain soreco.ch to dc=soreco, dc=ch).</p> <p>In some cases you may want to set the import context to a organization unit so that only the users located below the organization unit are synchronized.</p> <p>Use the Browse button to select the import context from the server specified above (if the connection fails, check the connection data).</p>
Import users of group	<p>Limit the synchronization of users to a specific user group. Only users that are located below the import context and are members of the specified user group are synchronized.</p> <p>Use the Browse button to select the user group from the server specified above (if the connection fails, check the connection data).</p>
User Filter	LDAP filter expression to make sure only user objects are synchronized.
Synchronization time	Enter the time of day when synchronization should take place. Format is HH:MM (24h format).
LDAP attribute mapping	<p>Default attribute mappings can be adjusted.</p> <p>Further LDAP attributes can be mapped to additional user properties. Enter the LDAP attribute name in the first column and the name of the additional user attribute in Axon.ivy into the second column.</p>

The external security system configuration can be changed by using the **Edit** button on *Security System* section of the application configuration screen.



Warning

If you change the security system configuration then some users that were imported before may now no longer match the import filter criteria. This means that those users are deleted from the system and all their tasks will be changed to state UNASSIGNED. All tasks in state UNASSIGNED have to be reassigned manually to other users.

Application Default Settings



Configured by files

Since 7.2 the application settings might be configured via “app.yaml”. In this case they still can be viewed in the Admin UI, but changes have no effect on the actual config.

The button **configure default settings** on the application configuration screen leads to the *Application Default Settings*.

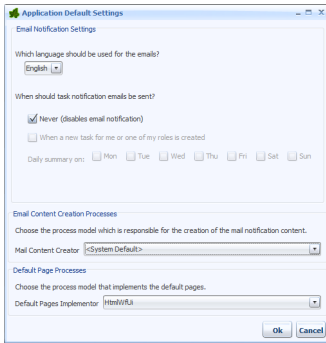


Figure 9.9. Application Default Settings

Email Notification Settings

This section contains the default settings for notification emails. The settings are used when a new user is created.

Email Notification Settings

It is possible to be notified if a new task is created that is related to you (either by direct assignment or by assignment to a role you own). Furthermore, a daily digest mail with a summary of all open tasks can be sent to you by Axon.ivy.

Which language should be used for the emails	Choose in which language you would like to receive the emails. If your preferred language is not contained, please contact your Axon.ivy administrator.
Never (disable email notification)	You can switch off the sending of all notification mails by ticking this checkbox. If you do so, you cannot set the other options.
When a new task for me or one of my roles is created	If this is set, you will receive a notification email whenever a new task is created that is assigned either directly to you or to one of the roles you own.
Daily Summary on	Choose the days on which you want to receive an email with a summary of all your open tasks.



Tip

If you want that temporary no mails are sent (e.g. for holidays), then just tick the **Never** checkbox. The email sending is now switched off, but the previous settings are still stored. As soon as you untick the **Never** checkbox, your standard configuration is back again.

Email Content Creation Processes

In this area of the *Application Default User Settings* dialog, you can select a deployed process model that contains notification processes for *Daily Summary* or for *New Task* notification respectively. Before you see any available processes in the combo box you have to **deploy** at least one PMV library including an Email Notification Process.



Tip

Choose *<System Default>* to use the built-in notification mails of the Axon.ivy Engine.

Default Executed Processes

This section allows you to override the default pages (Application Home, Task List, Process Start List, Login and End) of an Axon.ivy web application.

You can create your own process model to display these pages or you can use the Axon.ivy Workflow UI (HTML Version). See section create new application to learn how to use the "Axon.ivy Workflow UI (Html Version)".

Default Pages Implementor

<System Default> will show the default Axon.ivy pages.

HtmlWfUi will show up, if you have deployed the "Axon.ivy Workflow UI (Html Version)".

Any process model deployed, which contains a request start with a signature for at least one page will be shown in the dropdown. Instead of showing the default pages, the corresponding process of the selected process model is started. If the process start signature of the default page is not available in the selected project, the default page from Axon.ivy will be used.

Process Models and Process Model Versions

An application can have multiple process models. Each process model can have different versions. A process model corresponds to the concept of a project on the Axon.ivy Designer. A process model version corresponds to a versioned snapshot of that project at some point of time.

The Concept of Versions

A process model can exist in multiple versions called process model versions. These versions allow you to make changes in a project and deploy it again without having to worry about the compatibility of currently running cases. If the new version is not working as expected, you can always go back to a previous working version.

Creating Process Models and Process Model Versions

To create a new process model, you have to select the application in which you would like to create it. Press the **new process model** button in the toolbar (🛠️). You will be asked for a name (typically you chose the name to be the same as the name of the project you intend to deploy within this process model) and an optional description.

Once you have created the process model, you can continue with adding a first process model version. To do so, select the process model from the tree on the left hand side and press the **new process model version** button (🛠️). You will be prompted to enter a name and an optional description again.



Tip

It is recommended to use the description field of the new process model version to document the major changes from the previous version.

Manage Activation and Release State

Applications, process models and process model versions all have an activation state. The process model version additionally has a release state.

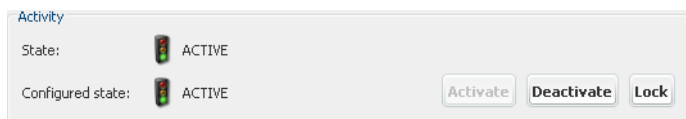


Figure 9.10. Managing the activation state

To bring a process model version into the **Active** state, the process model version itself, the parent process model and the parent application have to be in the state **Active**. Furthermore, the release state of the process model version has to be in **RELEASED** or **DEPRECATED**. The same is necessary for all required libraries of that process model version.

When looking at the details of an application, process model or process model version you will always see a **Configured state** and a **State**. The **Configured state** is the one you can freely change and the **State** is the actual state that depends on the states of the parent process model and application and of the release state.

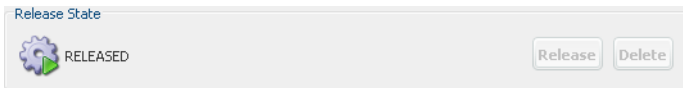


Figure 9.11. Managing the release state

Release States

Name	Description
PREPARED	The process model version has been created and a project may already be deployed. However, the process model version is not used yet.
RELEASED	The process model version is the currently released version. This means that all new cases will be started within this version. Only one version in a process model may be in this state.
DEPRECATED	The process model version has previously been in state RELEASED . But another process model version was released and activated recently. All cases that were started in this process model version will finish in this version. As soon as there are no more running cases in this version and all process model versions that are using it are not in state RELEASED nor DEPRECATED , the state will change to ARCHIVED automatically.
ARCHIVED	The process model version has previously been in state RELEASED or DEPRECATED . There are now no more cases that are running in this version, it is safe to delete it. Process model versions in this state are not started and therefore only use minimal resources. You may set it back again to RELEASED .
DELETED	The process model version has been deleted. All data that belong to this version are deleted too.

Table 9.11. Release states

Activation states

Name	Description
INACTIVE	No new process in this process model version can be started and all running cases and tasks have been suspended.
ACTIVE	New processes in this version can be started and all tasks are active.
LOCKED	No new process can be started, but the currently running tasks can continue to their next savepoint.
DEACTIVATING	The state is changing to INACTIVE .
ACTIVATING	The state is changing to ACTIVE .
LOCKING	The state is changing to LOCKED .

Table 9.12. Activation states

Project dependencies

Projects can depend on each other. The dependencies are configured in the deployment descriptor of the project. This dependencies are evaluated and resolved during the deployment of a project. On the process model version detail page you

can press **Dependencies...** to see the resolved dependencies of the project deployed to the process model version. You see the process model versions that the current process model version requires. And also vice versa which process model versions require the current process model version.

To change a resolved dependency select the dependency and press **Edit Selection**. You can then select another version that fulfills the dependency requirements.

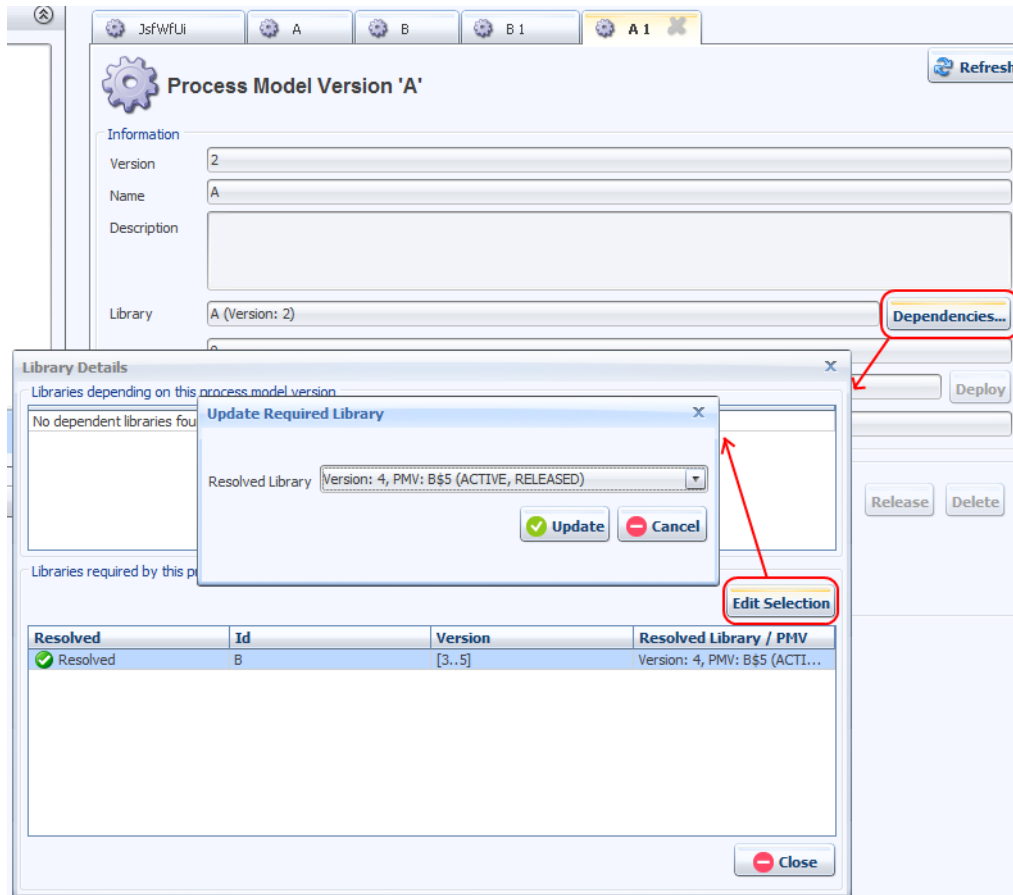


Figure 9.12. Project Dependencies

Deployment wizard



Create by file

Since 7.2 new applications can be deployed simply by dropping its files into a the deploy directory. There is no need to use the time consuming deployment wizard anymore. See “Deployment”.

