

Axon.ivy 6.7

Engine Guide

Axon.ivy 6.7: Engine Guide

Publication date 23.08.2017

Copyright © 2008-2017 AXON IVY AG

1. Introduction	1
Audience	1
Overview	1
Installation Environment	1
Engine Types	3
Server Platforms	4
System Administrator Tasks	5
Workflow Administrator Tasks	7
2. Installation	9
Upgrading from an older version	9
Standard Edition Installation	10
Enterprise Edition Installation	11
Install Axon.ivy Engine	12
System Database	15
Elasticsearch installation	23
3. Configuration	26
How to start the Engine Configuration application	26
Engine Configuration	26
Control Center	32
Configure Tomcat	35
Additional Security Configuration	37
Error Handling	37
GC Optimization	39
System Database Encryption	39
4. Integration	41
Introduction	41
Apache Integration	42
Microsoft IIS Integration	44
Axon.ivy Cluster Integration	68
Web Application Firewall	70
5. Administration	72
Opening the administration tool	72
Applications	73
Application Default Settings	76
Process Models and Process Model Versions	78
Project Deployment	80
Business Calendar	86
Environments	88
Users and Roles	96
System Properties	101
Engine infos	103
HTML Workflow UI	105
Email Notification	109
6. Monitoring	114
Logging	114
Process Element Performance Statistic and Analysis	118
Java Management Extensions (JMX)	119
VisualVM	123
Engine Administration	125
7. Miscellaneous	126
Rich Dialogs	126
HTML Dialogs	130
Replacing Java Runtime with newer version	130
Update Notification	130
Tool Reference	131
8. Troubleshooting	136
Introduction	136
Program / Engine start problems	136

Blocking Application/UI 138

Chapter 1. Introduction

Audience

This guide is intended for

- *System Administrators* that need to install, configure and administrate the Axon.ivy Engine
- *Workflow Administrators* that have to manage the processes, users, tasks and cases running on the Axon.ivy Engine

Overview

Axon.ivy consists of two parts.

The *Axon.ivy Designer* which allows you to build Web Applications and Rich Internet Applications (known under the term RIA). Web Applications are applications that run on a web server and communicate with the application users over a web browser (Internet Explorer, Firefox, etc.). Rich Internet Applications on the other hand are applications where the pure presentation layer of the user interface runs on the client and the logic of the UI and of the application runs on a server. The Axon.ivy Designer allows you to build applications that can:

- Run as Web Application with a HTML UI
- Run as a Rich Internet Application in a JVM
- Read and write data from/to databases.
- Call web services to get and set data from/to external systems
- Be internationalized by using the content management system.
- Perform Business Process Management tasks let different user interact with business processes.

The *Axon.ivy Engine* is responsible to execute applications in a productive environment. Once an application has been developed and tested with the Axon.ivy Designer, it may be deployed to the Axon.ivy Engine.

Installation Environment

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

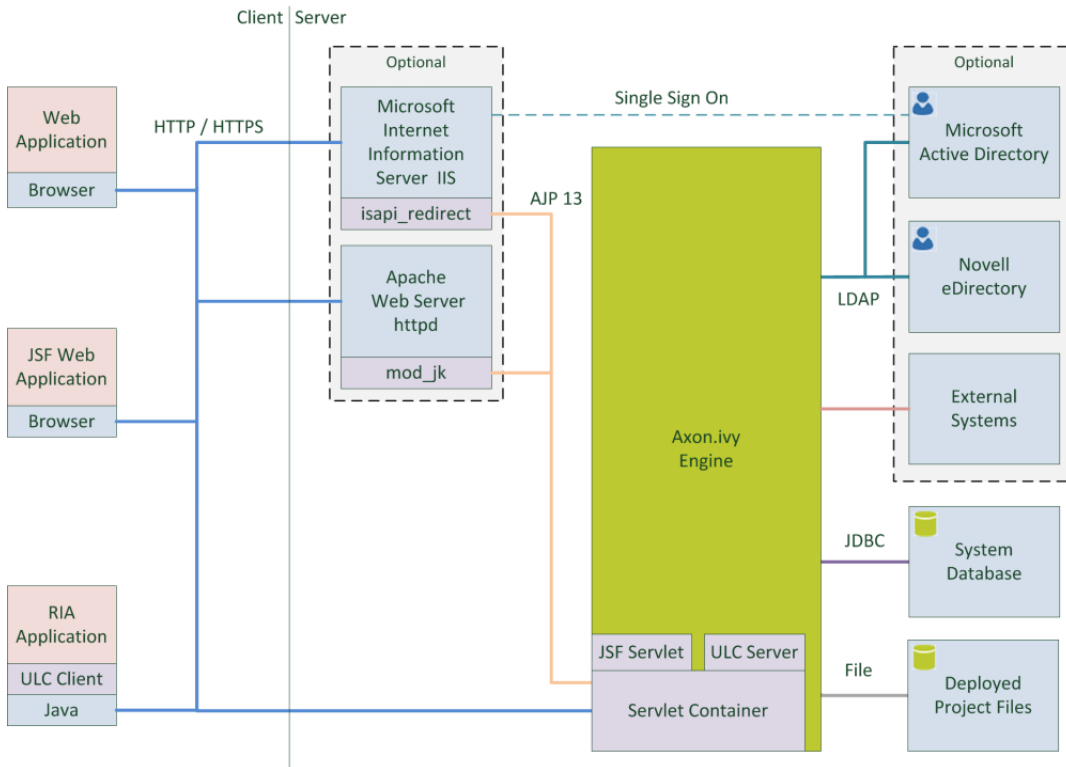


Figure 1.1. Axon.ivy Engine Installation

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 Java 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 web server like Microsoft Internet Information Server (IIS) or the Apache Web Server is put in front of the Axon.ivy Engine. The web 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.



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)

Path	Description
.../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.

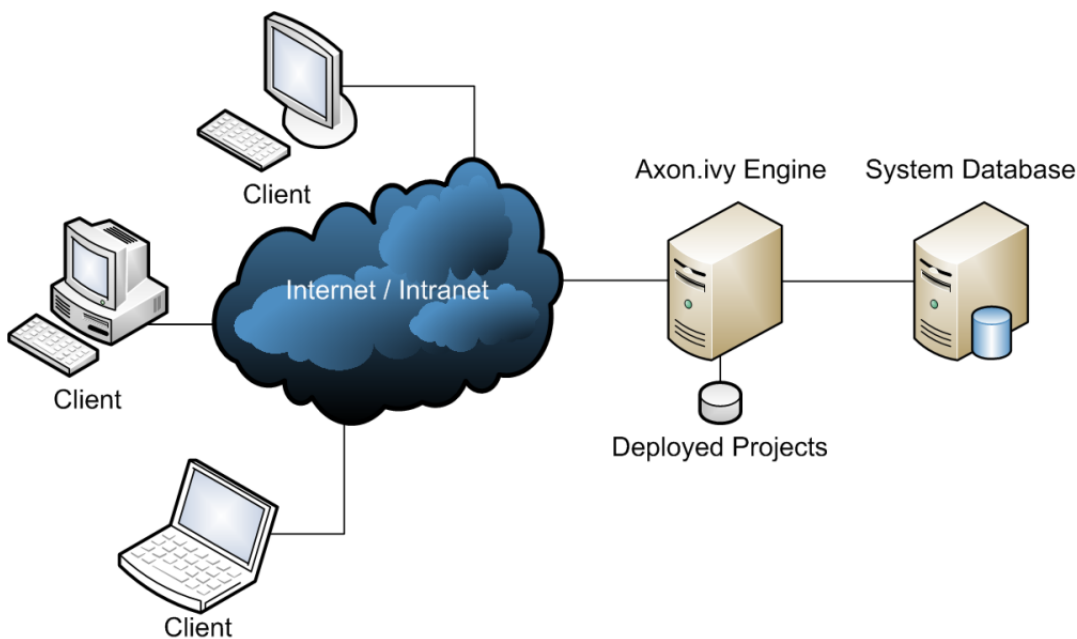


Figure 1.2. Axon.ivy Engine Standard Edition

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.

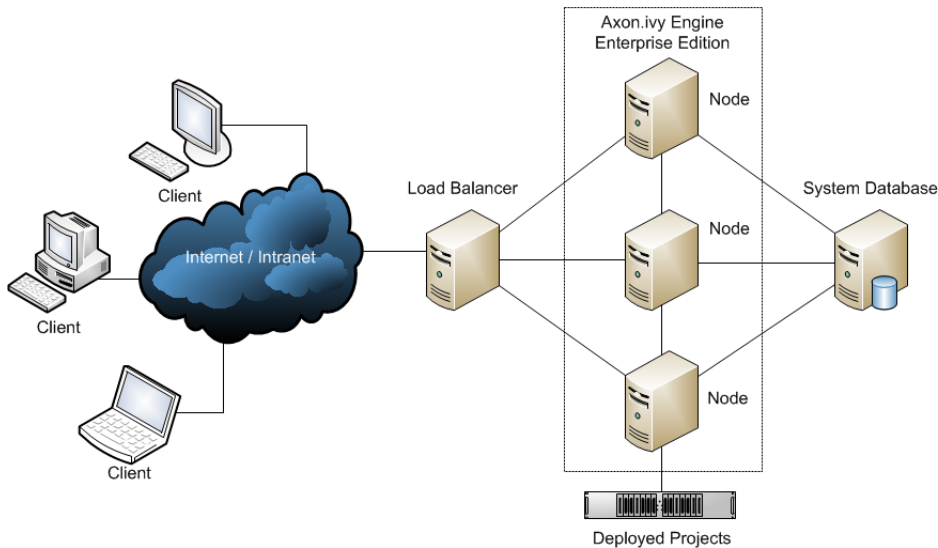


Figure 1.3. Axon.ivy Engine Enterprise Edition

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

Server Platforms

The Axon.ivy Engine supports the following platforms:

- Windows (Intel x86)
- Windows (Intel x64)
- Linux (Intel x86)

- Linux (Intel x64)

System Administrator Tasks

The following figure shows the main tasks for which an *Axon.ivy Engine system administrator* is responsible for:

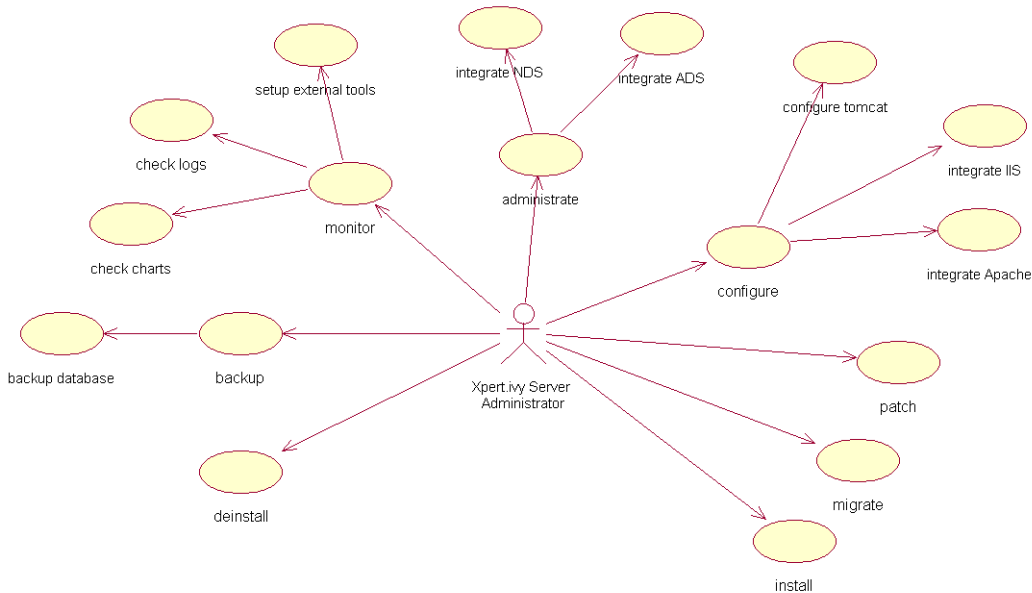


Figure 1.4. Axon.ivy Engine System Administrator Tasks

System Administrator Skills and Responsibilities

An Axon.ivy Engine administrator is responsible for installing, configuring, administrating, monitoring and backing up an Axon.ivy Engine. The following skill set is required:

- Knowledge of the platform(s) (Server Hardware and Operating System) where the Axon.ivy Engine is/are installed.
- Knowledge of the database systems that the Axon.ivy Engine uses as it's system database.
- Basic knowledge of the TCP/IP protocol.
- Knowledge of web servers such as *IIS* or *Apache* if the Axon.ivy Engine is to be integrated with them.
- Knowledge of an external security system such as *Active Directory Server* if it is to be used for user management.

An Axon.ivy Engine system administrator is in most cases a member of the IT / infrastructure department.

Installation

Installing an Axon.ivy Engine.

For more information about how to install an Axon.ivy Engine see chapter Installation

Migration

Migration of an old Axon.ivy Engine installation to a newer version.

For more information about how to migrate Axon.ivy Engine to a newer version see section Upgrading from an older version in chapter Installation.

Patching

Patching of an existing Axon.ivy Engine installation.

For more information about how to patch an Axon.ivy Engine installation see section Install Patch and Install Hot Fix in chapter Installation.

Configuration

Configuration of an Axon.ivy Engine. There are different levels of configuration:

- *Offline configuration*: You can configure an Axon.ivy Engine with the *Axon.ivy Engine Configuration* application. The engine must be restarted to make the configuration changes active.
- *Online configuration*: With the *Engine Administration* application you can configure system properties without the need to restart the engine in order for the changes to take effect.
- *Tomcat configuration*: You can configure the internal *Tomcat* application server (session timeout) in the Tomcat configuration files.

For more information about offline configuration see chapter Configuration.

For more information about online configuration see section System Properties in chapter Engine Administration

For more information about tomcat configuration see section Tomcat Configuration in chapter Configuration.

Integration with IIS

Integration of Axon.ivy Engine into *Microsoft Internet Information* server.

For more information about how to integrate with the Microsoft Internet Information server see chapter Integration.

Integration with Apache

Integration of Axon.ivy Engine into the *Apache* web server.

For more information about how to integrate with the Apache web server see chapter Integration.

Tomcat Configuration

Configuration of the *Tomcat* application server which is embedded into the Axon.ivy Engine.

For more information about the Tomcat configuration see section Tomcat Configuration in chapter Configuration.

Administration

Creation and maintenance of applications, process models, process model versions. Configuration of external databases, etc.