The deployment wizard can be started on the toolbar.

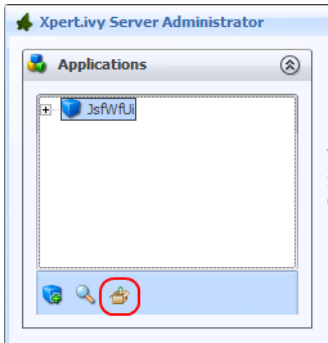


Figure 9.13. Command to Start the Deployment Wizard on the Toolbar

On the first page of the wizard select the projects that you want deploy. The projects can either be located on the server or on the client. If you select a folder then the wizard searches all packed (ivy archive) or unpacked projects located below that folders. If you select a *.zip file then the wizard searches all projects located inside the *.zip file. The found projects will be displayed. Select the projects you want to deployed and press **Next**.



Tip

Read the chapter *Deployment* in the Axon.ivy Designer Guide to find out how to create a zip file that contains all files of a project.



Tip

Read the chapter *Projects* in the Axon.ivy Designer Guide to find information about ivy archives and how to export them.

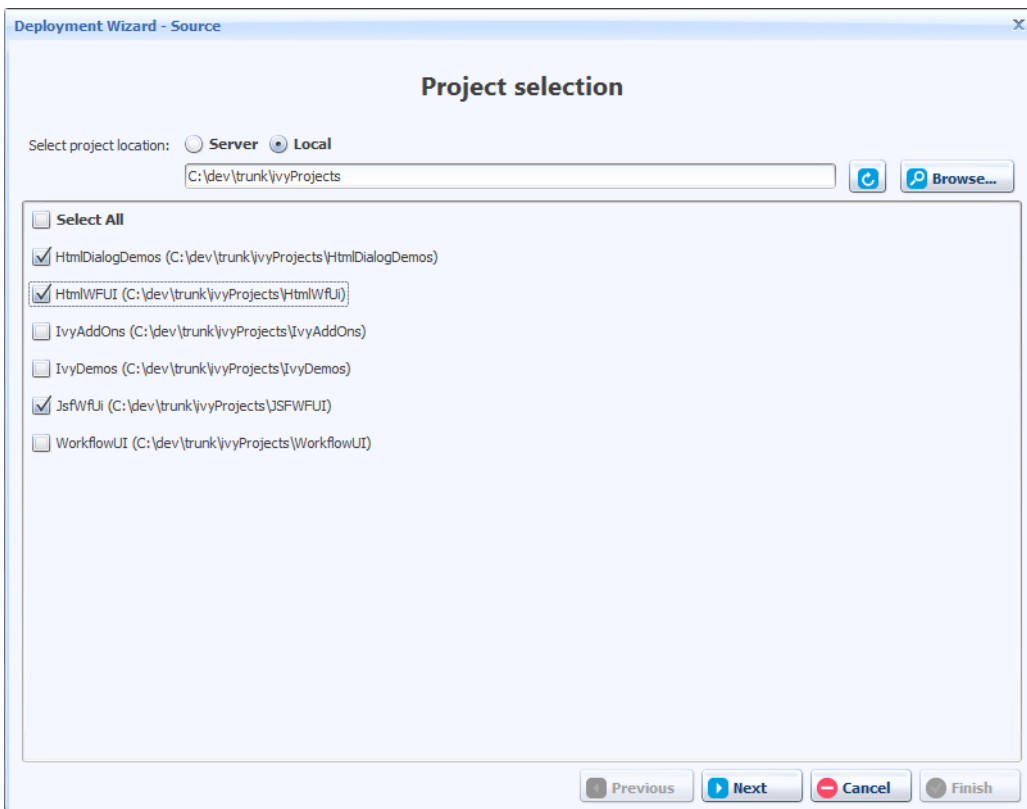


Figure 9.14. Deployment Wizard - Source

On the second page of the wizard select the applications to which you want to deploy your projects and press **Next**.

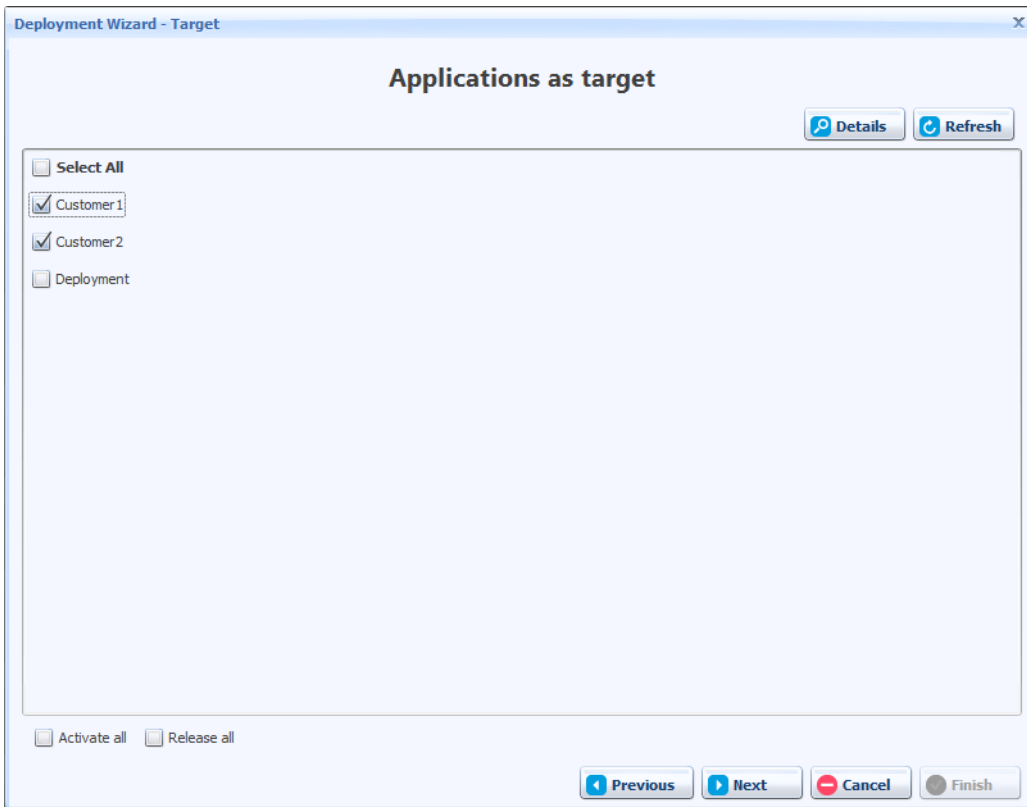


Figure 9.15. Deployment Wizard - Target

On the third page of the wizard you see a preview of the actions the wizard will perform. Note, that the wizard automatically choose the process model and process model version where it deploys a project to. If everything is as you expect press **Next**. If you do not like the automatically default behaviour press **Previous** and then on the previous page **Details** to choose another configuration. More information about process model and process model version can be found in the previous chapter.

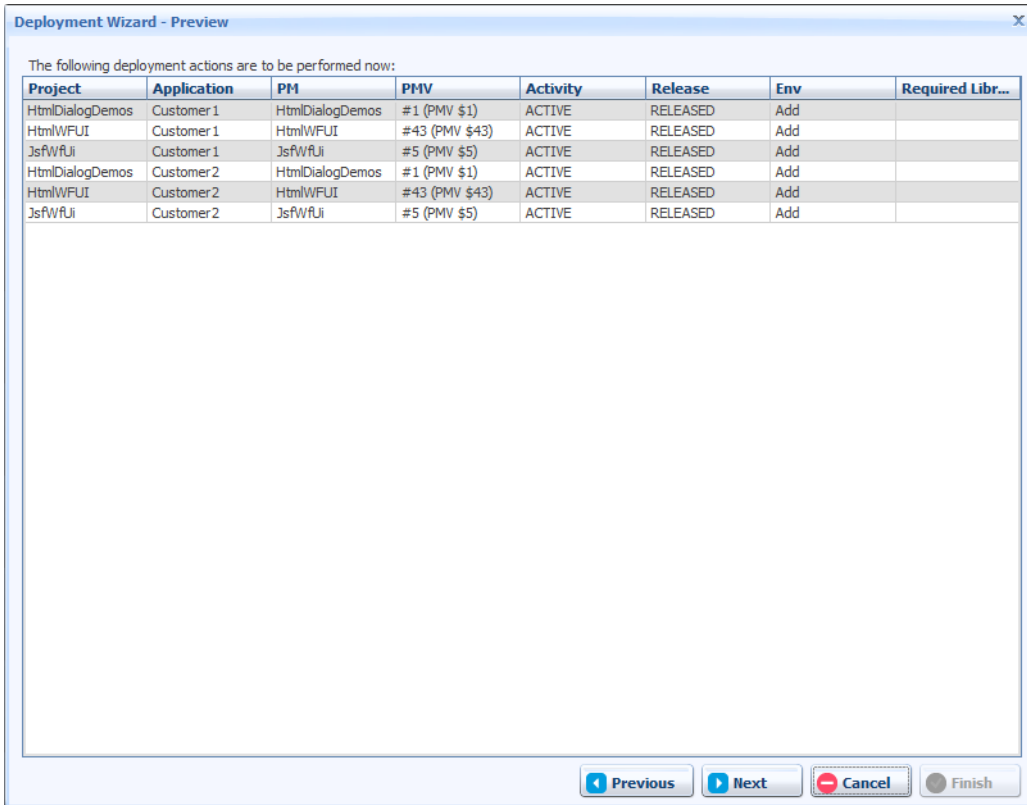


Figure 9.16. Deployment Wizard - Preview

On the fourth page of the wizard the projects are validated. If no errors are found you can press **Next**. If you get warnings please read them carefully and then decide if you want to proceed or not.