For more information about Axon.ivy Engine administration see chapter Engine Administration.

External Security System Integration

Configuration of an application, so that the users of the application are imported from and synchronized with a *Microsoft Active Directory* or *Novell eDirectory* server.

For more information about integrating with an external security system see section Configuring external security system in chapter Engine Administration.

Monitoring

Monitoring of a running Axon.ivy Engine installation to identify failures and or problems in the applications.

For more information about monitoring an Axon.ivy Engine see chapter Monitoring.

Setup of External Tools

Installation and set up of external tools that help checking the health of a running Axon.ivy Engine installation.

Engine Backup

Backup of all files and resources that contain runtime information of an Axon.ivy Engine.

Backup of Databases

Backup of the system database and all external databases which are accessed by Axon.ivy Engine.

Uninstallation

Uninstallation of an Axon.ivy Engine.

Workflow Administrator Tasks

The following picture shows the main tasks that an *Axon.ivy Engine workflow administrator* is responsible for:

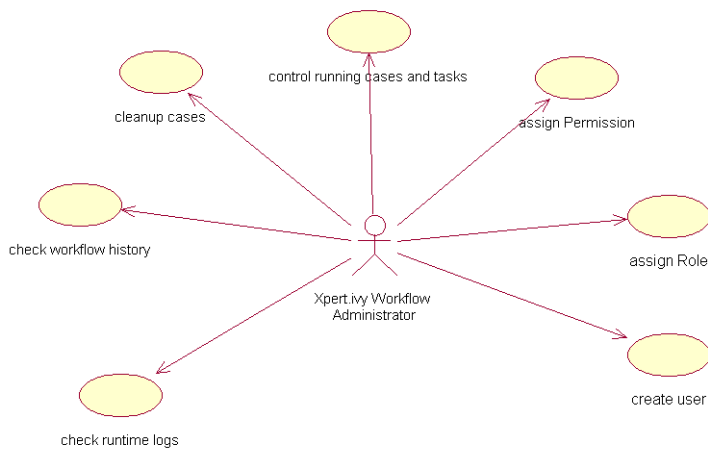


Figure 1.5. Axon.ivy Engine Workflow Administrator Tasks

Workflow Administrator Skills and Responsibilities

An Axon.ivy Engine workflow administrator is responsible for the creation of users, assigning of roles and permissions, controlling the state of the running cases and tasks, cleaning up cases, checking the workflow history and inspection of the runtime logs for problems. An Axon.ivy Engine workflow administrator must have the following skills:

- Knowledge of the running business processes.
- Knowledge of the business process roles.
- Knowledge of the users and their roles and permissions within the running business processes

An Axon.ivy Engine workflow administrator is in most cases a member of the organisation department.

User Creation

Creation of authorized users for a running application.

Role Assignment

Assignment of roles to users.

Permission Assignment

Assignment of permissions to users and roles.

Controlling of Running Cases and Tasks

Controlling of the running cases and tasks of a workflow process. Unassigned tasks (tasks with no role or no user who is responsible for them) must be assigned by the workflow administrator to a user or role. Furthermore, it is possible that system tasks fail. In this case the workflow administrator is responsible to solve such a problem.

Cleanup of Cases

Finished cases should be cleaned up periodically. This reduces the disk space occupied by the system database uses.

Checking of Workflow History

The workflow system maintains a history with a log entry for every workflow step. This history should be periodically checked for errors or unexpected behaviour.

Checking of Runtime Logs

The process model prints out certain events (errors, warnings etc) to the runtime logs. Those logs should be checked periodically to identify problems.

Chapter 2. Installation

Upgrading from an older version



Warning

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

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* from the *configuration* directory of the old installation directory to the *configuration* directory of the new installation. This file contains the system database configuration and connection parameters.
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* directory 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.

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 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 Axon.ivy Engine Configuration Application (see section System Database). Or set *autoConvert* property to *true* in the *serverconfig.xml* in 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 Elasticsearch installation for more information.
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. Install Axon.ivy Engine as Windows Service or Linux systemd service if desired.
2. Integrate Axon.ivy Engine into web servers if necessary.
3. Configure Axon.ivy Engine system properties and create and configure 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 **AxonIvy** and password **AxonIvy**.

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 Engine Configuration. 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 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 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.

The following platforms are supported:

CPU	Architecture	Operation System	Installation ZIP File
Intel	x86	Windows	AxonIvyEngineX.Y.Z_Windows_x86.zip
Intel	x64	Windows	AxonIvyEngineX.Y.Z_Windows_x64.zip
Intel	x64	Linux	AxonIvyEngineX.Y.Z_Linux_x64.zip
Intel	x64	Linux and Windows*	AxonIvyEngineX.Y.Z_Slim_All_x64.zip

Table 2.1. Supported Axon.ivy Engine Platforms

* The slim engine is delivered with launchers for Linux and Windows, but without a JRE, Portal and projects (Workflow UIs). 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.

**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>application/</i>	
<i>ServerConfiguration/</i>	Contains the Axon.ivy Engine configuration application
<i>System/</i>	Contains the Axon.ivy Engine administration application
<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

Folder or File Name	Description
<i>signed/windows_native/</i>	Native Windows libraries
<i>configuration/</i>	Contains the Axon.ivy Engine configuration data
<i>demo.lic</i>	Demo licence file
<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
<i>log4jconfig.xml</i>	Logging configuration
<i>serverconfig.xml</i>	System database 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>configuration/</i>	The OSGi configuration
<i>features/</i>	Installed OSGi features
<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 2.2. 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 ControlCenter .
<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 AxonIvyEngineConfig .
<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 AxonIvyEngineService .

Table 2.3. Windows Programs

Unix Programs

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

File	Description
<i>AxonIvyEngine</i>	Script that starts the Axon.ivy Engine. For more information see section AxonIvyEngine .
<i>AxonIvyEngine.sh</i>	Deprecated engine start script (use <i>AxonIvyEngine</i> instead).
<i>AxonIvyEngine.conf</i>	Java virtual machine configuration (Xms, Xmx, JMX, ...) for the Engine.
<i>AxonIvyEngineConfig</i>	Launches the Axon.ivy Engine Configuration program that allows to configure the Axon.ivy Engine. For more information see section AxonIvyEngineConfig .
<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>AxonIvyEngineService.sh</i>	Scripts that starts the Axon.ivy Engine service.
<i>control.conf</i>	Java virtual machine configuration (Xms, Xmx, JMX, ...) for the control tools (ControlCenter & Config)
<i>ControlCenter</i>	Script to launch the control center that allows to configure, start and stop the Axon.ivy Engine. For more information see section ControlCenter .
<i>InstallService.sh</i>	Script to install the Axon.ivy Engine as a daemon. For more information see section InstallService.
<i>launcher.sh</i>	Script to launch a Java program.

Table 2.4. Unix 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
- Oracle
- Microsoft SQL Server
- DB2 for iSeries (AS400)
- Postgre SQL

Password Encryption

Passwords are stored encrypted in the system database using state of the art encryption algorithm. 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.

Compatibility

You have to use at least MySQL Version 5.1 and the database engine InnoDB.

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 request
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 2.5. 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 2.6. 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 2.7. 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
Collation	utf8_unicode_ci

Table 2.8. MySQL Default Database Configuration

Oracle

Information

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

Compatibility

You have to use at least Oracle 10g.

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

Field Name	Value(s)	Description
Port	1521, 1522, 1523, ...	The IP port where the database system is listening for request
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
Password	****	The password of the database user who is used to connect to the database system

Table 2.9. 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.PersistencyException: 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`. This must be done in either the `.ilc` file (on Windows) or the shell script (on Linux) of the server instance that you're starting.

System properties are set as additional Java VM properties, e.g. - `Dch.ivyteam.ivy.persistence.db.oracle.MaxOpenCursors=2000` to set the maximum number of open cursors to 2000. See Java VM help if you need further information.

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. Tables, indexes and views are created in the schema of this user.
Password	***	The password for the given user.

Table 2.10. 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 2.11. 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. For more information about the jTDS JDBC Driver see jtds.sourceforge.net

Compatibility

You have to use at least Microsoft SQL Server 2005.

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	The database JDBC driver to use.

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

Table 2.12. Microsoft SQL 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` 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 2.13. Microsoft SQL Creation Parameter

Driver

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

JDBC Driver Name	net.sourceforge.jtds.jdbc.Driver
JDBC Connection URL Format	jdbc:jtds:sqlserver://<host>[:<port>]/<database name>

Table 2.14. Microsoft SQL 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 2.15. Microsoft SQL Server Default Database Configuration

DB2 for iSeries (AS400)

Information

IBM calls its database products DB2. It is important to understand that DB2 for z/OS is not the same as DB2 for LUW (Linux, Unix, Windows) and not the same as DB2 for iSeries. This chapter describes DB2 for iSeries (formally AS400). For more information go to www.ibm.com.

Compatibility

You have to use at least DB2 for iSeries V6R1M0.

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	DB2 iSeries (AS400)	The database system to use.
Driver	DB2 iSeries (AS400)	The database JDBC driver to use.
Host	localhost, testdbserv, ...	The name of the host where the database system is running
Port	50000 , 50001, 50002, ...	The IP port where the database system is listening for request
Schema Name	AxonIvy, AxonIvyEngine,	The name of the SQL schema (library) to use. Name with more than 10 characters are not supported.
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 2.16. DB2 for iSeries 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 a DB2 for iSeries database system:

Field Name	Value(s)	Description
Schema	AxonIvy, Ivy, ...	The name of the database schema that will be created and where the Axon.ivy

Field Name	Value(s)	Description
		Engine system database tables will be created in.

Table 2.17. DB2 for iSeries Creation Parameter

Driver

The following table shows information about the JDBC driver used by Axon.ivy Engine to connect to DB2 database systems:

JDBC Driver Name	com.ibm.as400.access.AS400JDBCdriver
JDBC Connection URL Format	jdbc:as400://<host>[:<port>]/<schema name>

Table 2.18. DB2 for iSeries Driver

Character set & Collation

DB2 for iSeries (AS400) uses CCSID (coded character set identifier) for encoding definitions which can be configured on system level. To influence the sorting and comparing you can configure the additional connection properties in the engine config ui. The supported properties are listed in the IBM Toolbox for Java JDBC properties. Especially the sort properties `sort`, `sort language` and `sort weight` are interesting for case insensitive searches.

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

Compatibility

You have to use at least PostgreSQL 8.3.

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 request
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 2.19. 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 2.20. 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 2.21. 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 2.22. PostgreSQL Default Database Configuration

Elasticsearch installation

To quickly search through Business Data, ivy makes use of the powerful Elasticsearch search engine. You can use the bundled Elasticsearch server or use an external Elasticsearch cluster.

Bundled Elasticsearch server

By default an Elasticsearch server is bundled with the Axon.ivy Engine. This Elasticsearch instance is started when the Engine starts and runs in an own process. The default Elasticsearch installation is reachable only on `localhost` and is unprotected.

The following System properties can be customized to configure the bundled Elasticsearch servers. The `Elasticsearch.BundledServer.DataPath` should be set to a directory outside of the Engine installation directory. Typically, there's no need to change the other ones.

System Property name	Description
<code>Elasticsearch.BundledServer.DataPath</code>	The path to the directory where the bundled Elasticsearch server stores data. By default the data is store in the <code>elasticsearch/data</code> directory inside the Engine installation directory. We recommend to use a seperate data directory that is located outside of the Engine installation directory to ease the Engine migration.
<code>Elasticsearch.BundledServer.ClusterName</code>	The name of the cluster of the bundled Elasticsearch server.

Table 2.23. System properties for the bundled Elasticsearch server

The Java virtual machine arguments used to start the bundled Elasticsearch server can be configured in the file `elasticsearch/config/jvm.options`.

External Elasticsearch server

If you perform an Axon.ivy Engine Enterprise Edition installation or choose to install an external Elasticsearch server, you can follow the Elasticsearch installation steps. Currently we support Elasticsearch server versions in the 5.5.x range. If your Elasticsearch server is running on another server instance you have to secure the access to that instance.

You have to let the engine know where your Elasticsearch server instance is running by configuring the following System properties:

System Property name	Description
<code>Elasticsearch.ExternalServer.Url</code>	The URL of the external Elasticsearch server.
<code>Elasticsearch.ExternalServer.UserName</code>	Name of the user to use to authenticate in the external Elasticsearch server.
<code>Elasticsearch.ExternalServer.Password</code>	Password of the user to use to authenticate in the external Elasticsearch server.
<code>Elasticsearch.IndexNamePrefix</code>	The name prefix of the indices to use to store business data. If multiple ivy Engines use the same Elasticsearch server instance, you need to change this property, that every ivy Engine has an unique indices. For every business data type an Elasticsearch index will be created. E.g. for type <code>ch.ivy.Dossier</code> the index name is <code><IndexNamePrefix>-ch.ivy.dossier</code> . Default: <code>ivy.businessdata</code>

Table 2.24. System properties for an external Elasticsearch server

Disk Space

The disk space needed to store business data on the Elasticsearch and system database server depends on the size and number of values that are stored. The following table shows how much space is needed for business data values with a size between 2 and 4 KB.

Number of Values	System Database (MySQL)	Elasticsearch
20'000	150 MB	110 MB
100'000	720 MB	230 MB

Number of Values	System Database (MySQL)	Elasticsearch
1'000'000	6.9 GB	2.2 GB

Table 2.25.

Chapter 3. Configuration

How to start the Engine Configuration application

There are several possibilities to launch the Axon.ivy Engine Configuration application:



Note

Note, that the configuration procedure implies sufficient administration and access rights. For example on a Windows Server 2008, you have to run the Engine Configuration tool with the "Run as Administrator" option.

Launch from 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* in the *bin* directory of the Axon.ivy Engine installation directory.

Linux: Start the *ControlCenter.sh* shell script in the *bin* directory of the Axon.ivy Engine installation directory to start the ControlCenter program.

Direct Launch

You can start the Axon.ivy Engine Configuration program directly.

Windows: Start the *AxonIvyEngineConfig.exe* in the *bin* directory of the Axon.ivy Engine installation directory.

Linux: Start the *AxonIvyEngineConfig.sh* shell script in the *bin* directory of the Axon.ivy Engine installation directory.