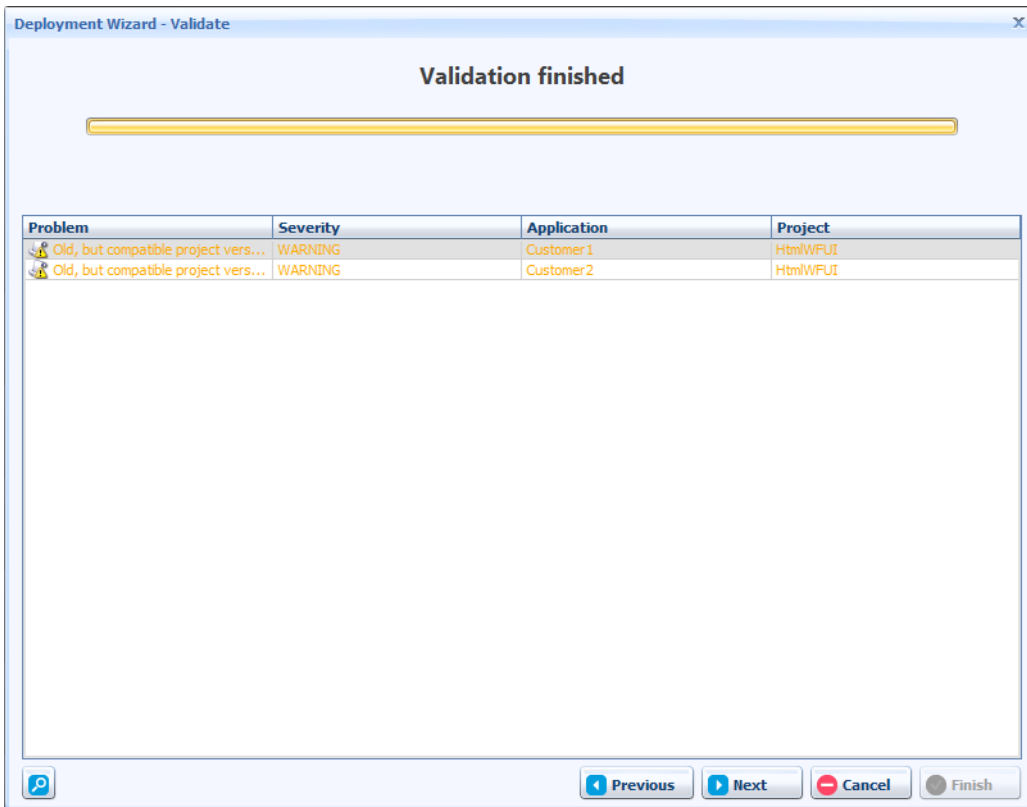


Figure 9.17. Deployment Wizard - Validate

On the last page of the wizard the projects are deployed. During the deployment a log is written which contains information about the tasks that are executed. Press **Finish** to close the wizard.

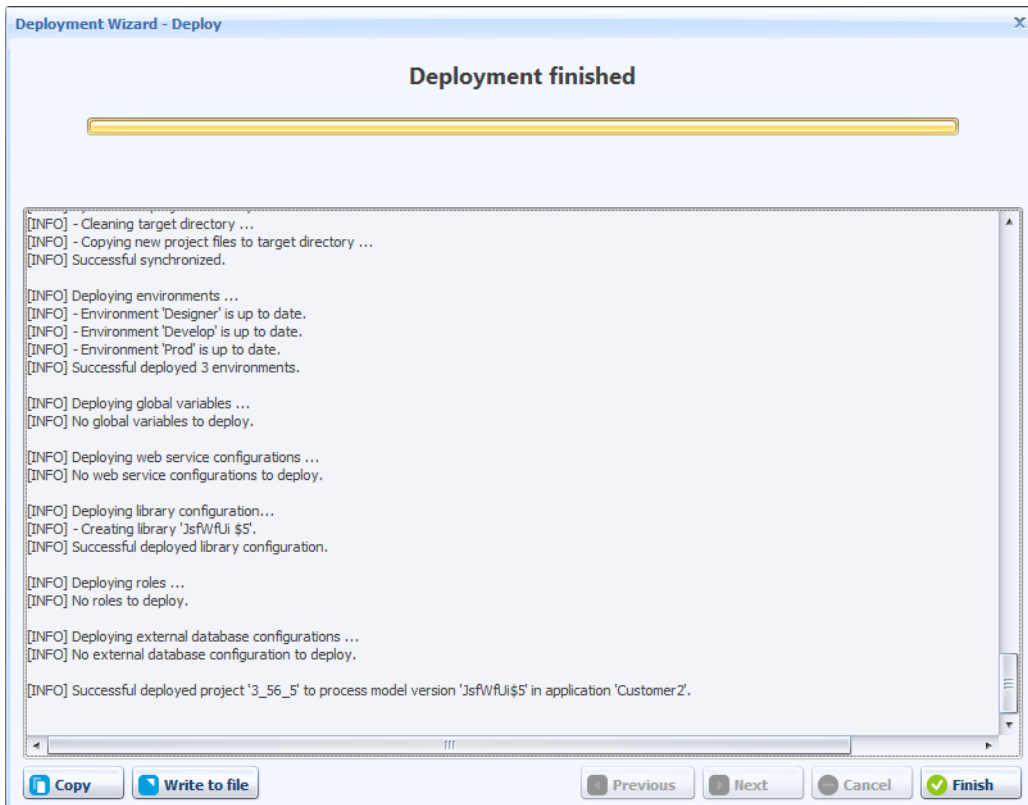


Figure 9.18. Deployment Wizard - Deploy

Business Calendar

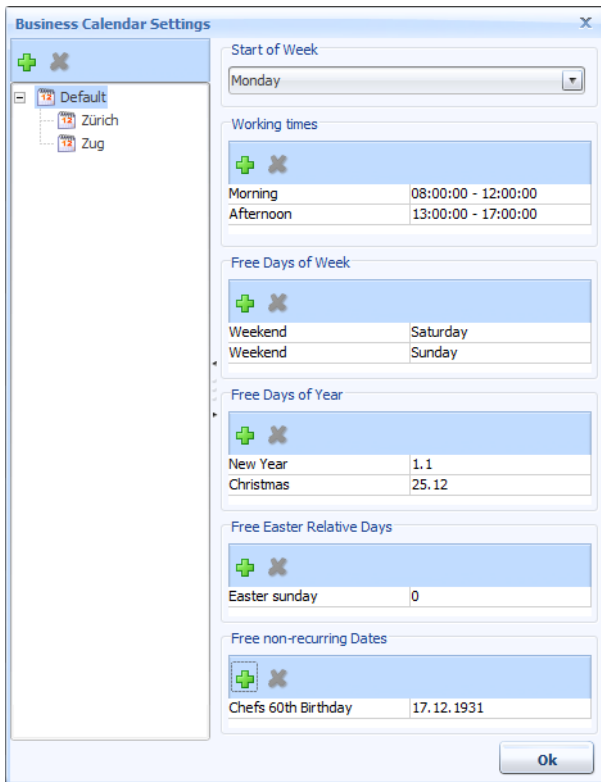
A business calendar defines the following three things:

- Days which are free of work. E.g. 25. December.
- Period in a Day which is working time. E.g. 08:00-17:00
- First Day of the Week. E.g. Monday

Business calendars are used to calculate time periods in working time. For example a period of 3 working days starting on Friday would end on Tuesday if Saturday and Sunday are defined as free days. For more information consult the documentation of [ch.ivyteam.ivy.application.calendar.IBusinessCalendar](#) in the Public API.

Every application has at least one business calendar. The business calendars can be configured using the **Configure Calendar Settings** Button on the Application Configuration screen.

Every environment can have its own default business calendar (see chapter Environment Business Calendar).



Business Calendar Definition

Business calendars are defined in a hierarchy. The children inherit all definitions of all its parents. All business calendars must directly or indirectly inherit from the Default business calendar.

Start of Week

The *Start of Week* defines the first day of a week. In Switzerland and its neighbour countries this is Monday. In other countries like Great Britain the week starts on Sunday.

Working Time

The *Working Time* defines which time of a day is working time. E.g. 08:00-12:00, 13:00-17:00

The working time applies to all days which are not defined as free days. It is not possible to define different Working Times for different days.

Free Days of Week

The *Free Days of Week* define at which days of the week no work is done. E.g. Saturday, Sunday

Free Days of Year

The *Free Days of Year* define at which days of the year no work is done. These free days of the year will be free every year at the same day. E.g. 1.1. (New Years Day), 25.12 (Christmas).

Free Easter Relative Days

The *Free Easter Relative Days* define days of the year no work is done. These free easter relative days will be free every year at another day depending on the easter day. E.g. 0 (Easter Day), 1 (Easter Monday), -2 (Good Friday).

Free non-recurring Dates

The *Free non-recurring Dates* define non-recurring days when no work is done. These free non-recurring days will only be free once. E.g. 7.12.1931.

Environments



Configured by files

Since 7.2 environments can be configured via “ivy.yaml” or app.yaml. In this case the environment configs can still be viewed in the Admin UI, but changes have no effect on the actual config.

This section briefly discusses the usage of environments in your projects. Developers should have the possibility to configure multiple environments (pointing to an infrastructure) and decide at runtime which environment should be used for the application. For instance you can have a development environment, a test environment and a productive environment. Here are some examples where environments can be used

- Companies provide different environments for their software products, like Development, Test and Productive. Each environment has its own infrastructure including databases, web services and other connections used by the project.
- Multi Client Capability. When the user logged into the system he can choose the mandant (e.g. Company 1, Company 2, etc.) and works with the data of the selected company. In the background the right databases connections, web services and other services for the selected environment will be used.