Launch in Headless Mode

You can start the Axon.ivy Engine Configuration with the option `-headless` to start the program in the headless mode. Headless mode is useful if you operate in a non graphical user interface environment. If you start the program with the `-headless` option, a rich dialog application engine is started. Once the engine is up and running, it prints out a URL to the console. You can now use another computer, to start a web browser and type in the provided URL. The web browser will start the Axon.ivy Engine Configuration user interface using *Java Webstart*. You can use it to configure the Axon.ivy Engine as explained in the this chapter. When you are finished with the configuration, switch back to your server and press a key in your console to stop the rich dialog application engine.

Windows: Use the *AxonIvyEngineConfigC.exe* in the *bin* directory of the Axon.ivy Engine installation directory with the option `-headless` to start the Axon.ivy Engine Configuration in headless mode.

Linux: Use the *AxonIvyEngineConfig.sh* shell script in the *bin* directory of the Axon.ivy Engine installation directory with the option `-headless` to start the Axon.ivy Engine Configuration in headless mode.

Engine Configuration

The Axon.ivy Engine Configuration application's user interface is divided into tabs. On the Licence tab you can upload a licence. On the System Database tab the system database can be configured, created or converted. On the Administrators tab system administrators can be registered. This tab is only enabled if a valid system database is configured. On the Web Server tab the web server protocols and ports can be configure. This tab is also only enabled if a valid system database is configured.

Click *Save* to store the modified data on all tabs. Hit *Discard Changes* to reload the configuration from the filesystem and database. The Cluster tab is only visible if you have installed an Enterprise Edition Licence.

The configuration on the System Database tab is stored in the configuration file `configuration/serverconfig.xml`. The configurations on the other tabs are stored in the system database that is configured in its own tab.



Note

The changes that you make with the Engine Configuration do not become active unless you restart the engine.

Licence

On the *Licence* tab you have to upload a valid licence:

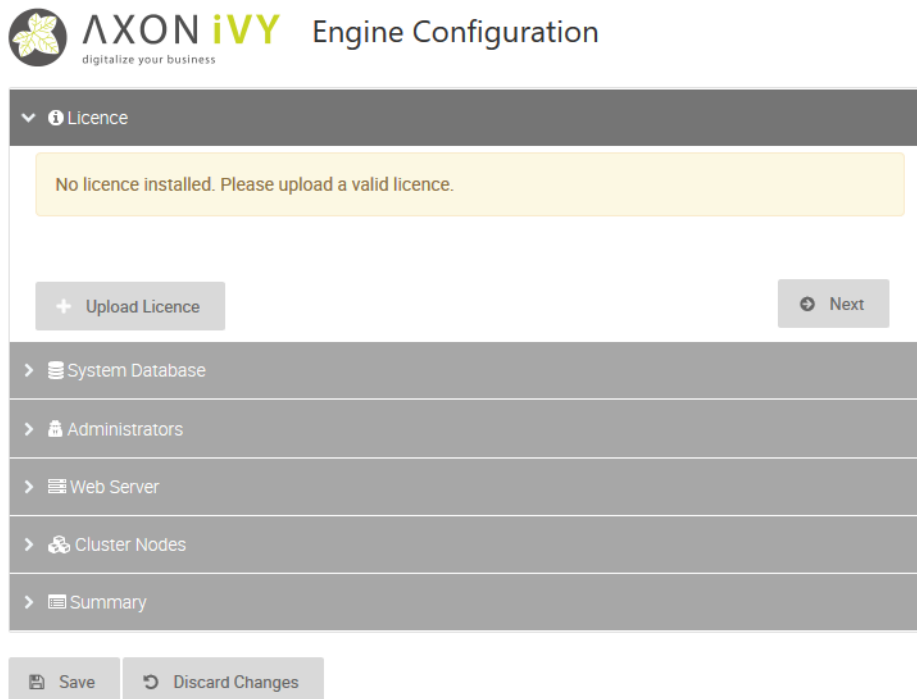


Figure 3.1. Axon.ivy Engine Configuration Licence Tab

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:

The screenshot shows the 'System Database' configuration tab in the Axon.ivy Engine Configuration interface. The top navigation bar includes 'Licence' and 'System Database'. A light blue banner at the top of the configuration area displays a warning: 'Connection state unknown' with a checkmark icon and the text 'Please check the connection to the Database.' A 'Check Connection' button is located to the right of the banner. Below the banner, the configuration fields are as follows:

Database	MySQL	▼
Driver	mySQL	▼
Host	dbServer	
Port	3306	<input checked="" type="checkbox"/> Default
Database Name	myDatabase	
Username	user	
Password	••••••••	

Below the fields, there is an 'Additional Properties' button with a plus icon. At the bottom of the configuration area, there are three buttons: 'Create Database', 'Convert Database', and 'Next'. The bottom navigation bar includes 'Administrators', 'Web Server', 'Cluster Nodes', and 'Summary'. At the very bottom of the interface, there are 'Save' and 'Discard Changes' buttons.

Figure 3.2. 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
- Oracle
- Microsoft SQL Server
- DB2 for iSeries (AS400)
- 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.

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. 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.

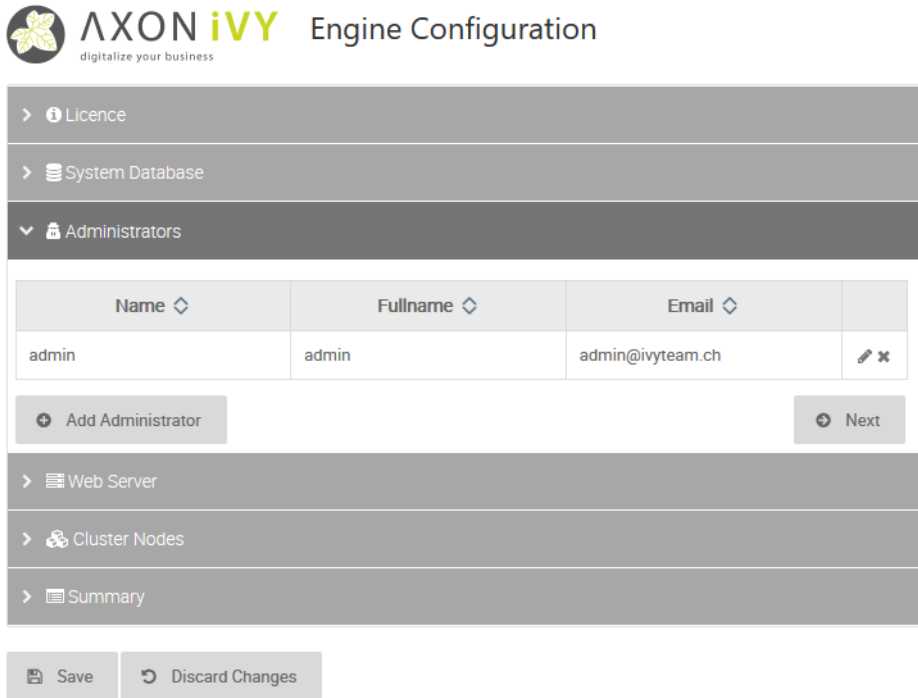


Warning

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** Engine Configuration
digitalize your business

> Licence

> System Database

▼ Administrators

Name	Fullname	Email	
admin	admin	admin@ivyteam.ch	 

+ Add Administrator Next

> Web Server

> Cluster Nodes

> Summary

Save Discard Changes

Figure 3.3. 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:

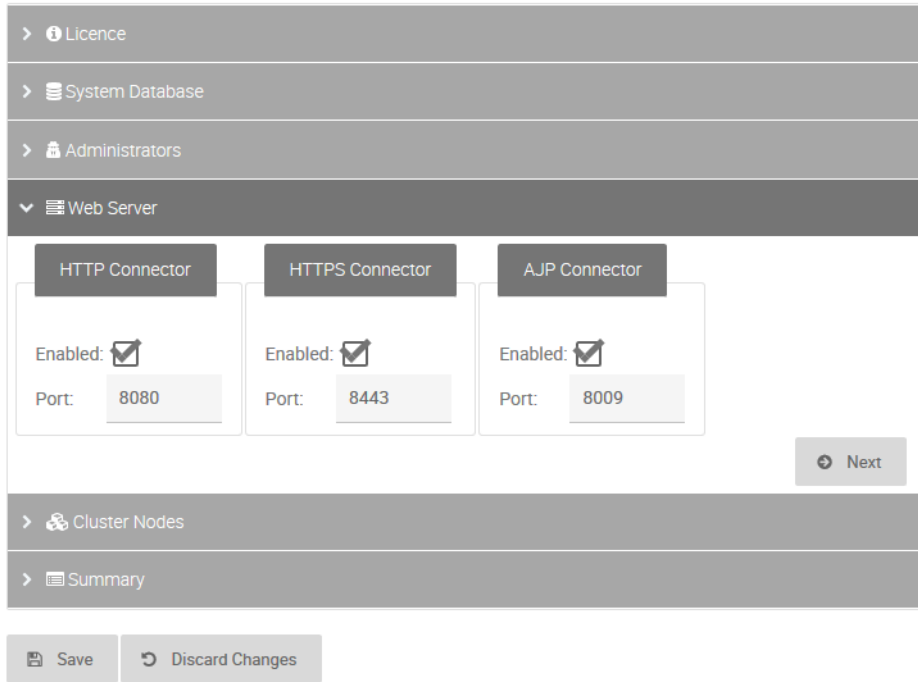


Figure 3.4. Axon.ivy Engine Configuration WebServer Tab

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 3.1. 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:

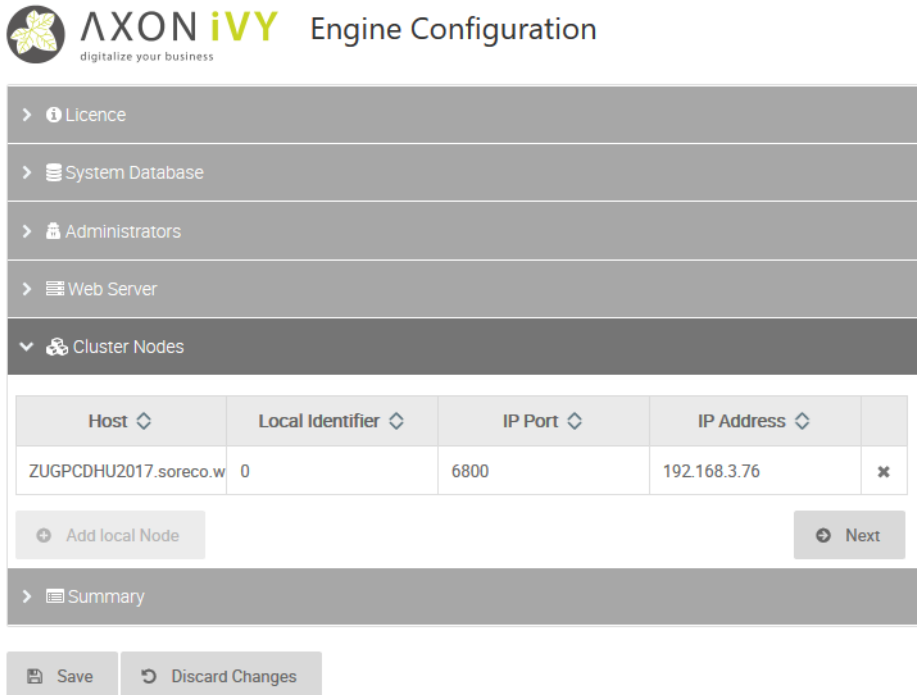


Figure 3.5. Axon.ivy Engine Configuration Cluster Tab

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.

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 *ControlCenter.sh* program located in the *bin* folder.

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.

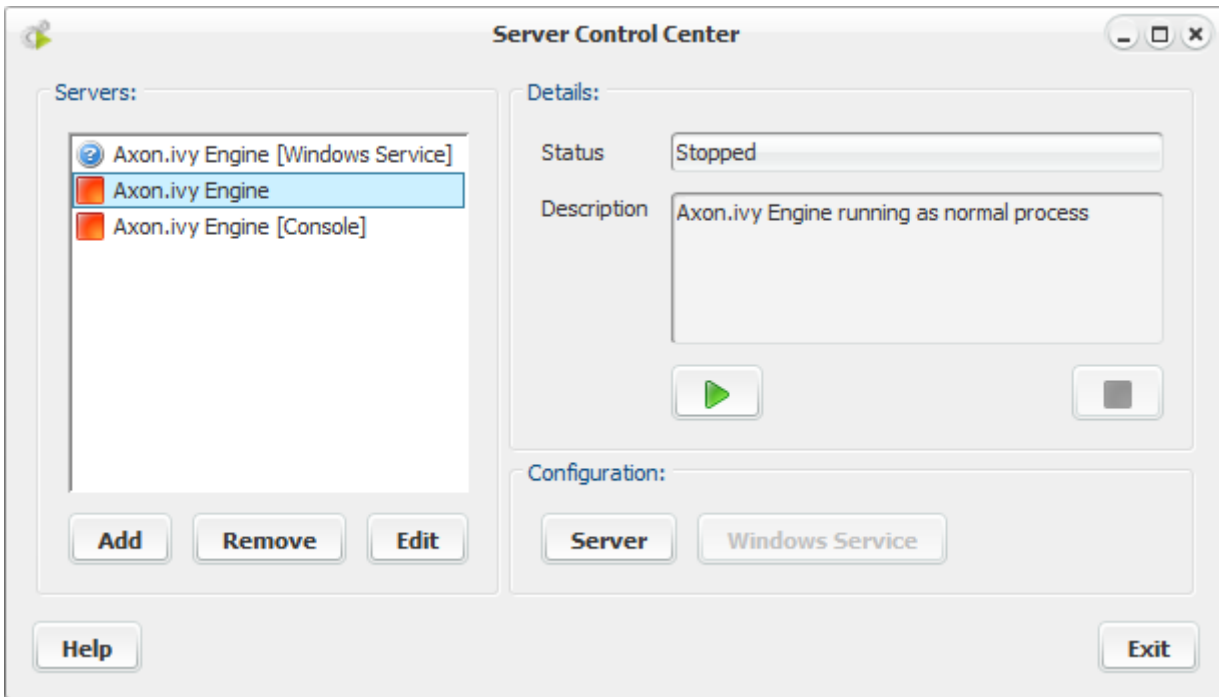


Figure 3.6. The Control Center

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:

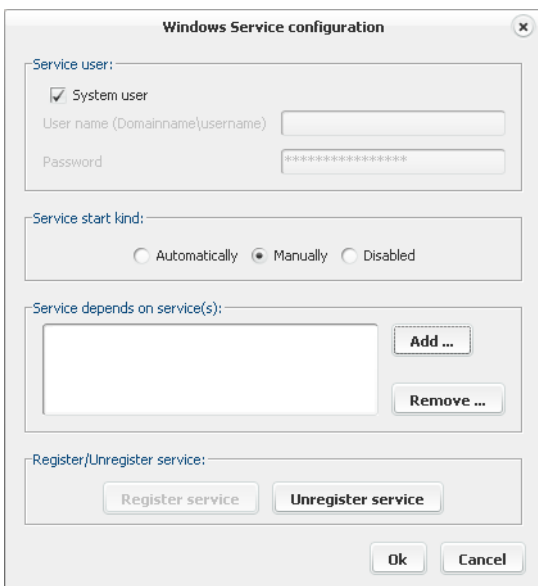


Figure 3.7. Configuring Axon.ivy Engine as Windows service

First of all press **Register Service** to register the service and to enable the rest of the configuration sections.