If your projects use environments, you have to configure the respective environment configurations on the engine. The first time you deploy an application that uses environments, the environments of the project will automatically be added on the engine as well. Each application manages his own environments. If you already define your different environments in your project on the designer, you don't need to reconfigure these environments on the engine again. This has the advantage that you can already test different environments at design time, before you deploy your projects on the engine. Each application contains a Default environment where all default values of the global variables and default database configurations are defined. In addition to the default environment, each application has one or more user defined environments.

In order to change the environment for an application at runtime you must go to the details of an application and change the active environment for that application and press **Set**

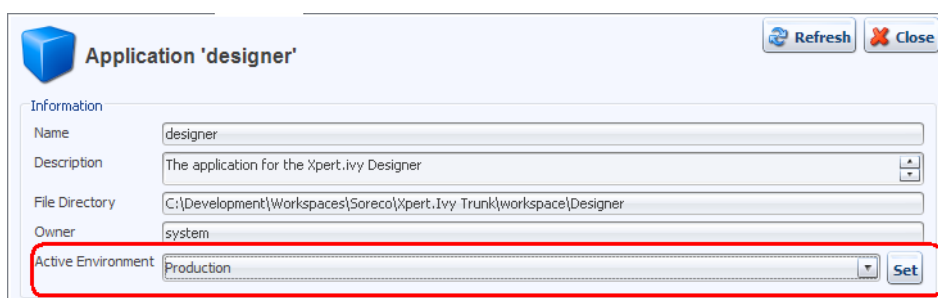


Figure 9.19. Set active environment for an application

Environments acts as a container for

- Global Variables
- External Database Configurations
- Web Service Configurations
- REST Client Configurations

- Business Calendar Configurations

Configuration of environments

Environment configurations can be changed by double-click on the environment entry in the list. You will see that the environment acts as a container for global variables and external database configurations. The description of the environment can be used to provide important infrastructure information about the server, databases and other stuff, which can be relevant to other users.

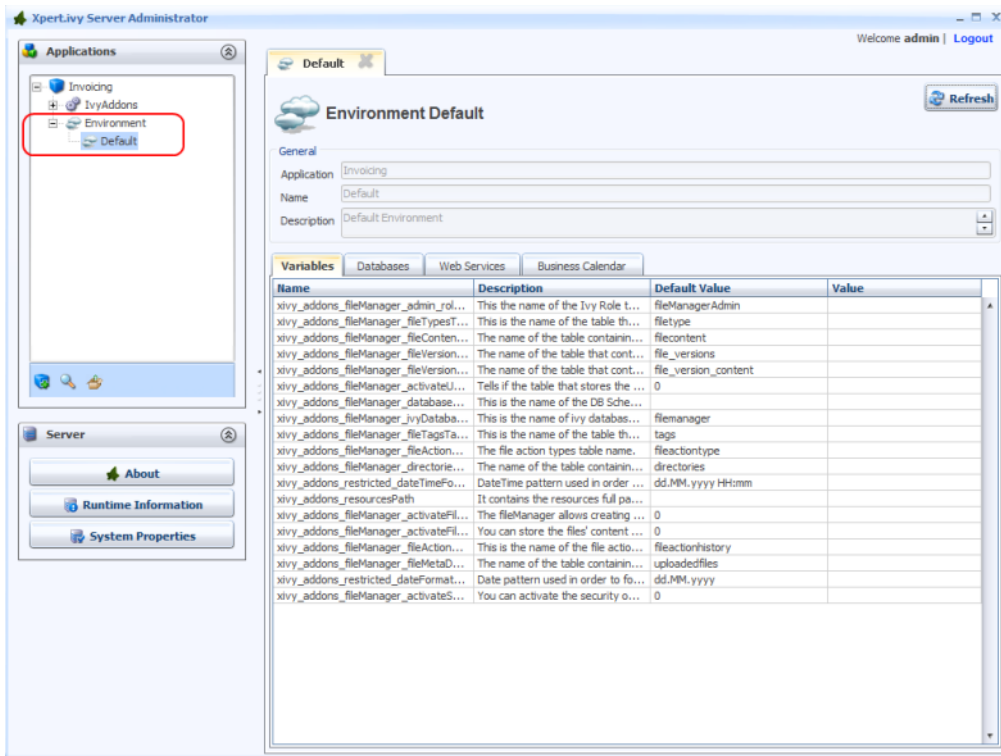


Figure 9.20. Environment details

Global Variables

Global Variables acts as global constants which can be used in your application. Global Variables are simple Key/Value pairs which can be specified by the developer. Some examples for global variables are:

- Company data (name, address, contacts)
- Simple Rule Values (e.g. credit account)
- Path values for saving files
- Path values for 3rd party systems and some other variables

Global variables must be defined at design time. **It is not possible to create new variables on the engine.** Each variable has a default value, configured at design time or in the default environment. In order to override the values of global variables for an environment just double-click the variable entry and override the values in the detail dialog. The default value of a global variable must be set in the Default environment.

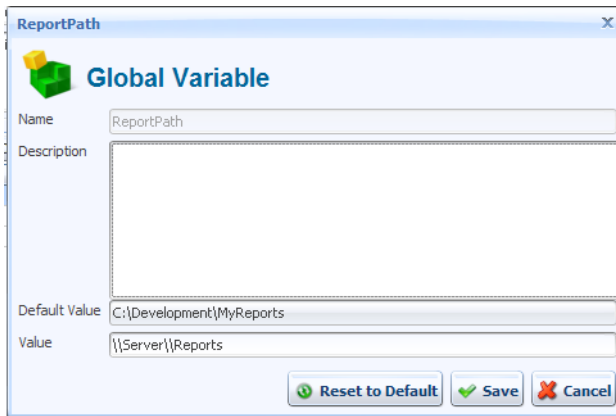


Figure 9.21. Details of a global variable

If the global variable is a default one, the system asks you to override the value for the environment. If you press "yes" the value of the global variable is overridden for the environment. You can always reset the global variable, by pressing **Reset to Default**.

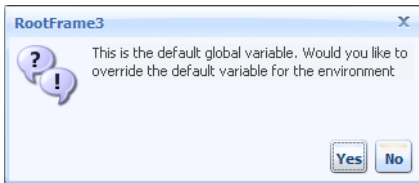




Figure 9.22. Question to override the default value of the global variable

External Databases

If your projects use external databases, you have to configure the respective database connections on the engine. The first time you deploy a project that uses external databases, the database configurations of the project will automatically be added on the engine as well. Also available environment configurations which already done at design time, will be added. Since you probably use a test database during development and you want to use your production database on the engine, you can use the different environments to set a different configurations. Overridden Database configurations for an environment will be displayed with the icon . Default database configurations or database configurations which are not overridden for the environment will be displayed with the icon .

To change the connection settings, select on the entry for a database in the list. You will see the connection URL used to connect to the database in the selected environment, the maximum number of connections, all currently open connections and the last executed statements.

If the selected environment is not the default environment, and you select a database configuration, which has no environment connection settings, you only can configure it. If you press **Configure**, the system asks you to override the database configuration for the selected environment. Press **Yes** to override the database configuration for the environment. You can always reset the database configuration to the default one, by pressing the button **Restore to Default** in the detail dialog, or right click on the database in the list and use the Menu item **Restore to Default**.

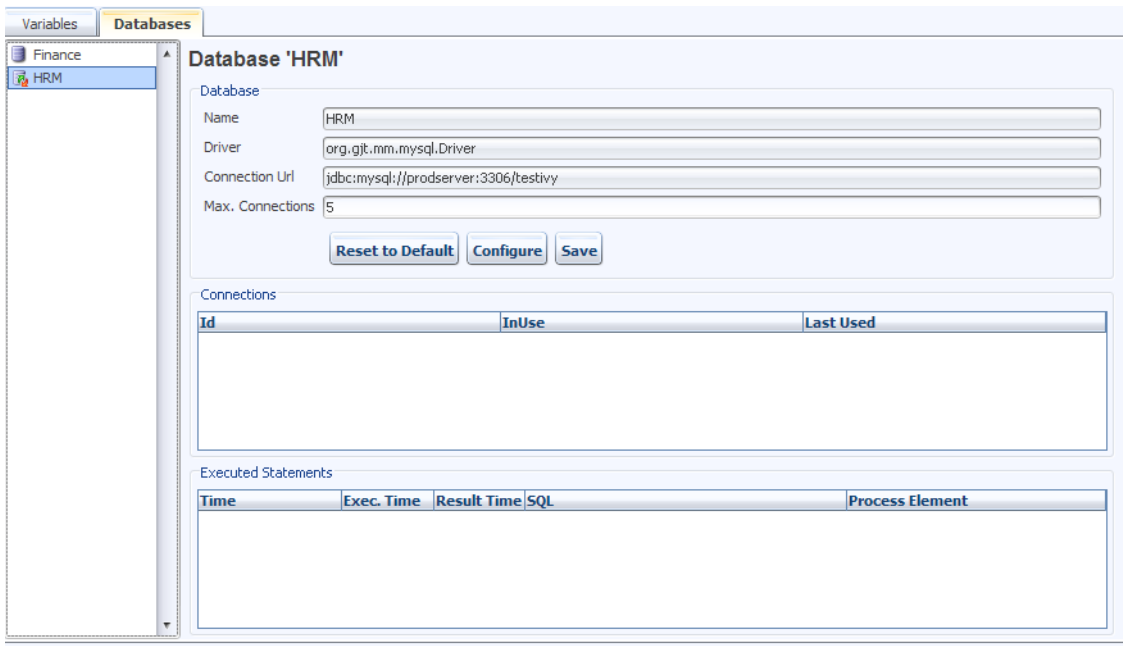


Figure 9.23. Databases

You can directly edit the maximum number of connections that may be simultaneously used.

To change the connection settings, press the **Configure** button next to the driver and connection URL fields. This will bring up a dialog in which you can configure the database product, driver, hostname, database, username, password and additional connection properties.

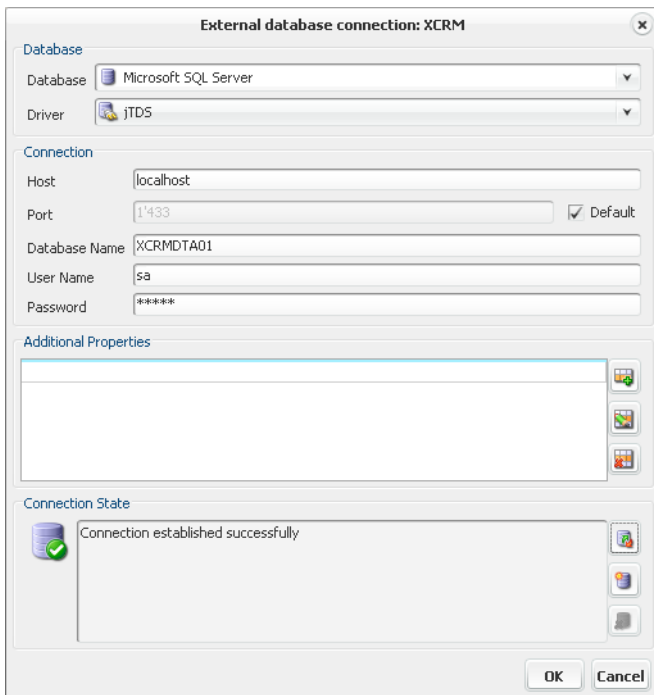


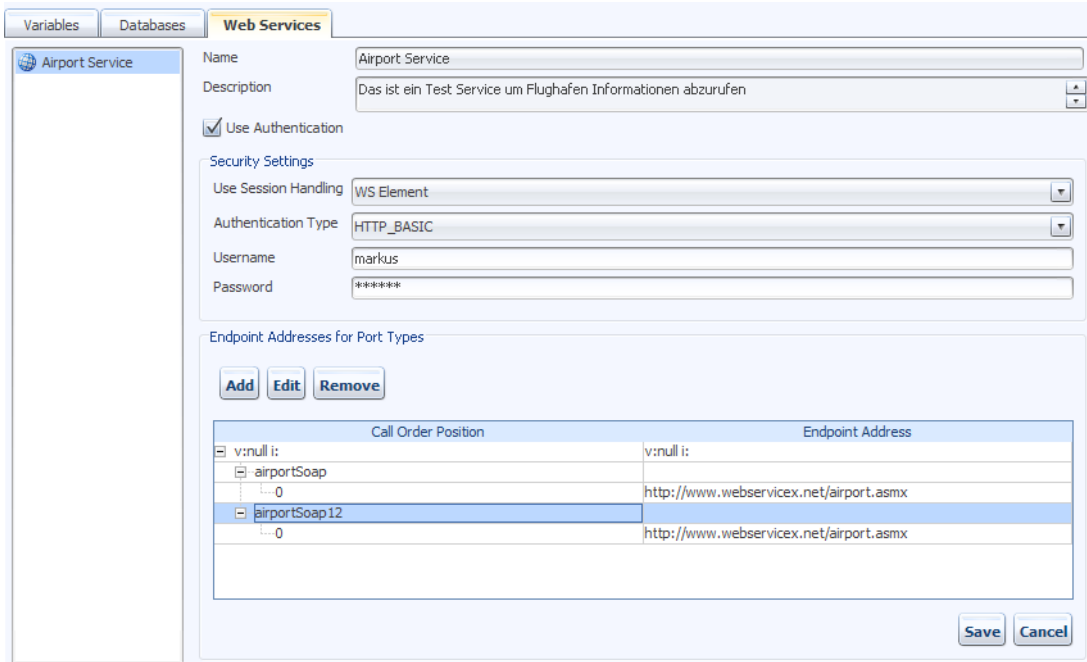


Figure 9.24. Configuring the connection of an external database

Web Services

If your projects use web services, you have to configure the respective Web Service on the engine. The first time you deploy a project that uses Web Services, the Web Service configurations of the project will automatically be added on the engine

as well. Also available environment configurations which already done at design time, will be added. Since you probably use a test environment during development and you want to use your production environment on the engine, you can use the different environments to set a different configurations. Overridden Web Service configurations for an environment will be displayed with the icon: . Default Web Service configurations or Web Service configurations which are not overridden for the environment will be displayed with the icon .



In order to edit a Web Service configuration just select the Web Service from the list and the details of the selected Web Service will be displayed in the detail panel.

- | | |
|-----------------------|---|
| Name | The name of the Web Service. This attribute can not be modified on the engine. |
| Description | An optional description of the Web Service. This attribute can not be modified on the engine |
| Use Authentication | The directory on the file system where all the files of this application and its process models will be stored. You can specify an absolute directory (e.g. C:/Data/IvyApps/App1 or /var/ivy/apps/app1) or a directory relative to the engine's installation directory (e.g. apps/App1). The directory may already exist, but it must be empty. |
| Session Handling Mode | You can chose session handling mode for current configuration. There are 3 modes available now: <ul style="list-style-type: none"> • NO there will be no session handling when you use this configuration • WSELEMENT invoking the same service from the same "WS step" process entry existing sessions will be reused • APPLICATION Within ivy applications whenever you invoke the same service existing session will be reused |
| Authentication Type | You can chose a authentication Mode for current configuration. There are 3 types available now: <ul style="list-style-type: none"> • HTTP BASIC Use the HTTP basic authentication for the Web Service call • HTTPS Use HTTPS transport |

- **DIGEST** Use DIGEST authentication

Username

When authentication is used, this username will be applied to get access to this web service. You can use IvyScripts in this field.



Tip

When you specify authentication in "WS step" process element, these settings (Username and Password) will be overridden. Use Scripts like "in.user" carefully, since you might use this WS entry in multiple processes with different data types.

Password

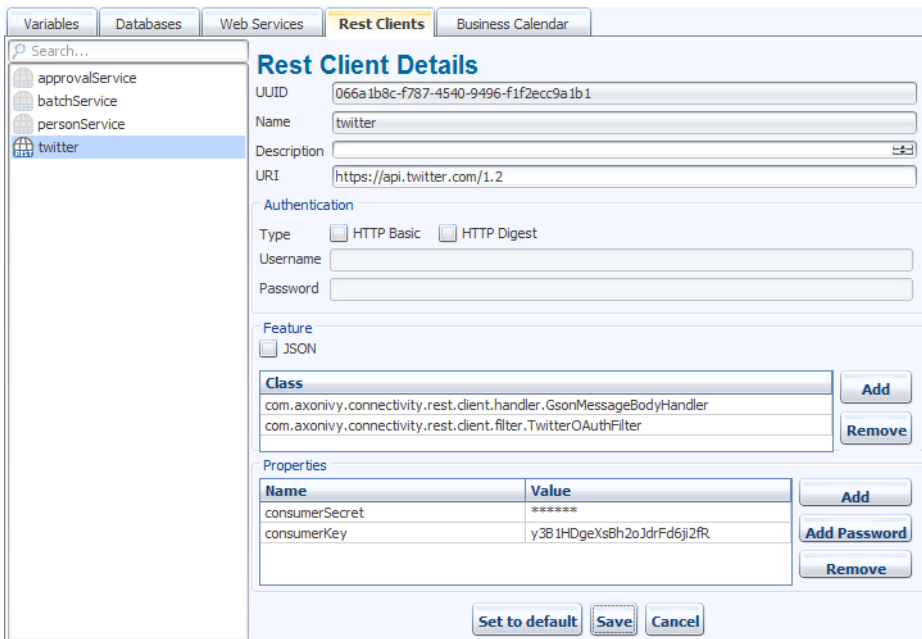
When authentication is used, the username above and this password will be applied to get access to web service. IvyScript is also interpreted in this field.

Endpoint Addresses for Port Types

The name of the Web Service. This attribute can not be modified on the engine.

REST Clients

If your projects use REST Clients, you have to configure the respective REST Clients on the engine. The first time you deploy a project that uses REST Clients, the REST Client configurations of the project will automatically be added on the engine as well. Also available environment configurations which already done at design time, will be added. Since you probably use a test environment during development and you want to use your production environment on the engine, you can use the different environments to set a different configurations. Overridden REST Client configurations for an environment will be displayed with the icon: . Default REST Client configurations or REST Client configurations which are not overridden for the environment will be displayed with the icon .



In order to edit a REST Client configuration just select the REST Client from the list and the details of the selected REST Client will be displayed in the detail panel.

UUID

Universal unique identifier of the REST Client. The REST Client can be referenced by this uuid. Cannot be modified.

Name

The name of the REST Client. The REST Client can be referenced by this name. Can be modified. Note that references using the name will break if you change it.

Description	Description of the REST Client.
URI	The base URI under which the remote service publishes its resources (e.g. <code>https://api.twitter.com/1.1</code>). The URI can contain template placeholders which are resolved later by the client user (e.g. <code>https://api.twitter.com/{version}</code>).

```
ivy.rest.client("twitter").resolveTemplate("version", "1.1").get()
```



Tip

To consume a REST service running in the same Axon.ivy Engine / Application as the client a set of Axon.ivy placeholders can be used. These placeholders are automatically resolved: `{ivy.engine.host}`, `{ivy.engine.http.port}`, `{ivy.engine.context}`, `{ivy.request.application}`.