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 3.8. 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.

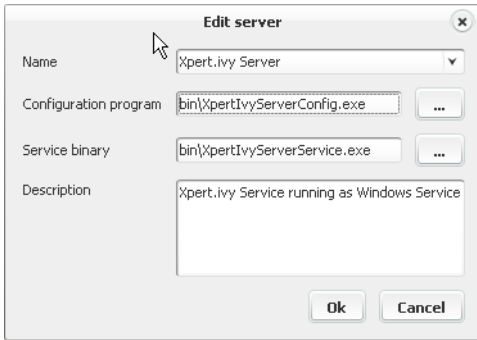


Figure 3.9. Create a new service in the server list

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.

Configure Tomcat

The directory *webapps/ivy/WEB-INF/* contains the *web.xml* file which is the webapp configuration file of the embedded Tomcat application server. See the Apache Tomcat documentation for more information.



Warning

Be very careful when changing the contents of this file. A wrong configuration or an invalid syntax in the file may harm the stability and correctness of your Axon.ivy Engine installation.



Note

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

Below is a description of settings often adjusted in an ivy *web.xml*

Session Timeout

In the *web.xml* you can set the timeout for the session. If a session (i.e. a user interaction with an application) is inactive for this amount of time, then the session will be closed. To adapt the default value look for the section below in the *web.xml* file and change the value accordingly:

```
<session-config>
<session-timeout>30</session-timeout>
</session-config>
```

Error Pages

It is possible to defined customized error pages, see chapter Custom Error Pages.

Security Headers

The ivy Engine comes with predefined HTTP Security headers for the /ivy web application. These Security Headers are added on the HTTP Responses to the Client Browser.

These security headers can be switched off or modified in the web.xml.

Header name	Value
X-Frame-Options	SAMEORIGIN
X-XSS-Protection	1; mode=block
X-Content-Type-Options	nosniff

Table 3.2. Default Values of the Security Headers



Note

Not all Security Headers are supported by all Web browsers.

Secure Session Cookies

If the Engine is accessed over HTTPS only (strongly recommended). The cookie should only be transported over HTTPS. To enable this behavior (or disable the transport over HTTP) uncomment following lines in the session-config part of the web.xml:

```
<cookie-config>
  <secure>true</secure>
</cookie-config>
```

Valves and Realms

The directory *webapps/ivy/META-INF/* contains the *context.xml* file which is the context configuration file of the embedded Tomcat application server. See the Apache Tomcat context documentation for more information. In there you can add realms and valves, see the Apache Tomcat documentation about realms or valves for more information.

The following valves are provided by Axon.ivy Engine:

Class	Documented in Chapter
ch.ivyteam.ivy.webserver.internal.PerformanceLogValve	“Request/Performance Logging”
ch.ivyteam.ivy.webserver.security.SingleSignOnValve	“Single Sign On”

Table 3.3. Valves provided by Axon.ivy Engine



Warning

Be very careful when changing the contents of this file. A wrong configuration or an invalid syntax in the file may harm the stability and correctness of your Axon.ivy Engine installation.

Custom Valve

You can configure any third party valve or even your own implementation of a valve. A custom valve implementation can only be loaded by the Axon.ivy Engine if its classes are accessible by the engines OSGi environment. This can be reached as follows:

1. The valve JAR must be implemented as OSGi bundle. This means that the standard JAR manifest (*META-INF/MANIFEST.MF*) must also contain headers to declare the *id*, *name* and *version* of this bundle. These headers will be set automatically if the bundle is created within the Axon.ivy Designer via *File > New > Other... > Plug-in Project*

2. Your bundle must require the bundle with the id `ch.ivyteam.tomcat` of the Axon.ivy Engine.
3. Your bundle must register itself as buddy of the tomcat bundle by setting the MANIFEST.MF header: `Eclipse-RegisterBuddy: ch.ivyteam.tomcat`
4. The package that contains the custom valve must be exported in the MANIFEST.
5. Export the bundle as JAR: Menu *Export > Deployable Plug-ins and Fragments*.
6. The custom bundle must be copied into the *dropins* directory of the Axon.ivy Engine.

Troubleshooting: If your valve is not working as expected, consult the *dropins/README.html*.

Sample *META-INF/MANIFEST.MF*:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ProcessingValve
Bundle-SymbolicName: com.acme.ProcessingValve;singleton:=true
Bundle-Version: 1.0.0.qualifier
Require-Bundle: ch.ivyteam.tomcat
Eclipse-RegisterBuddy: ch.ivyteam.tomcat
Export-Package: com.acme.valve
```



Tip

A full valve sample implementation can be found on GitHub: <https://github.com/ivy-samples/tomcatValve>

Additional Security Configuration

Additional Security Headers

Following additional security headers are recommended. Preferably set them on the Front-End Server (e.g. IIS, nginx, Apache).

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
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)
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 3.4. Additional Security Headers

Error Handling

When an error occurs while processing a user request an error screen is displayed to the user.

Show Error Details to End Users

By default, stacktraces, detailed error reports, etc. are not shown to end users because of security concerns when a application is running on the Engine. Instead only a unique Error Id is shown. In development or pre-production environments this behaviour is unnecessary. By changing the System Property `Errors.ShowDetailsToEndUser` to `true` this behaviour could be changed, so that all information are displayed to the user, like on the Designer.

Error Id

When 'Show Error Details to End Users' is deactivated then an error screen with an unique Error Id is shown to the user. This Error Id can be used to find the error in the log files.

Custom Error Page

An error page is used to display information about the error to the user. The content and appearance of the ivy error page can be customized. To do so, you can adjust the existing error page `webapps/ivy/ivy-error-page.xhtml` and its resources.

You are free to create your own error page(s), place them into `webapps/ivy/` and add them to `web.xml` as follows:

```
<error-page>
  <location>/custom-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 as described in Servlet 3.0 JSR in chapter 10.9.2 Error Pages.

```
<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>
```



Tip

To retrieve information about the thrown exception and the environment during the request the Public API of `ch.ivyteam.ivy.webserver.ErrorPageMBean` could be used. An instance will be available as managed bean on the error page. Check the default error page `ivy-error-page.xhtml` for an example usage.

Error Report

On a error screen it is possible to generate an error report. This report contains information about the error, the system environment and the current state of the Engine.



Tip

To create a report without an explicit error navigate to the following URL `http://{server}:{port}/ivy/error`.

IIS Error Handling

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

Since Axon.ivy 5.1 the IIS integration script configures the IIS to show the detailed error page of the Engine (see Step 'Configure Error Page' of the IIS integration chapter). With the following steps the setting could be reset the default IIS behavior. This could make sense to hide error details because of security concerns:

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 (in the right hand side)
4. Select the “Detailed errors for local requests ...” radio button and click on OK.

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 3.5. 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!

Chapter 4. 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.

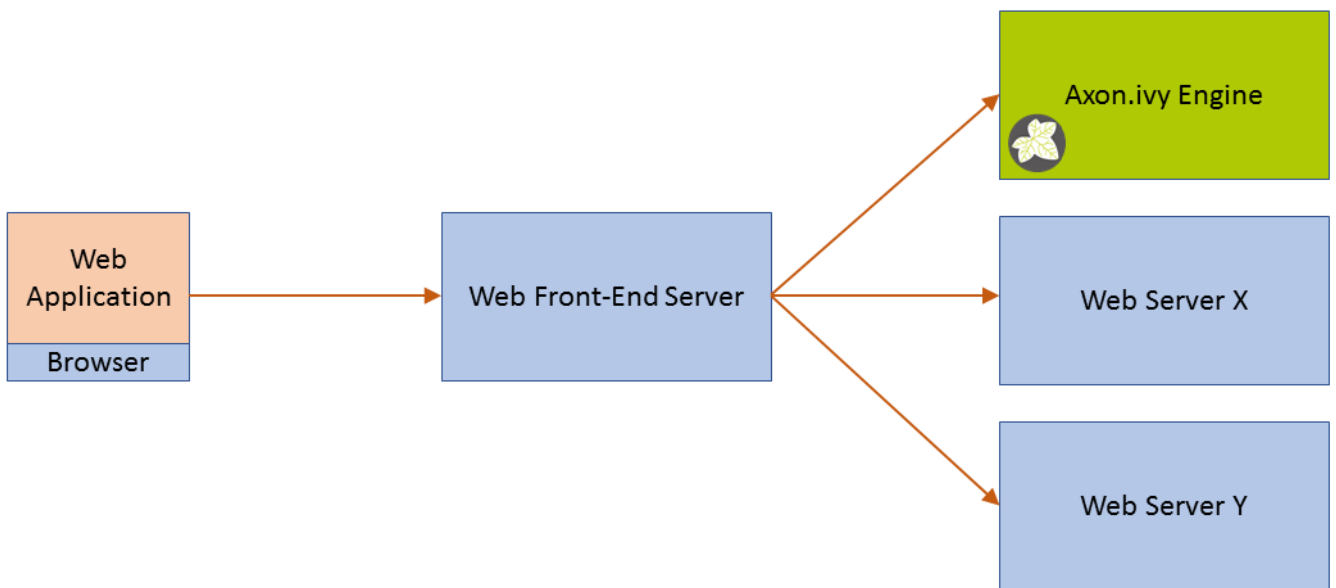


Figure 4.1. Web Front-End Server

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 (x86): *misc/apache/win/x86*
- Apache HTTP Server for Windows (x64): *misc/apache/win/x64*
- Apache HTTP Server for Linux (x86): *misc/apache/linux/x86*
- Apache HTTP Server for Linux (x64): *misc/apache/linux/x64*
- IIS for Windows (x86): *misc/iis/x86/*
- IIS for Windows (x64): *misc/iis/x64/*

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”

Update external base URL

Update the System properties with your external base URL after the web server integration. Those properties are required for creating external links (e.g. links in task mails):

- `WebServer.ExternalProtocol`(e.g. https)
- `WebServer.ExternalHostName`(The external host name)
- `WebServer.ExternalPort`(e.g. 443 for 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/win/(x86/x64)` 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>
```

- 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
```

- Update the external base URL in the Admin UI
- 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.

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

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

- Enable the new module

```
sudo a2enmod jk
```

- Update the *worker.properties* file according to the examples in the *[[Axon.ivy Engine install path]]/misc/apache/linux/ (x84/x64)*. 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
```

- 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 in the Admin UI
6. Restart the Apache HTTP Server and the Axon.ivy overview page should be accessible under *http://apacheHostName/ivy*

Change context URI /ivy/

The context URI `/ivy/` can be changed inside the Axon.ivy Engine by changing the system property `WebServer.IvyContextName` (See chapter System Properties to learn about how to change a system property). However, if you change the context URI on the Axon.ivy Engine you also have to change the context URI in the Apache HTTP Server integration. This can be done by changing the last line of the `mod_jk.conf` configuration file:

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

If you want to have `/xpertline/` as the context URI change this line to the following:

```
JkMount /xpertline/* AxonIvyEngine
```

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 /xpertline/* 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 7 (Windows Server 2008)

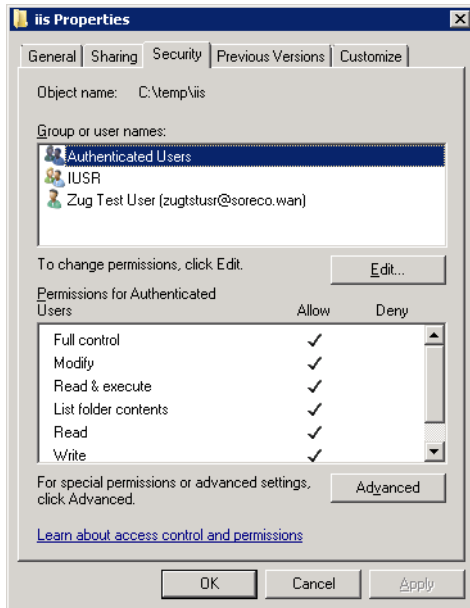


Note

There is a batch script `autoconfig.bat` in the folder `misc\iis\{x64/x86}` of your engine installation, which installs and configures the IIS automatically on a Windows 2008 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

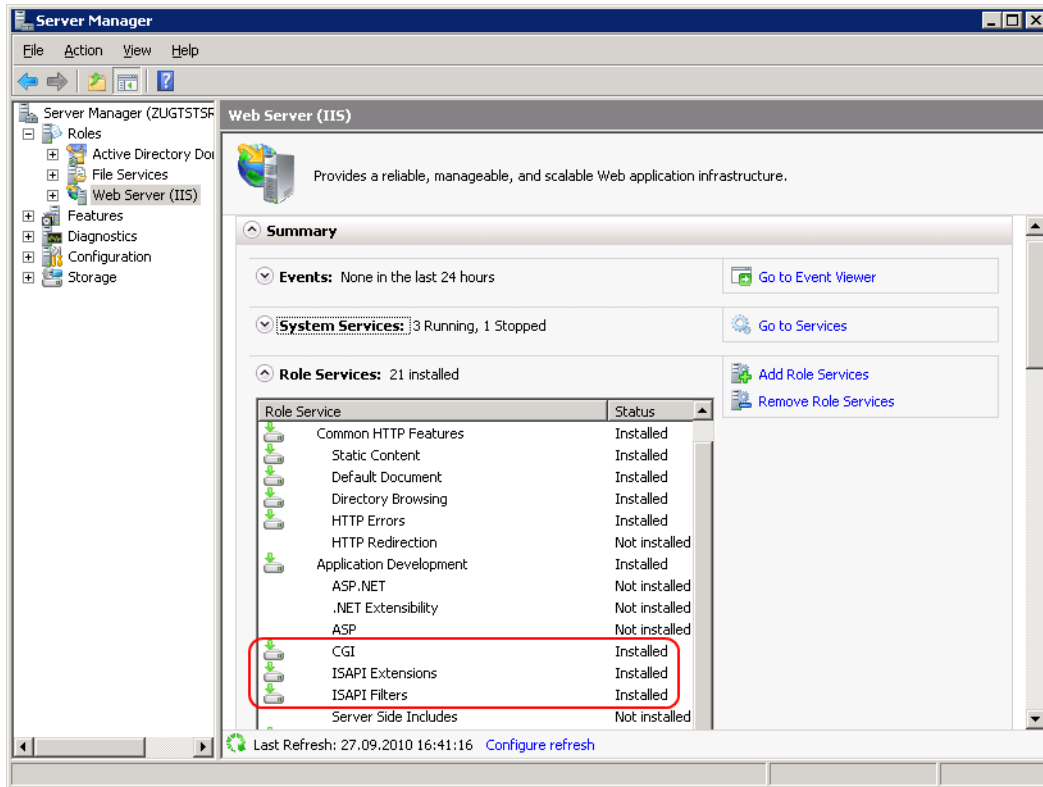


Note

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 > Control Panel > Administrative Tools > Server Manager*). Navigate to the node *Server Manager > Roles > Web Server (IIS)* and select it. 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

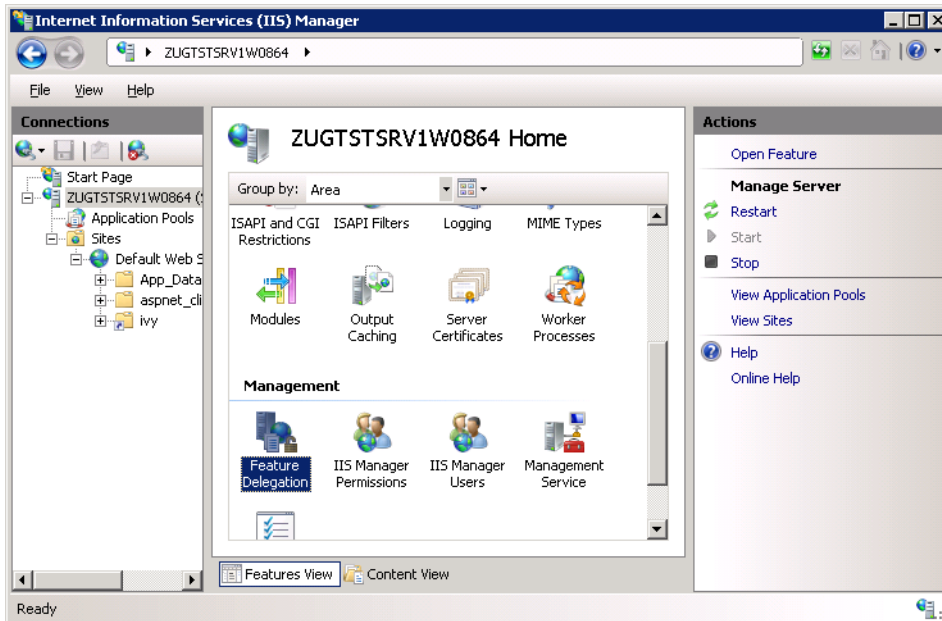


Note

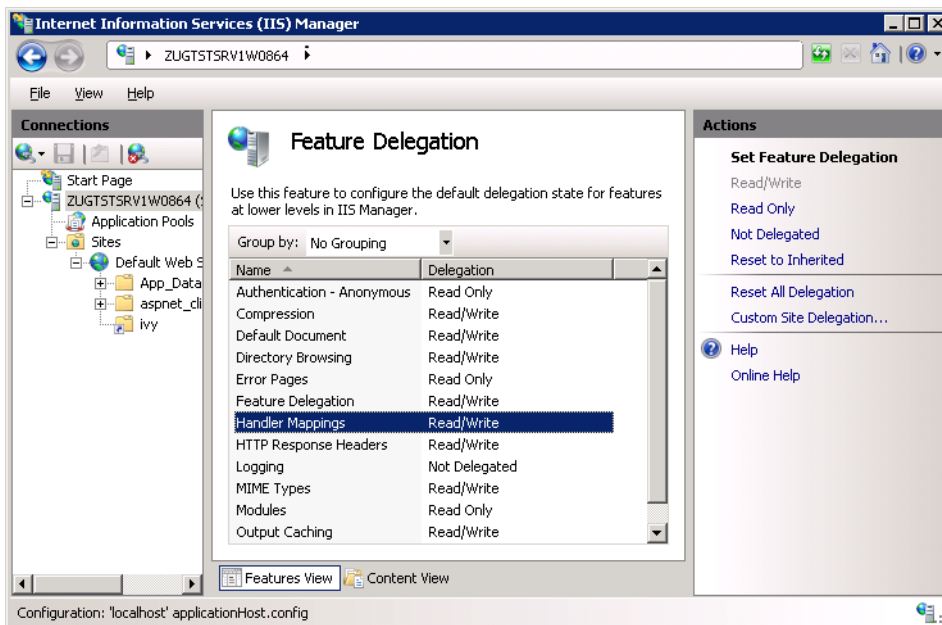
The following command automatically sets the feature delegation:

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

Open the Internet Information Services (IIS) Manager (*Start > Control Panel > Administrative Tools > 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



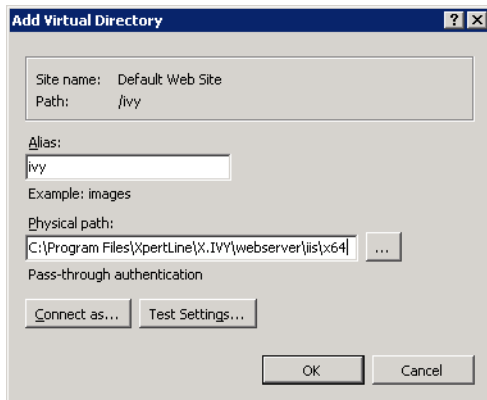
Note

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

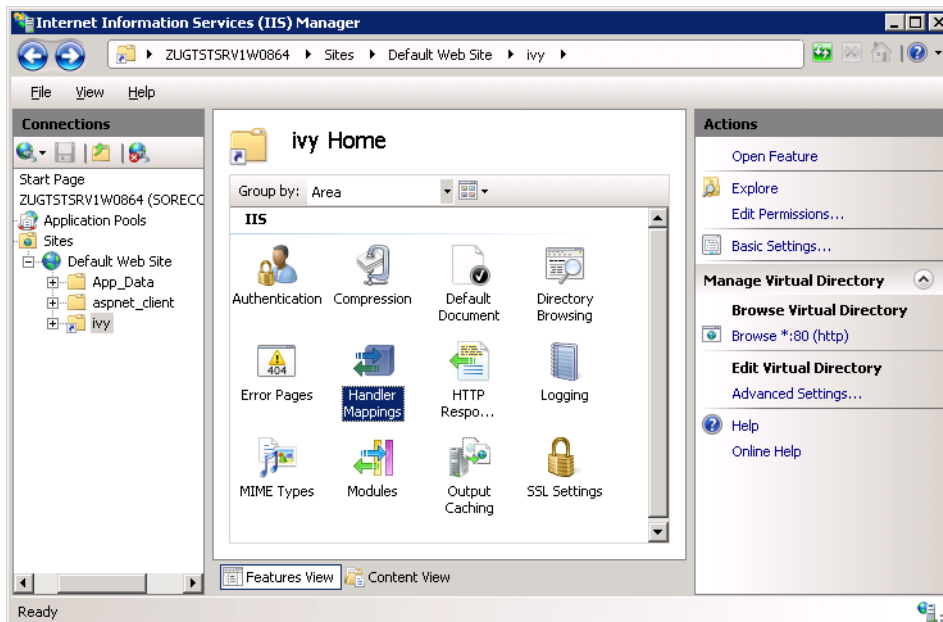


Note

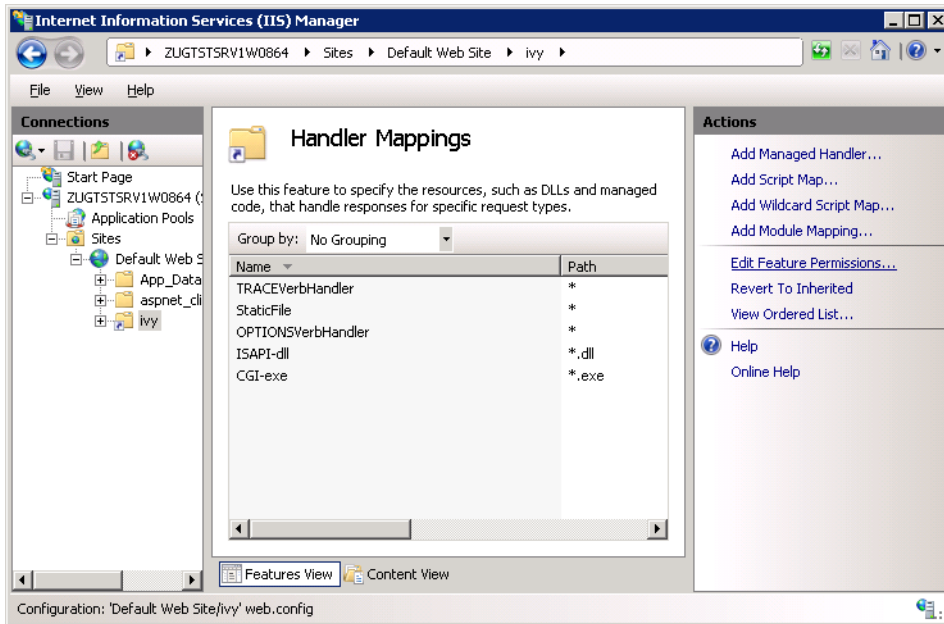
The following command automatically sets the feature permission for the ivy virtual directory:

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

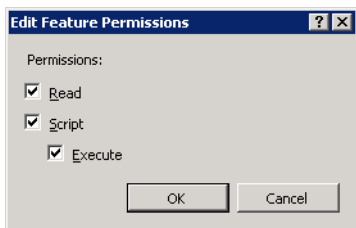
Select the new created Virtual Directory `ivy` in the `Connections` pane and open the `Handler Mappings` entry in the `Feature View`:



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:

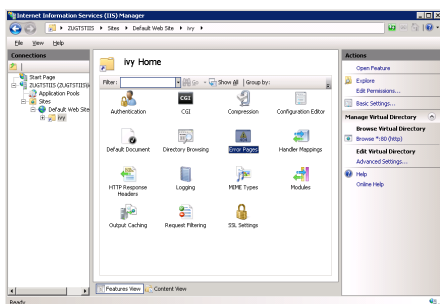
```
appcmd.exe set config "Default Web Site/ivy" /section:system.webServer/httpErrors /errorMode:Detailed
```



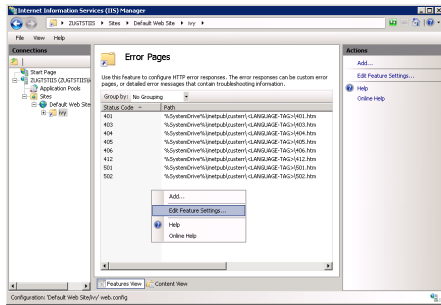
Tip

See chapter Error Handling for more information about this configuration.

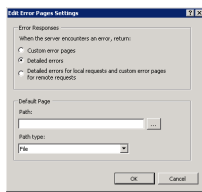
Select the new created Virtual Directory `ivy` in the Connections pane and open the `Error Pages` entry in the Feature View:



Right click and select the **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

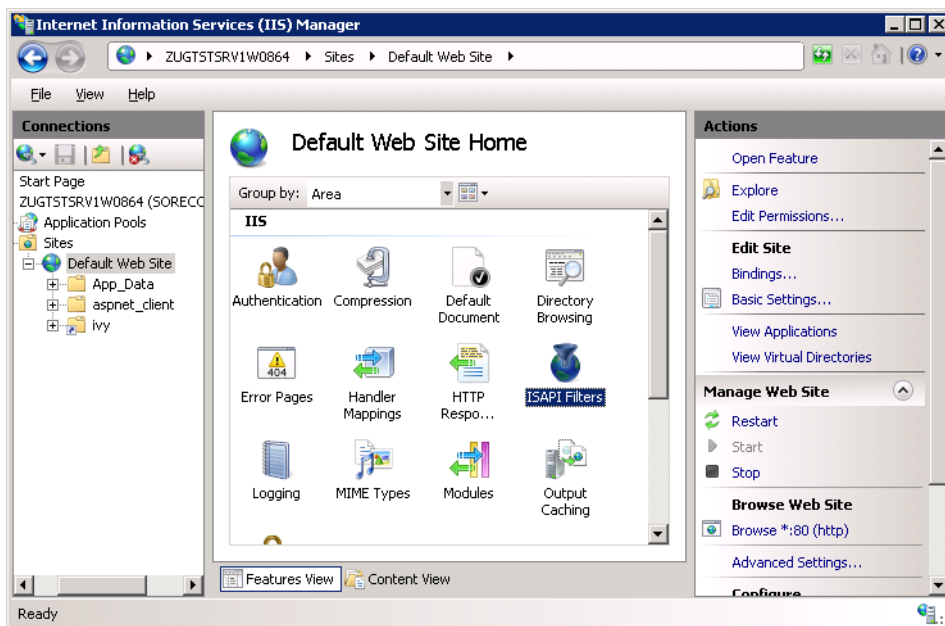


Note

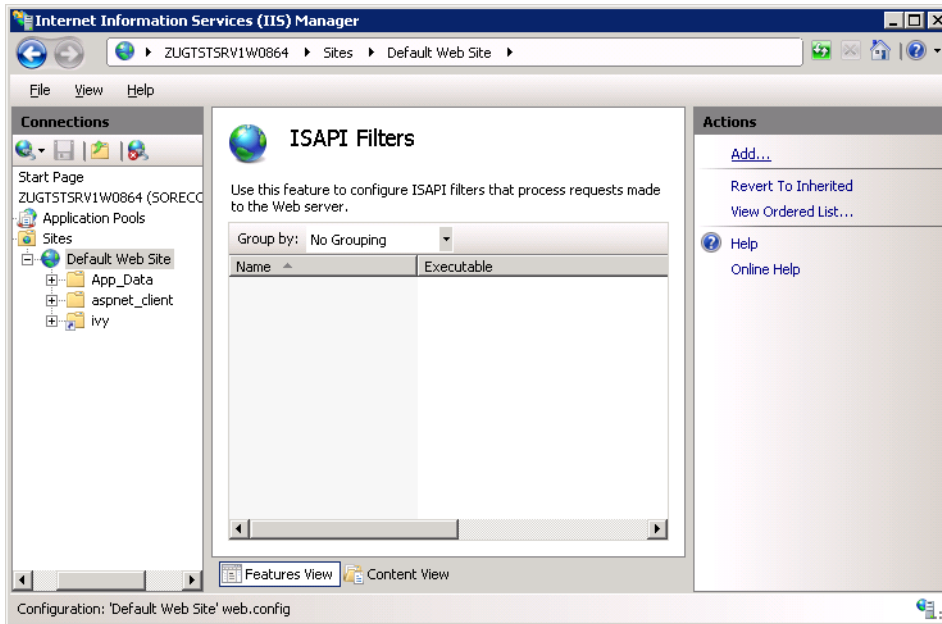
The following command automatically adds the ISAPI Filter:

```
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']
```

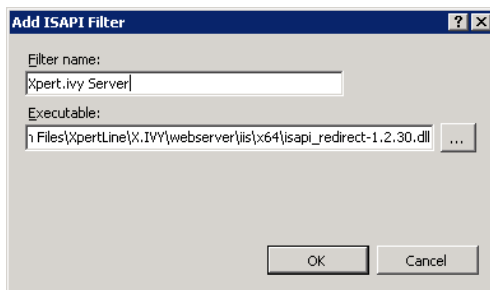
Select the **Web Site** in the **Connections** pane and open the **ISAPI Filters** entry in the **Feature View**:



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

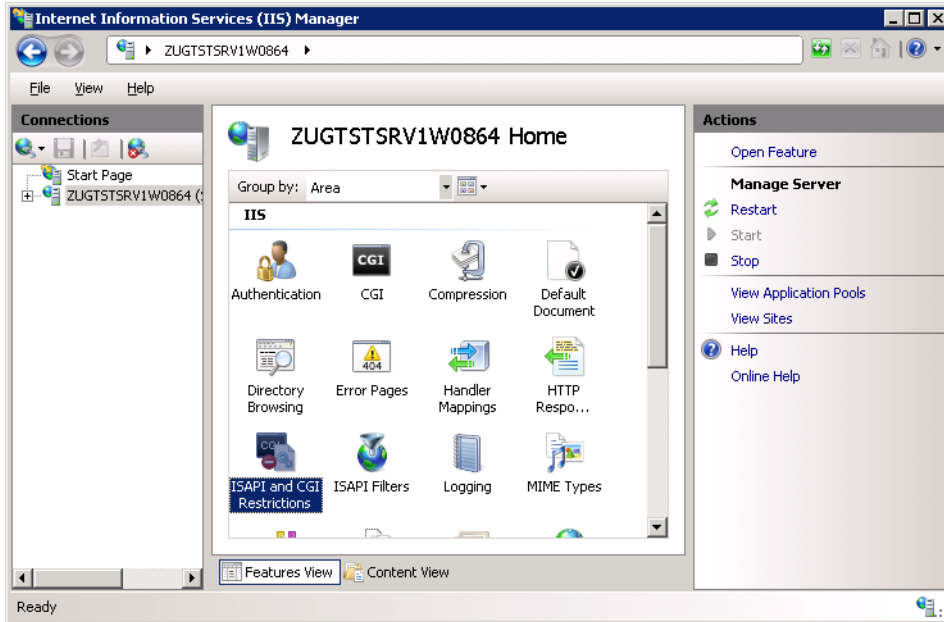


Note

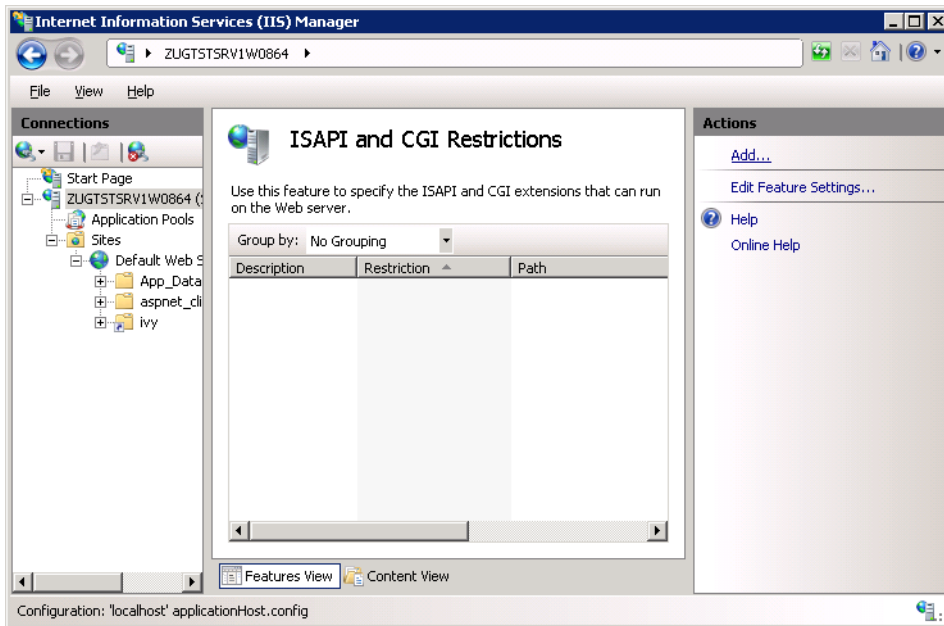
The following command automatically adds the ISAPI Restriction:

```
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']
```

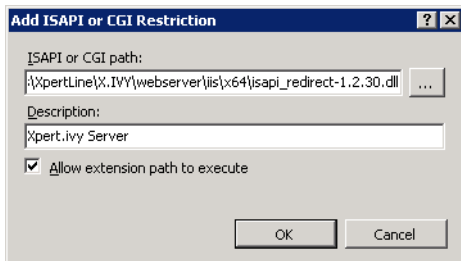
In the Connections pane select the node that represent your machine and open the ISAPI and CGI Restrictions entry in the Features View:



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 in the Admin UI
12. Check if the integration is working by opening a web browser on the address `http://<your host>/ivy/`

IIS 8 (Windows Server 2012)

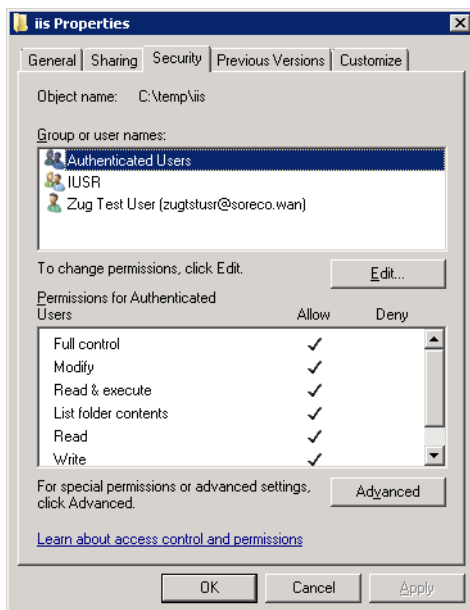


Note

There is a batch script *autoconfig.bat* in the folder *misc\iis\<x64/x86>* 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**



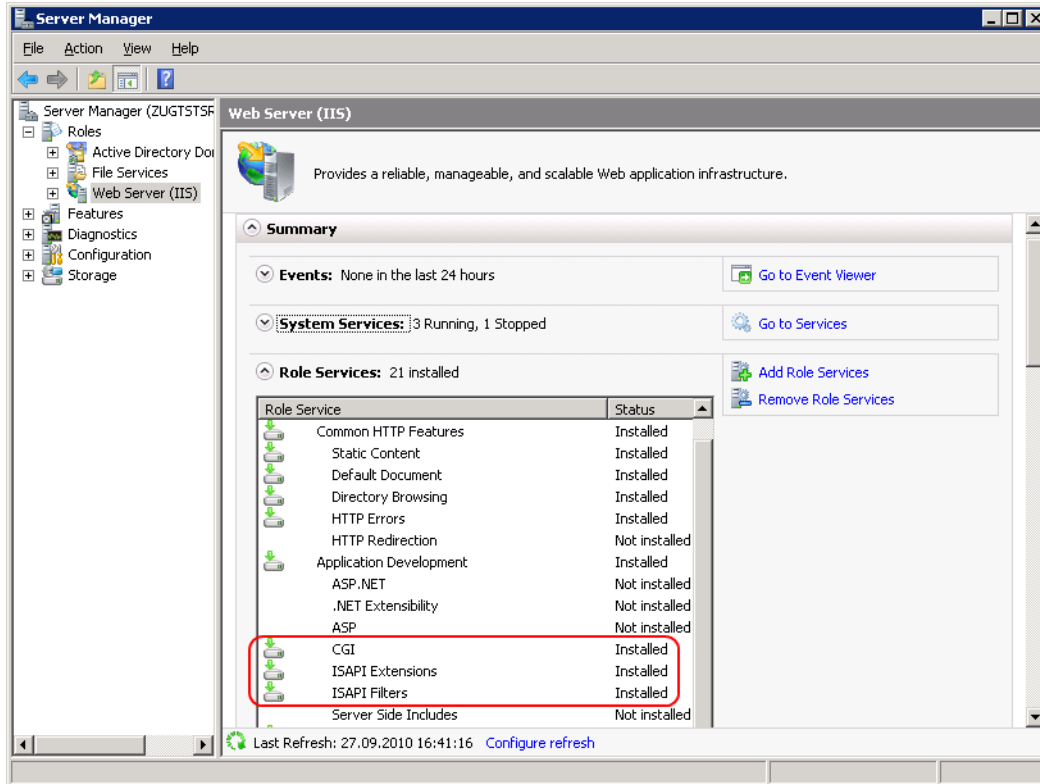
Note

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

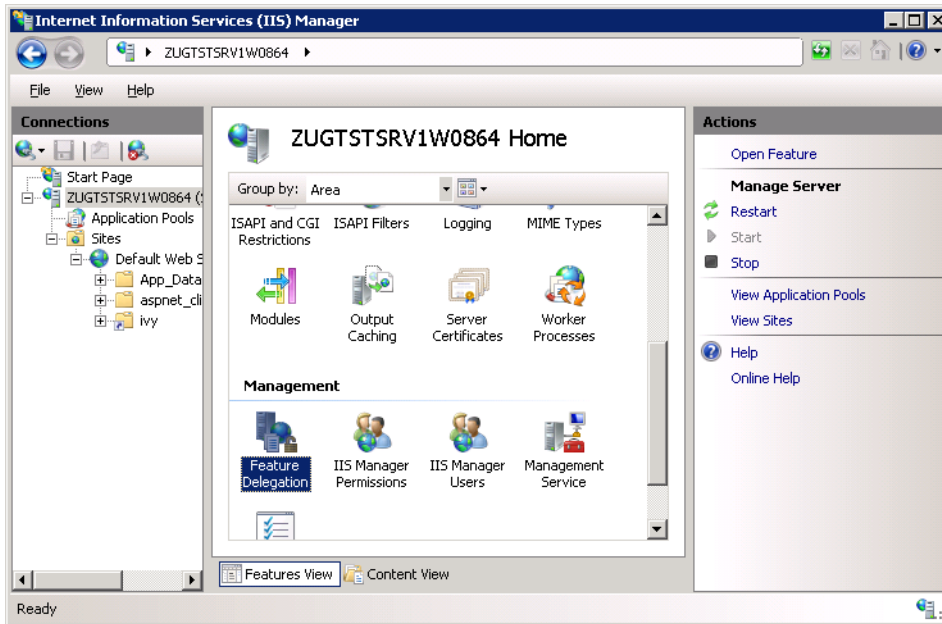


Note

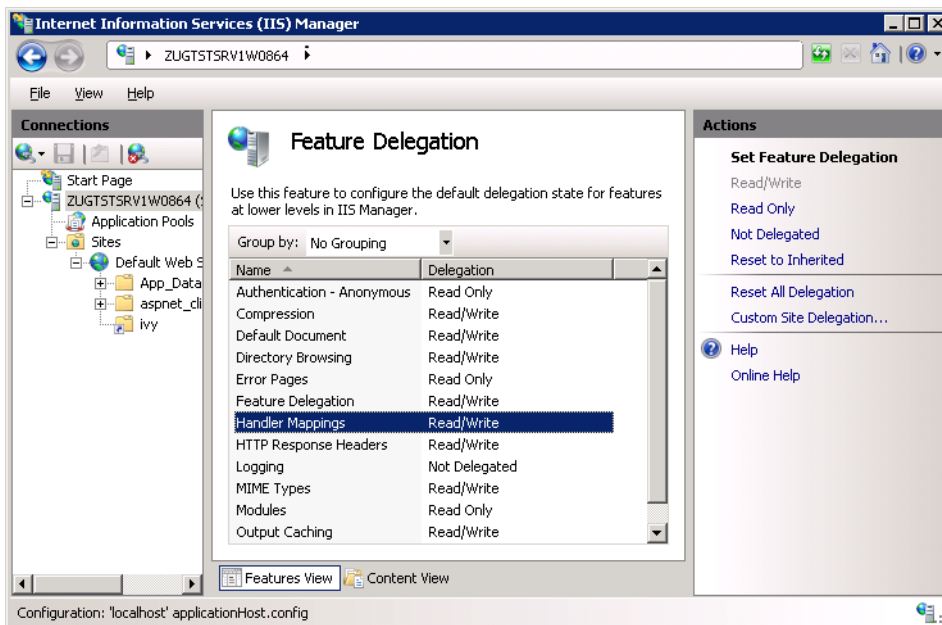
The following command automatically sets the feature delegation:

```
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



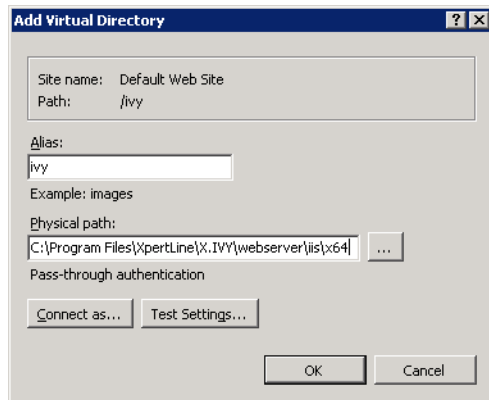
Note

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

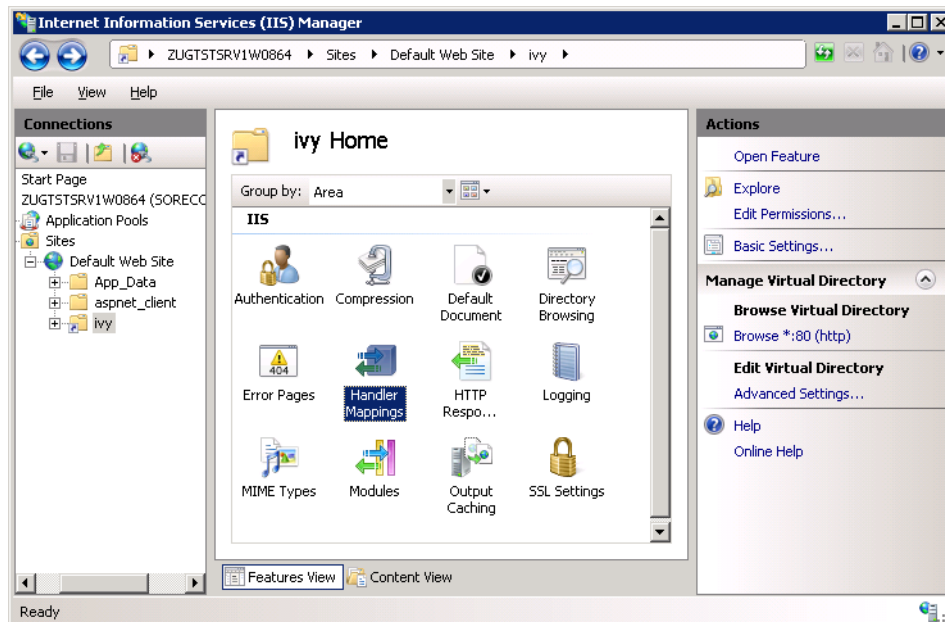


Note

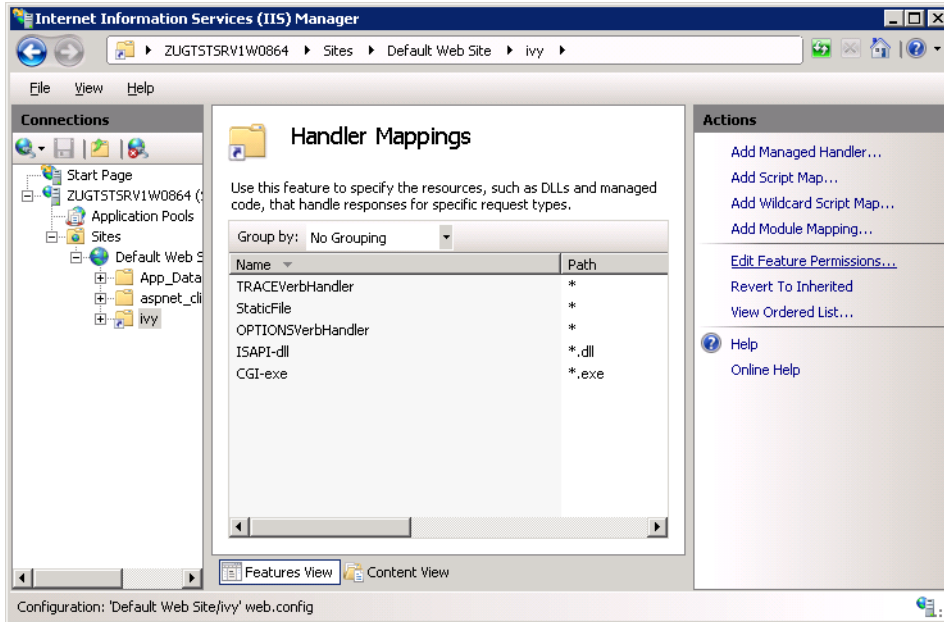
The following command automatically sets the feature permission for the `ivy` virtual directory:

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

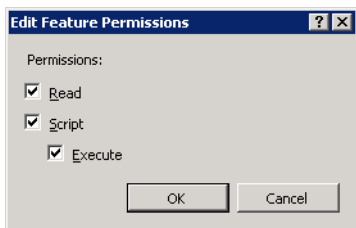
Select the new created Virtual Directory `ivy` in the `Connections` pane and open the `Handler Mappings` entry in the `Feature View`:



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:

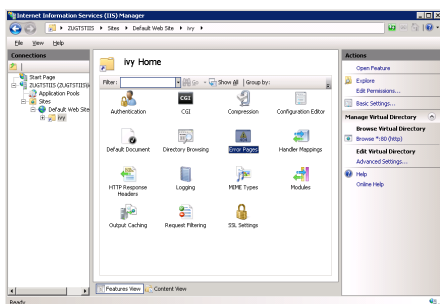
```
appcmd.exe set config "Default Web Site/ivy" /section:system.webServer/httpErrors /errorMode:Detailed
```



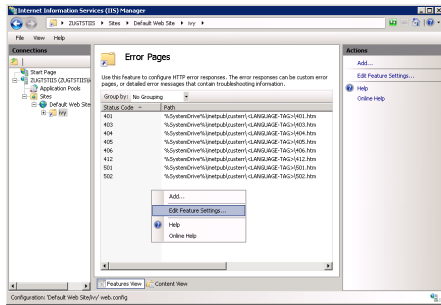
Tip

See chapter Error Handling for more information about this configuration.

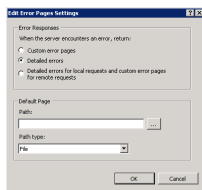
Select the new created Virtual Directory `ivy` in the Connections pane and open the `Error Pages` entry in the Feature View:



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

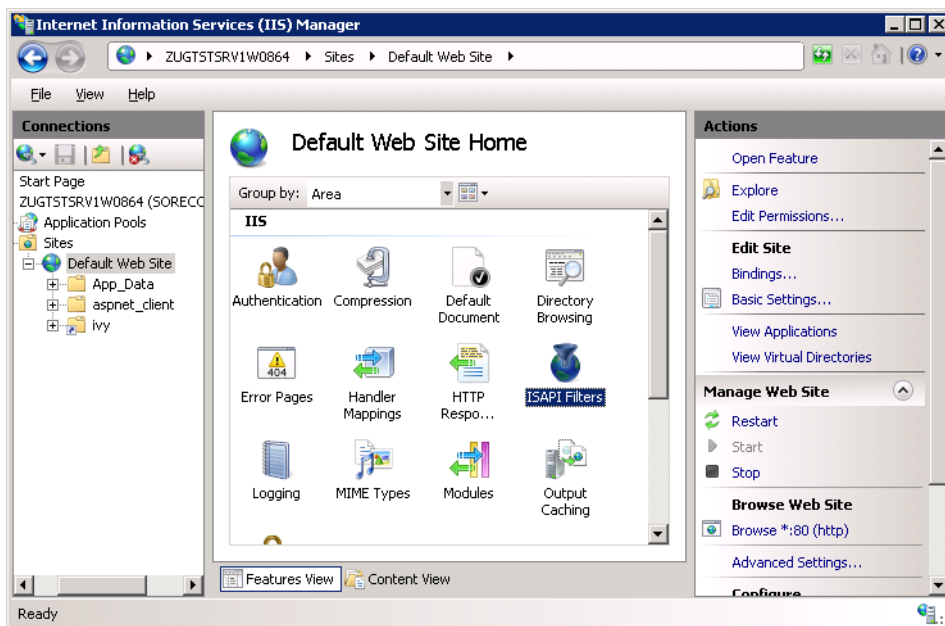


Note

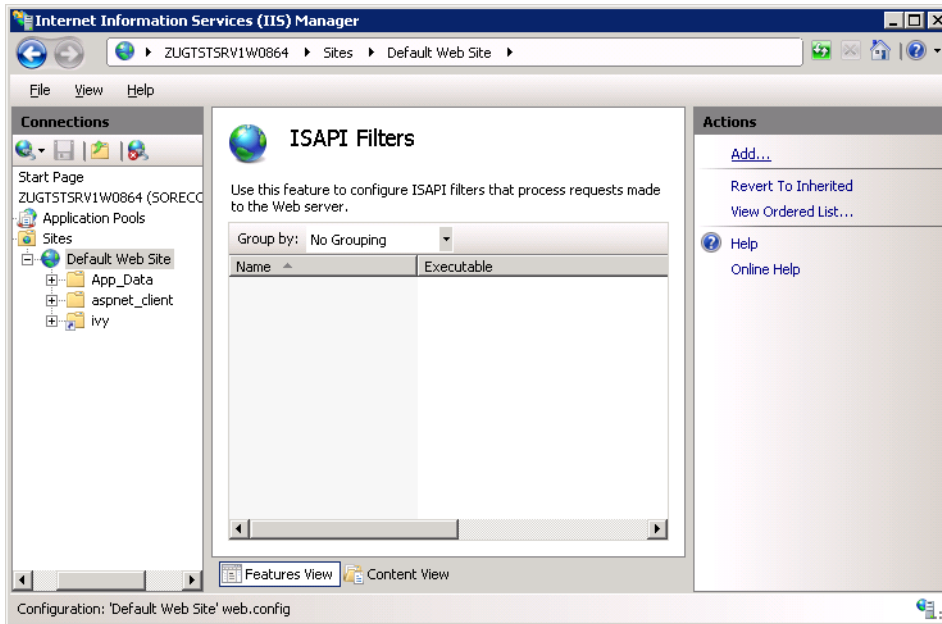
The following command automatically adds the ISAPI Filter:

```
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']
```

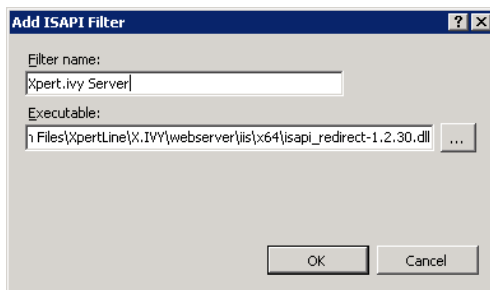
Select the **Web Site** in the **Connections** pane and open the **ISAPI Filters** entry in the **Feature View**:



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

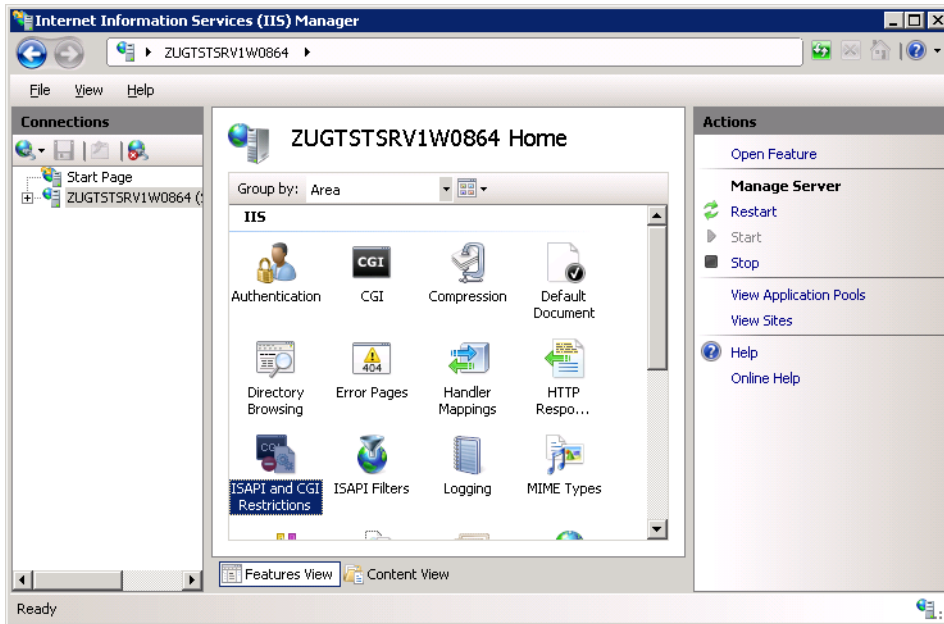


Note

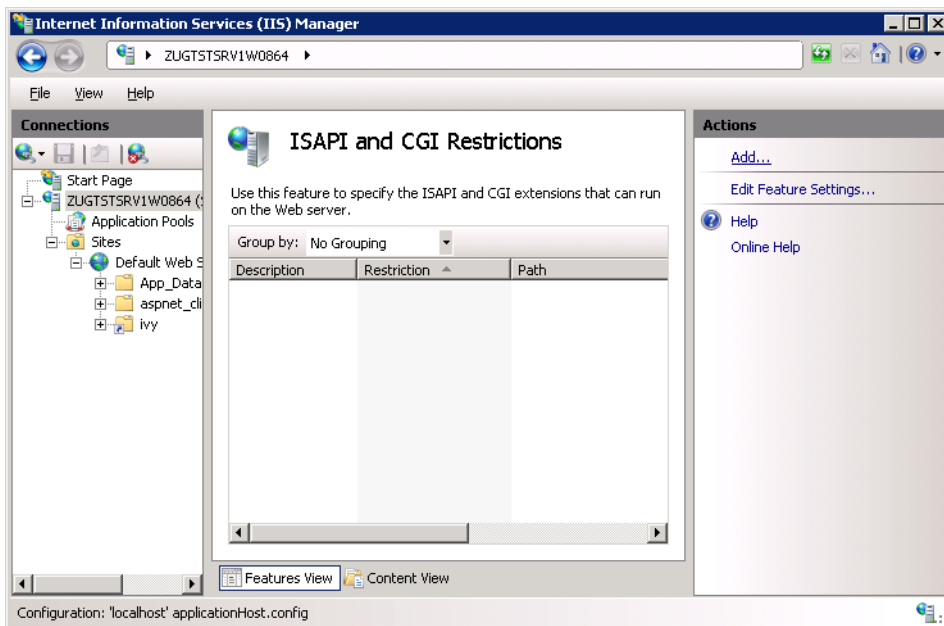
The following command automatically adds the ISAPI Restriction:

```
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 in the Admin UI
12. Check if the integration is working by opening a web browser on the address `http://<your host>/ivy/`

Change context URI /ivy/

The context URI `/ivy/` can be changed inside the Axon.ivy Engine by changing the system property `WebServer.IvyContextName` (See chapter System Properties to learn about how to change a system property). However, if you change the context URI on the Axon.ivy Engine you also have to change the context URI in the Microsoft IIS integration. This can be done by changing the last line of the *uriworkermap.properties* configuration file:

```
/ivy/*=AxonIvyEngine
```

If you want to have `/xpertline/` as the context URI change this line to the following:

```
/xpertline/*=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. For detailed explanation see “Web Server Ports”.