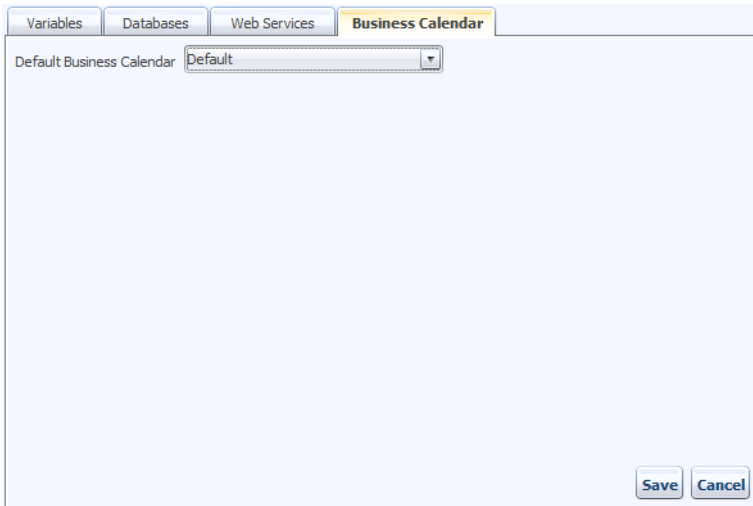
E.g. `http://{ivy.engine.host}:{ivy.engine.http.port}/{ivy.engine.context}/api/{ivy.request.application}/my/service`

Authentication	HTTP Basic	Adds support for HTTP Basic authentication.
	HTTP Digest	Adds support for HTTP Digest authentication.
	Username	The name of the user used to authenticate the client.
	Password	The password of the user used to authenticate the client.
Features	JSON	Adds a feature so that responses in JSON are mapped to Java Objects and Java Objects in requests are mapped to JSON.
	Features List	Shows the configured "features" classes. The classes configured here are registered in the <code>WebTarget</code> using the method <code>register(Class)</code> . The classes needs to implement a JAX-RS contract interface and must have a default constructor.
	Add	Adds a new feature class.
	Remove	Removes the selected feature.
Properties	Properties Table	Properties can customize the settings of the REST Client or one of its features. Well known properties of the client are documented here: <code>org.glassfish.jersey.client.ClientProperties</code> . The properties configured here are registered in the <code>WebTarget</code> using the method <code>property(String, Object)</code> .
	Add	Adds a new property.
	Add Password	Adds a new password property. The value of a password property is not visible in the table and is stored decrypted in the configuration file.
	Remove	Removes the selected property.

Business Calendar

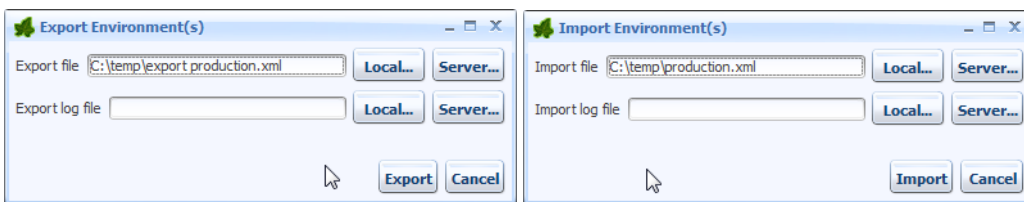
Each environment can define its own default business calendar which is going to be used to calculate time periods in working time if the environment is active. By default the global default business calendar is set.

Business calendars are defined on the Application (see chapter Application Business Calendar)



Export/Import Environment(s)

One or multiple environments can be exported into or be imported from a XML file. All configurations that belong to the selected environments will be exported or imported (see previous sub chapters). This can be used to transfer environment configurations from one Axon.ivy Engine installation to another. It is even possible to modify the exported XML before it is re-imported on the same or another engine.



The files can be chosen from the local computer that the user is working on or from the server.

Note, that the export/import log file does not have to be set. In any case, you will see an graphical log of the results during the export or import.



Note

The imported environment data has always precedence over the data in any existing environments. Exactly speaking, the following rules apply:

- If a configuration exists both in the import file and in the target environment, then the data for this configuration will be taken from the import file.
- If a configuration exists in the import file but in the target environment, then the configuration from the import file will be created in the target environment.
- If a configuration does not exist in the import file but exists in the target environment, then the configuration in the target environment will be deleted.
- If a configuration is defined in the import file but does not have a corresponding default configuration in the target application then nothing happens.
- If the import file contains an environment that does not exist in the target application, this environment is not imported.

Users and Roles

Roles and users are always configured per application. On the application details panel, you will find a group **Security System**. There the number of all existing users and roles are shown.

If you are using an external security system such as *Microsoft Active Directory* you can force a synchronization with the directory using the **Synchronize** button. To change the connection and import configuration, press the **Edit...** button.

Security System

Name: Xpert.ivy [Edit ...] [Synchronize]

Users: 2 [Show ...]

Roles: 1 [Show ...]

Figure 9.25. Overview of roles and users

User list

When clicking the **Show...** button next to number of users, you will be presented with a list of all existing users. You can create, edit and delete users. You can also set roles and permissions of any users individually.



Note

When using Microsoft Active Directory or Novell eDirectory as security system, you will not be able to create, edit or delete users. All these tasks need to be performed on the Active Directory or eDirectory directly and will then be synchronized with Axon.ivy.

Name	Full Name	E-Mail Address
testuser1	Test User 1	testuser1@ivyteam.com
testuser2	Test User 2	testuser2@ivyteam.com

[Create ...] [Edit ...] [Delete ...] [Roles ...] [Properties ...] [Permissions ...] [System Permissions ...] [Ok]

Figure 9.26. List of all users

Creating a New User

To create a new user press the **Create...** button. Enter the information for the new user (the username must be unique within the application). All other fields are optional.

Figure 9.27. Creating a new user

For the email notification settings you can decide whether the user uses the defaults defined by the application (they are shown in light gray if you choose so) or whether you want to define specific settings for the user. Note that each user can manipulate these settings in the **Workflow UI**.

Email Notification Settings

It is possible to be notified if a new task is created that is related to you (either by direct assignment or by assignment to a role you own). Furthermore, a daily digest mail with a summary of all open tasks can be sent to you by Axon.ivy.

Which language should be used for the emails	Choose in which language you would like to receive the emails. If your preferred language is not contained, please contact your Axon.ivy administrator.
Never (disable email notification)	You can switch off the sending of all notification mails by ticking this checkbox. If you do so, you cannot set the other options.
When a new task for me or one of my roles is created	If this is set, you will receive a notification email whenever a new task is created that is assigned either directly to you or to one of the roles you own.
Daily Summary on	Choose the days on which you want to receive an email with a summary of all your open tasks.



Tip

If you want that temporary no mails are sent (e.g. for holidays), then just tick the **Never** checkbox. The email sending is now switched off, but the previous settings are still stored. As soon as you untick the **Never** checkbox, your standard configuration is back again.

Editing an Existing User

You can change the details of an existing user by pressing the **Edit...** button. You can change all fields except the username. If the password field is left blank, it will not be changed. Otherwise the password of the user will be overwritten with the new value of the password field.

Deleting a User

To delete a user, press the **Delete...** button and confirm the deletion.



Warning

Deleting a user will change the state of all tasks, for which the user is currently responsible, to **UNASSIGNED!**

All those tasks must be reassigned to another user or role by a workflow administrator or they will never be finished.

Edit roles

Click the **Roles...** button to change the roles of a user.

You can add a role to a user by selecting the role from the list and then pressing **Add**. To remove a role, select it and press **Remove**.

Some roles may not be editable. This can have multiple reasons:

- The role may be *inherited* (indicated by a grey checkbox and the text Inherited). A role is inherited if it is not explicitly set, but the user owns a sub role (see example: Role 1 is inherited because user owns Role 1.1 and Role 1.3).
- You are using an external security system (e.g. ADS). In this case, you can not edit roles that are linked to a group on the directory server. To add such a role, add the user to the corresponding group on the directory server.

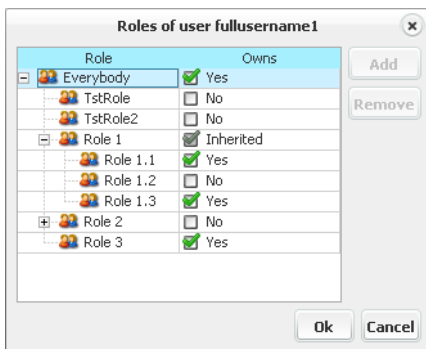


Figure 9.28. Manage roles of a user

Properties

In this dialog it is possible to manipulate the properties of all users. Properties are key/value pairs that can be accessed at runtime through IvyScript. If using the internal security system of Axon.ivy (see [Configuring an External Security System](#) for more information), then creation, editing and deletion are supported for all properties. If the users are synchronized with an external data source such as an **MS Active Directory** and a mapping between attributes from the corresponding user in the external system to the properties of the Axon.ivy user is configured, then editing and deletion of such properties is usually prohibited by the external security system and therefore not possible within Axon.ivy. Just use the interface to the external security system to manipulate the attributes directly there.

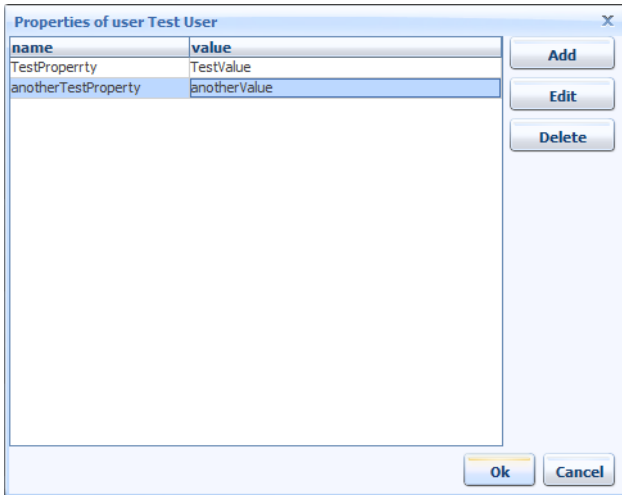


Figure 9.29. User properties

Just use the *Add* button to create new properties, the *Edit* button to manipulate the property value and the *Delete* button to remove the property again. Note, that editing of the value can be done directly in the table.

Edit Permissions / System Permissions

You can edit the permissions of a user by selecting the user and clicking **Permissions...** or **System Permissions...** respectively.

See section Permissions for more information.

Roles

Press the **Show...** button next to the number of roles on the Security System section of an application to see a list of all existing roles.

The roles are defined in the projects that you deploy in your application. *You can not add or delete roles on the engine!*

The list shows the roles with their name plus, if available, the external security name in square brackets.

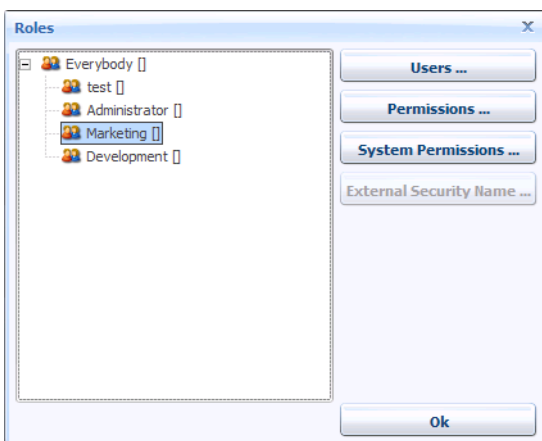


Figure 9.30. List of all roles

Users of a Role

To add or remove users of a role, click the **Users...** button. You should now see a list of users not owning the role on the left and a list of user that do own the role on the right. You can move users with the buttons in the middle from one list to another.



Note

When a user only inherits a role by owning a sub role thereof, the user will appear in the list of users *not owning* the role.

When a role is linked to a group in an external security system, you will not be able to edit the users of a role.



Figure 9.31. Users of a role

Edit Permissions / System permissions

You can edit the permissions of a role by selecting a role and clicking **Permissions...** or **System Permissions...** respectively.

See section Permissions for more information.

External Security Name

If you are using an external security system (e.g. Microsoft Active Directory) then you can link an Axon.ivy *role* to a *group* or another *structural node* (e.g. Organisation Unit) on the directory server. If a *group* is selected then all users that are members of this *group* will automatically receive the associated Axon.ivy *role*. If a *structural node* is selected then all users located below the *structural node* will automatically receive the associated Axon.ivy *role*.

Press **External security name** to edit or browse the name of the *group* or *structural node* whose users should receive the selected Axon.ivy *role*.

Permissions

Permission Kinds

There are two kinds of permissions:

System Permissions System permissions are valid system wide, e.g. on the whole engine.

Permissions Regular permissions are valid only within the application for which they are defined.

Assignment of Permissions

You can assign different permissions and system permissions to each user or role.

A permission can either be granted, denied or unspecified (not granted). The actual permissions of an user depend on the permissions set on the user itself and the permissions set on all roles that the user owns.

Grant permissions take precedence over Deny permissions, if set on the same level. On a user, inherited permissions can be overridden with an explicit Grant or Deny.



Warning

Inherited Deny permissions have no effect if the user has an explicit Grant permission or another role on which the permission is Granted. Explicit permissions always take precedence over inherited permissions.

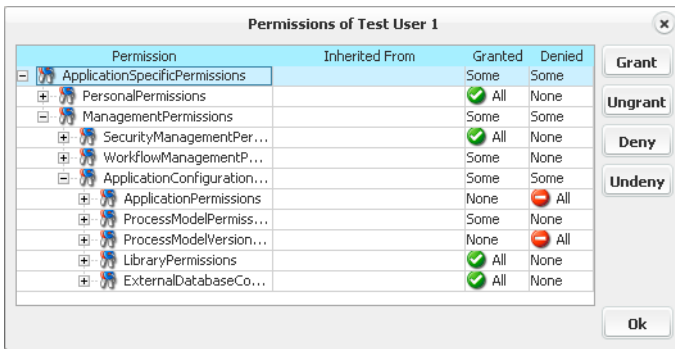


Figure 9.32. Editing the permissions of a user

System Properties



Configured by files

Since 7.2 the primary source of system properties are configurations set in files such as the “ivy.yaml”. If properties are defined in files, changes in the Admin UI have no effect. But the actual values can still be viewed here.

System properties are engine wide settings and are therefore valid for all applications. Be careful when changing those settings, since some particular combinations of settings may stop the engine from working properly.

The system properties can be accessed through the button **System Properties** in the **Engine** section of the left navigation bar.

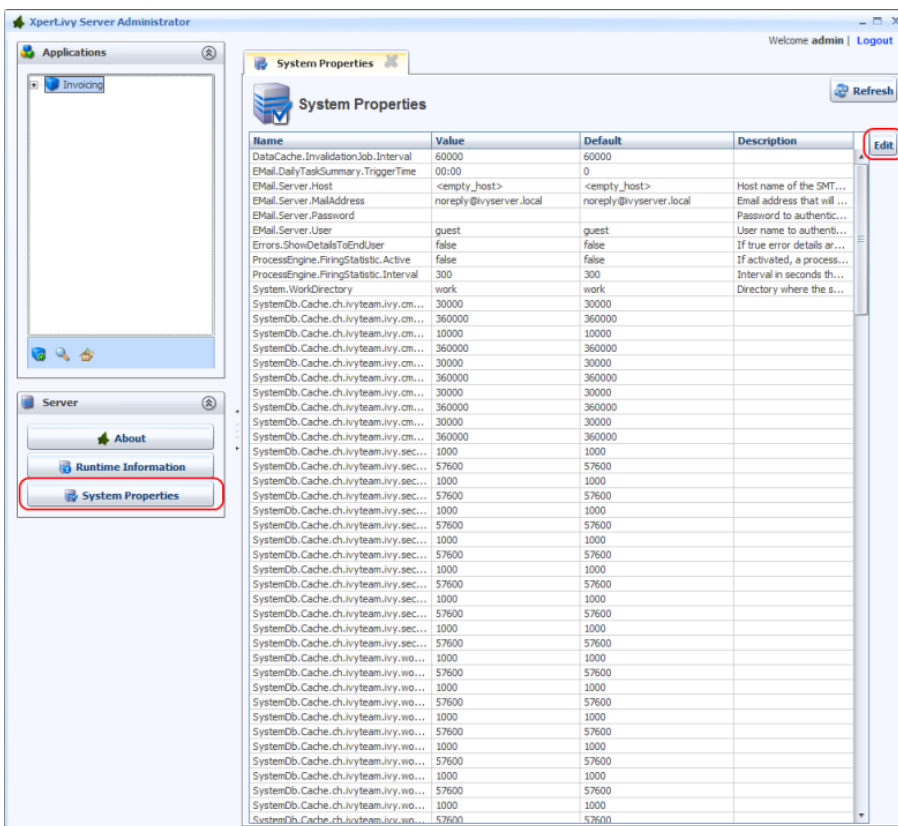


Figure 9.33. Overview of system properties

To change any properties, select the corresponding row in the table and press the **Edit** button or simply double-click on the row.

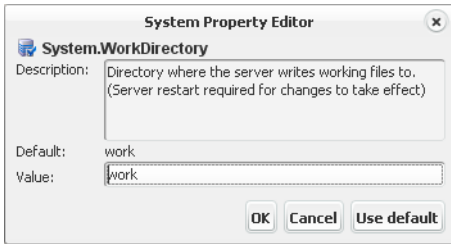


Figure 9.34. Editing a system property



Note

Some settings may not take effect until you restart the engine.

Engine infos

Runtime information

Pressing the button **Information** in the **Engine** section of the left navigation bar will provide you with runtime information about the engine. This includes memory usage and stack-traces for all running threads.

Class	Method	Line	File
java.lang.O...	wait	-2	Object.java
org.apache....	run	559	ThreadPool....
java.lang.T...	run	595	Thread.java

Figure 9.35. Runtime information

About

Pressing the button **About** in the **Engine** section of the left navigation bar will provide you with information about the version of Axon.ivy Engine and your operating system.

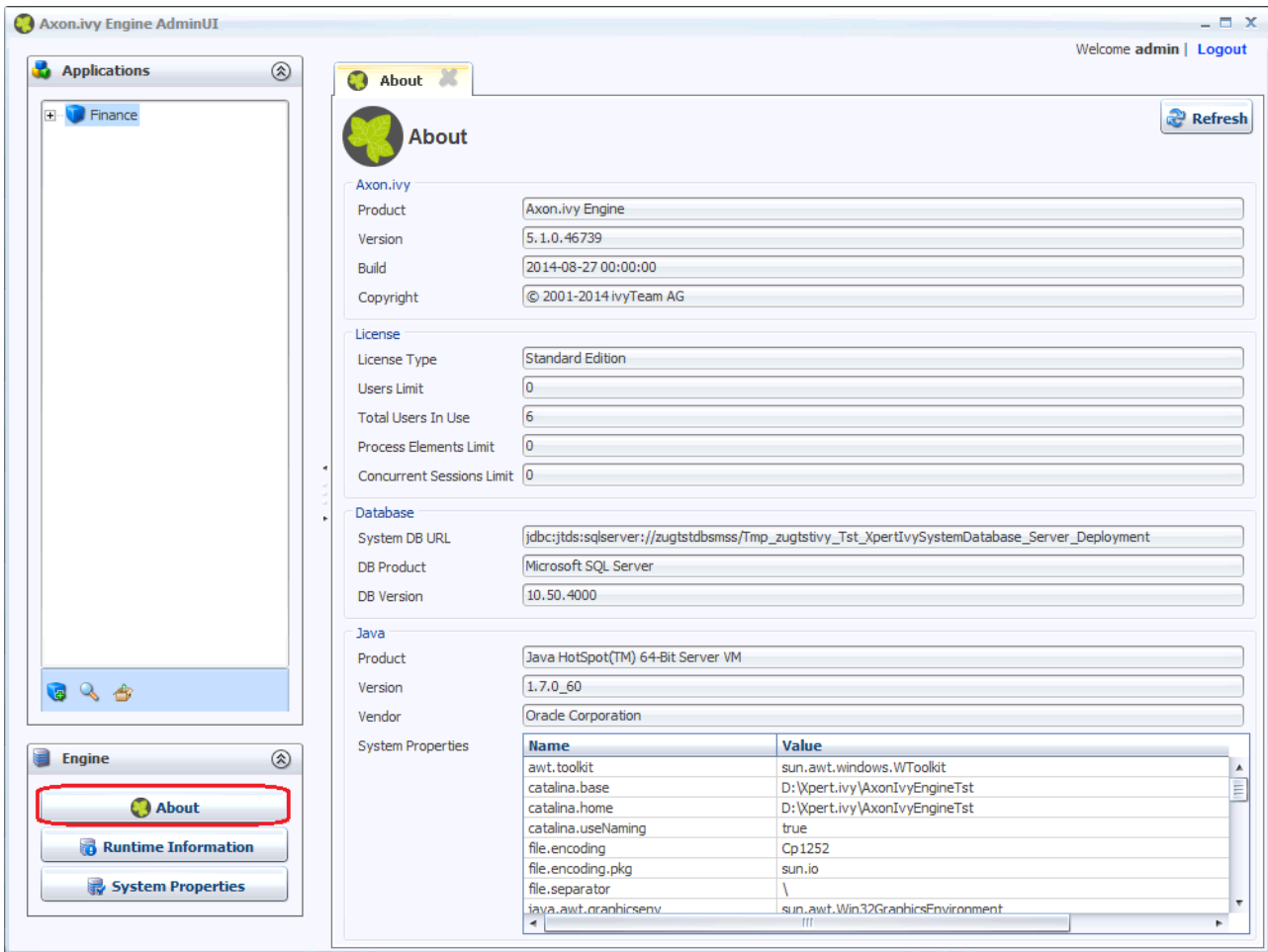


Figure 9.36. About

Control Center

The Control Center integrates all tools to configure the engine, the (Windows) service and to start/stop the installed Axon.ivy Engine.

To open the Control Center application, go to your Axon.ivy Engine installation directory and launch the *ControlCenter.exe* or the *ControlCenter* program located in the *bin* folder.

Launchers

The following program launchers are available for the Control Center program:

Platform	Console support	Launcher
Windows	no	bin/ControlCenter.exe
Windows	yes	bin/ControlCenterC.exe
Linux	yes	bin/ControlCenter

Table 9.13. ControlCenter Launchers

Start / Stop

To start the Axon.ivy Engine, simply choose the **Axon.ivy Engine** in the list on the left side and then press the green start button.

Alternatively, you can choose the **Axon.ivy Engine [Console]** from the list to start the engine within a console to which some information about the engine is logged. Please note that closing this console window will terminate the Axon.ivy Engine without shutting it down properly.

To stop the engine, click the red stop button.

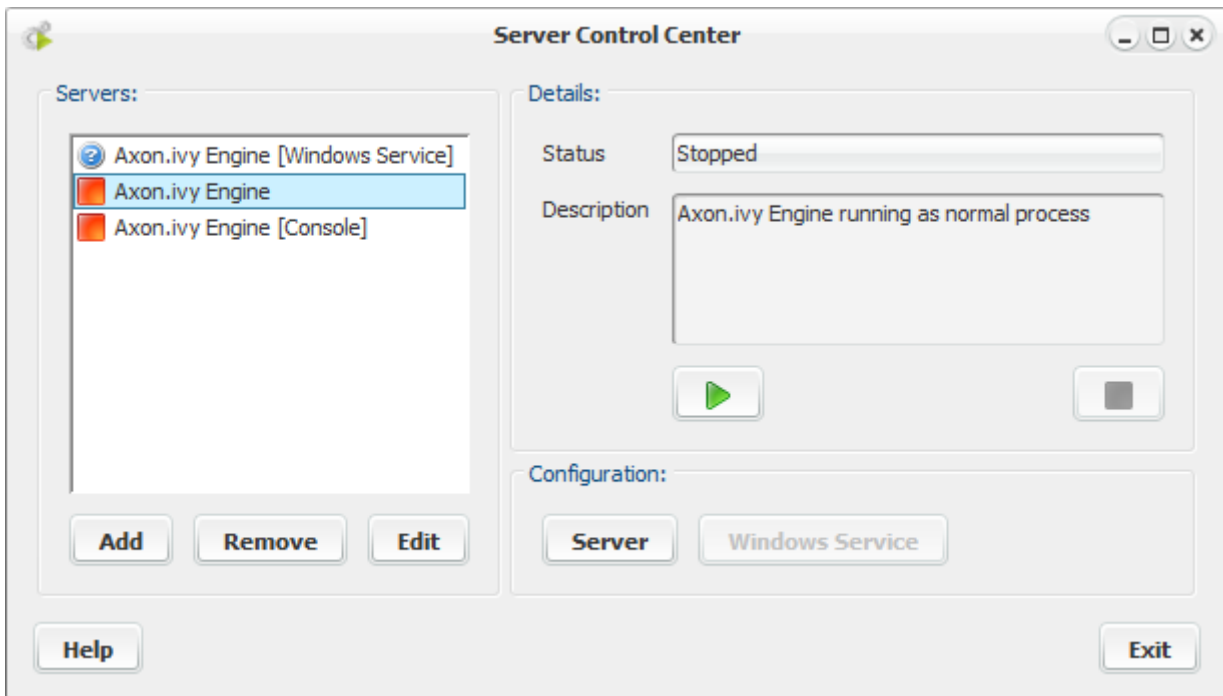


Figure 9.37. The Control Center

Configuring Windows Service (Windows only)

If you've installed the Axon.ivy Engine under a Windows operating system, you can register it as a Windows service. To do so, select the entry **Axon.ivy Engine [Windows Service]** from the list on the left and press the button **Windows Service** on the right. A dialog will open, prompting you for additional configuration data:



Figure 9.38. Configuring Axon.ivy Engine as Windows service

First of all press **Register Service** to register the service and to enable the rest of the configuration sections.



Tip

Service operations (register, unregister, start, stop) may fail because the current user does not own the necessary rights. In this case close the **Control Center** and start it again by right clicking on the **ControlCenter.exe** and choose the command **Run as administrator** from the context menu. After that, the service operation should work.

Now you may configure the user under which the service (and therefore the Axon.ivy Engine) will be executed. This can be either the system user or any other user with sufficient rights to start services and access the Axon.ivy Engine installation directory (read and write).

By default, the service start kind is **Manually**. To start the engine each time Windows is booted, choose the setting **Automatically**

The last thing that can be configured are the services that the Axon.ivy Engine depends on. This might be the database management system on which the system database is located or the web server in which Axon.ivy is integrated (IIS or Apache). All the services that you add in this list will be started before Axon.ivy and if any of these services fail to start, Axon.ivy won't start too.

After you have finished the configuration, click **Ok**. Now you will be able to start the engine from the control center or you may also use the Windows Service Management Console.

Testing the Engine

Once you've started the Axon.ivy Engine, try to open the following address in your preferred web browser: `http://ServerName:Port/ivy`. If a web page with the Axon.ivy logo appears, the installation and configuration of the Axon.ivy Engine was successful and you may continue with the next chapter.

Server List Configuration

The list with the engine types on the right may be extended by users. You may add other Axon.ivy Engine installations and you even can integrate other third party tools to start them from the Control Center.



Note

The indication whether the program behind an entry in the server list is running or not is only shown for the Axon.ivy Engine binaries of the installation the Control Center belongs to and for any Windows services (including the Axon.ivy Engine services). This applies too for the *show console* setting because only Axon.ivy Engine binaries can be started in a console (third party applications cannot).

Add opens a dialog to choose the type for the new entry. For integration of another Axon.ivy Engine binary or a third party tool, choose the first option (*ivyTeam based Server*), if you intend to integrate an Axon.ivy Engine as a Windows service or any other Windows service, then choose the second option (*Windows Service based server*).

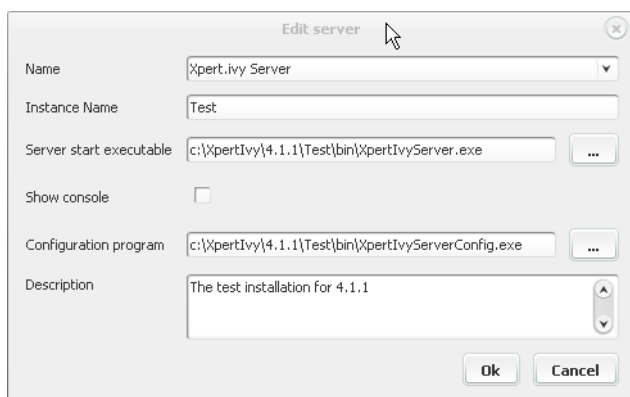


Figure 9.39. Create a new server in the server list

In the configuration dialog for a normal application you can set the base name and/or refine with the instance name (in the server list the instance name is printed in brackets after the name). Add the server binary (or your third party tool) in the *server start executable* and the configuration utility in the field *configuration program* (or the configuration program of your third party application). If and only if you choose the console based binaries (the ones with "C" at the end of the file name, e.g. AxonIvyEngineC.exe) then tick the check box *Show console*. It has no effect on all other binaries.

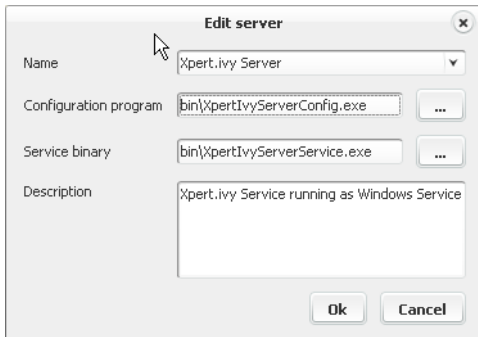


Figure 9.40. Create a new service in the server list

In the configuration dialog for adding/editing a service entry, you can choose an already existing service from the combo box or set the service name when you did not already register the service. Set the *configuration program* and the *service binary* similarly to the description above. For simply starting/stopping existing services from the Control Center, it is not necessary to define the *service binary*



Note

The name in this dialog must be exactly the same name that is used to register the service. Otherwise the lookup will not work.

Remove removes the selected entry from the list and *Edit* allows to edit the configuration for the selected entry in the server list.

Chapter 10. Troubleshooting

Troubleshooting

If you encounter any problems, check the following sources:

- | | |
|----------------|---|
| Axon.ivy Q&A | The Axon.ivy Q&A contains a considerable amount of questions and answers related to Axon.ivy Designer and Engine. |
| Stack Overflow | Problems related to common technologies like Java, JSF, Primefaces are answered on the web, e.g. on Stack Overflow. |
| Support | You can get support via ivy@axonivy.com (support may be subject to charging, depending on your licence agreement). |