In our scenario the AJP port of the Axon.ivy 5.x Engine was changed to 8010 and the Xpert.ivy 3.9 Server kept his default AJP port 8009.

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 *uriworkermapping.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 *uriworkermapping.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 6 (Windows Server 2003)

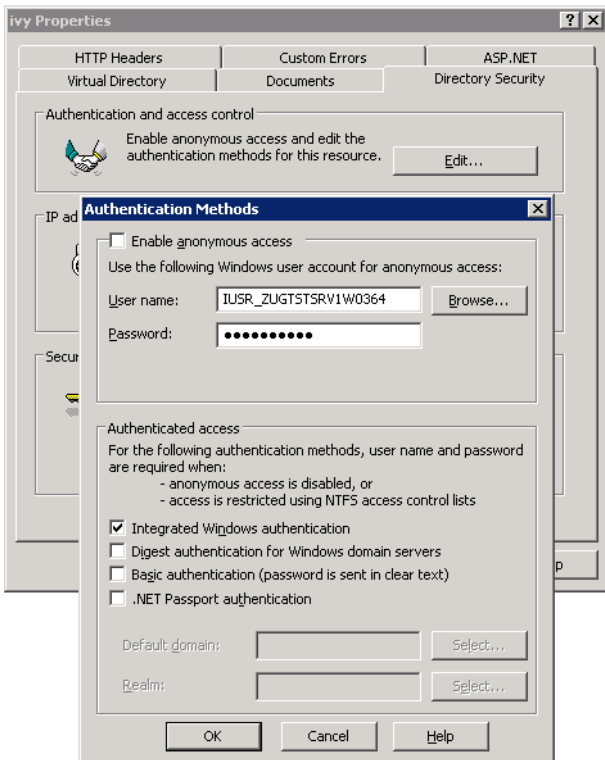
To activate the single sign on open the Internet Information Services (IIS) Manager (*Start > Control Panel > Administrative Tools > Internet Information Services (IIS) Manager*). On the left pane select the context menu *Properties* on the *ivy* Virtual Directory node. A dialog appears. On the tab *Directory Security* click on the *Edit ...* button in section *Authentication and access control*. A new dialog appears.

Select the check box *Integrated Windows authentication*. Make sure that all other authentication modes such as *Enable anonymous access* or *Digest authentication for Windows domain servers* are disabled, otherwise IIS will use those authentication modes and *Single Sign On* will not work.

Click *OK* again to close the *Properties* dialog.

To use only NTLM as security provider for the whole IIS, change into the *InetPub\Adminscripts* directory and execute following command:

```
Cscript adsutil.vbs set w3svc/NTAuthenticationProviders "NTLM"
```



IIS 7 (Windows Server 2008)



Note

There is a batch script *autoconfigSSO.bat* in the folder *misc\iis\{x64/x86}* of your engine installation. This script automatically sets up SSO on a Windows 2008 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

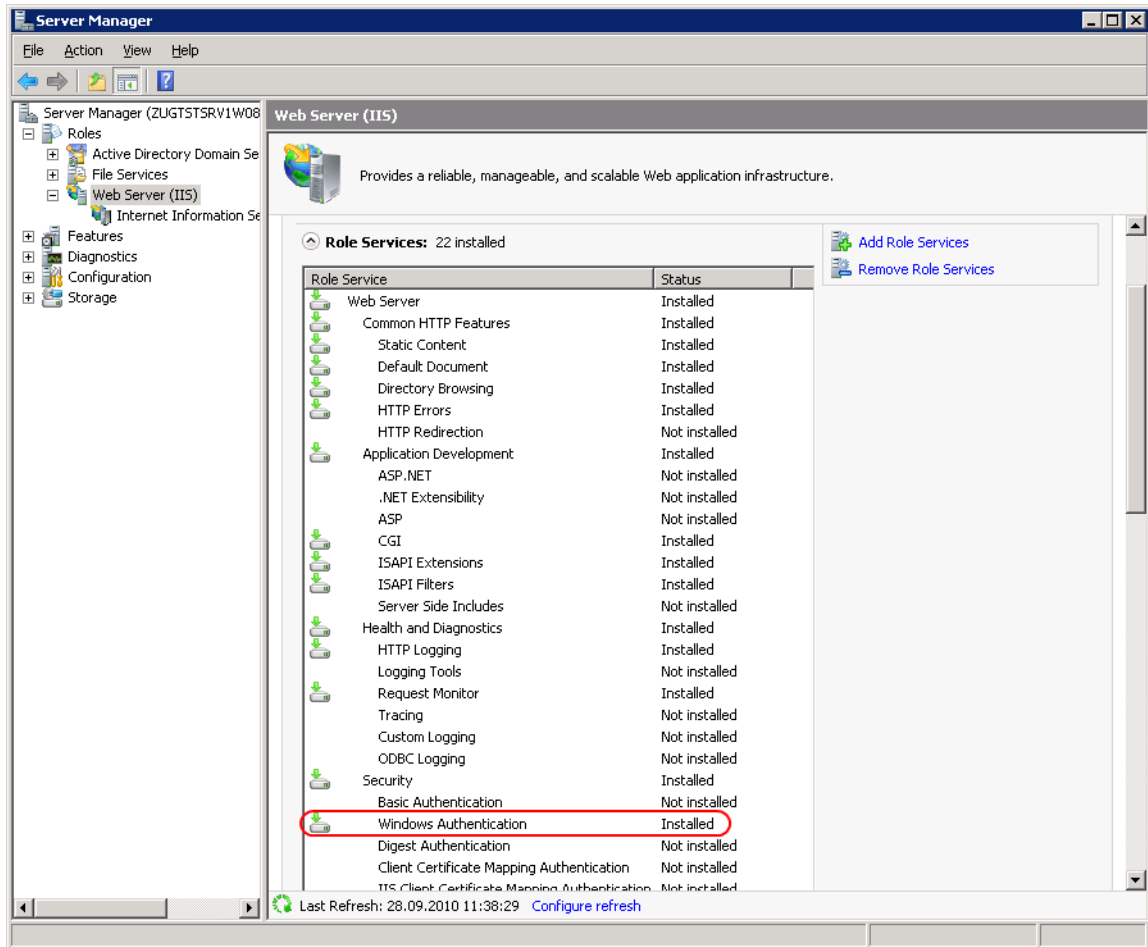


Note

The following command automatically installs the Windows Authentication:

PKGMGR.EXE /iu:IIS-WindowsAuthentication

Open the Server Manager (*Start > Control Panel > Administrative Tools > Server Manager*). Navigate to the node *Server Manager > Roles > Web Server (IIS)* and select it. 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

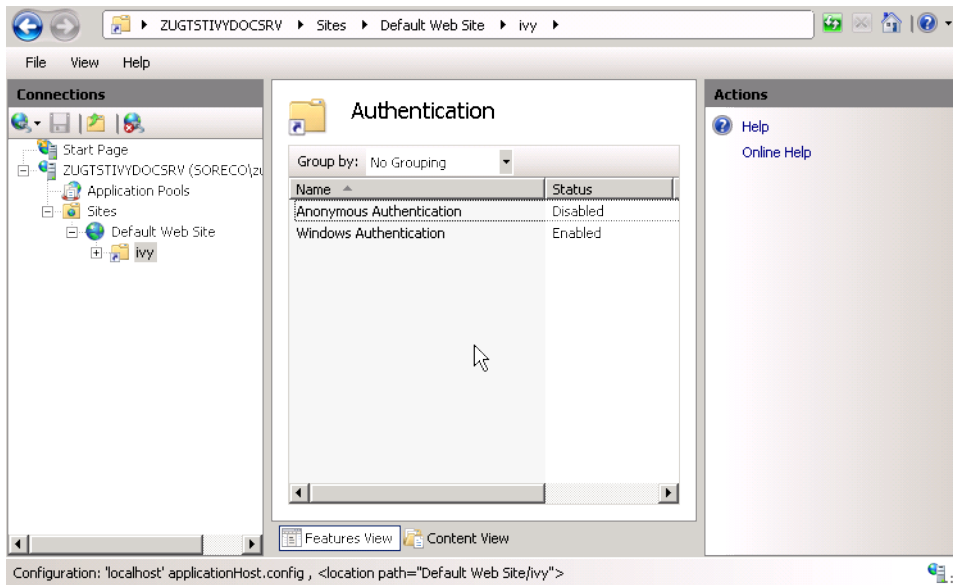
The following command automatically deactivates the Anonymous Authentication:

```
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 > Control Panel > Administrative Tools > 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

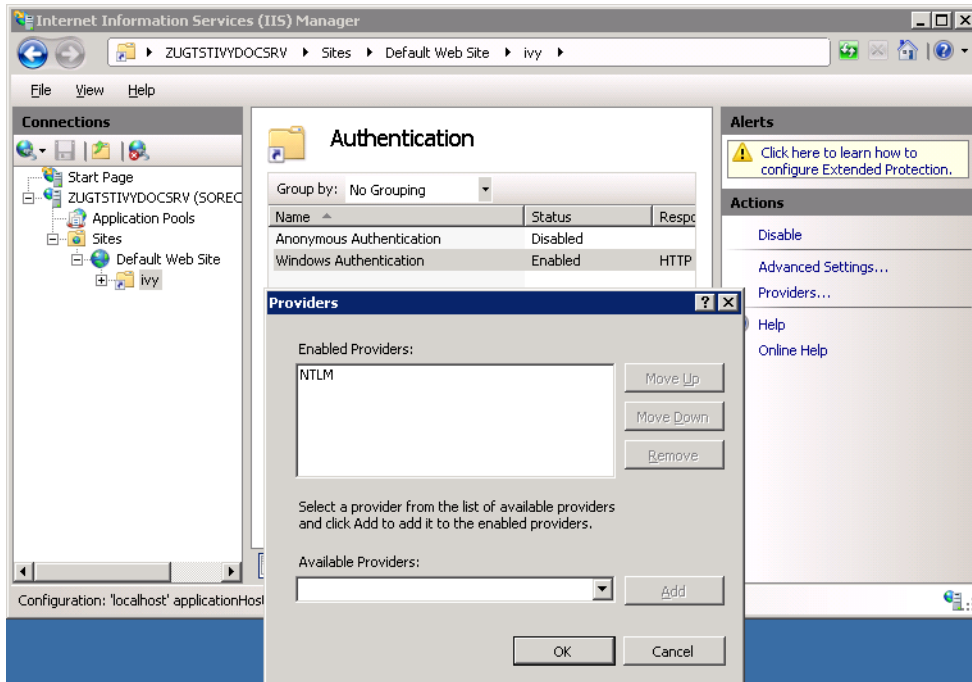
The following command automatically activates the Windows Authentication:

```
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.

IIS 7.5 (Windows Server 2008 R2)

Select the Windows Authentication and use the menu Providers ... in the Actions pane to configure the enabled providers. Remove all providers except NTLM from this list.



IIS 8 (Windows Server 2012)



Note

There is a batch script *autoconfigSSO.bat* in the folder *misc\iis\(\x64\x86)* 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

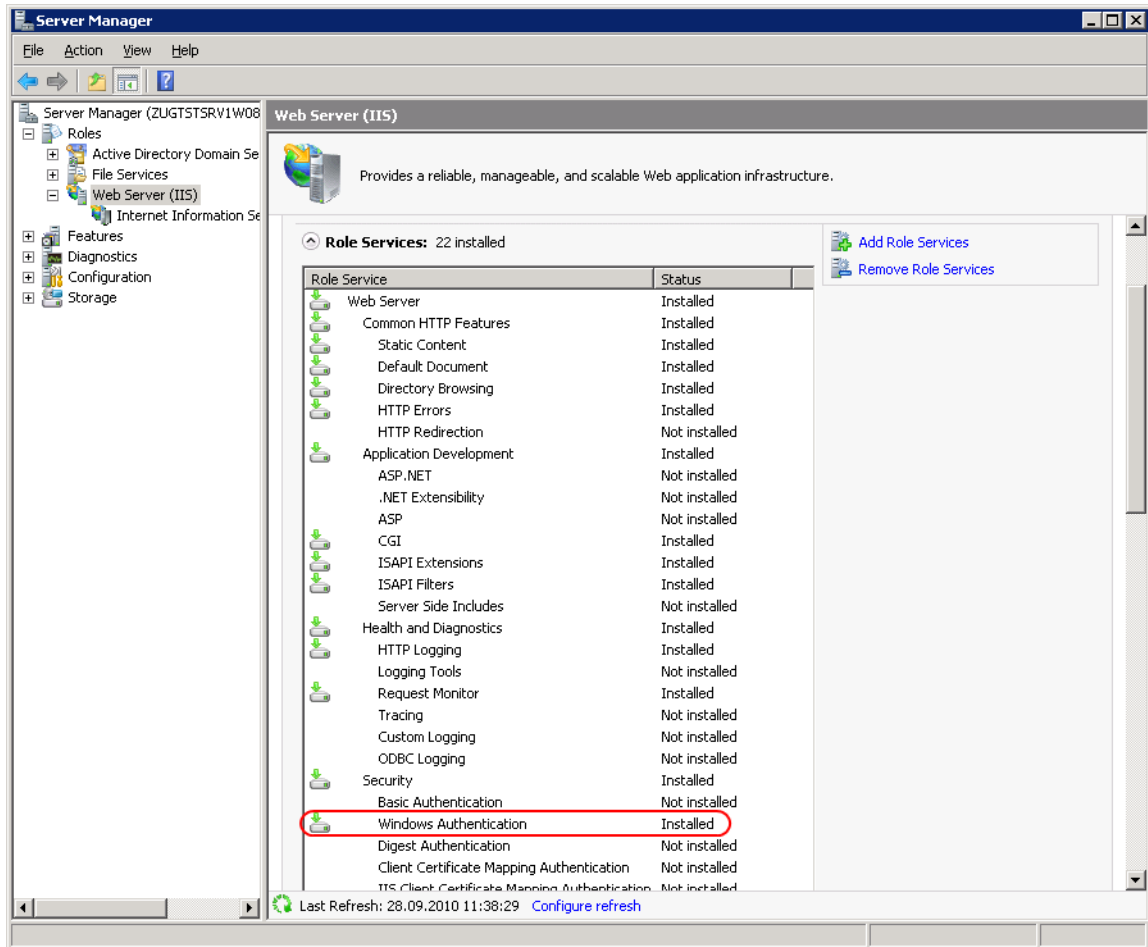


Note

The following command automatically installs the Windows Authentication:

PKGMR.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

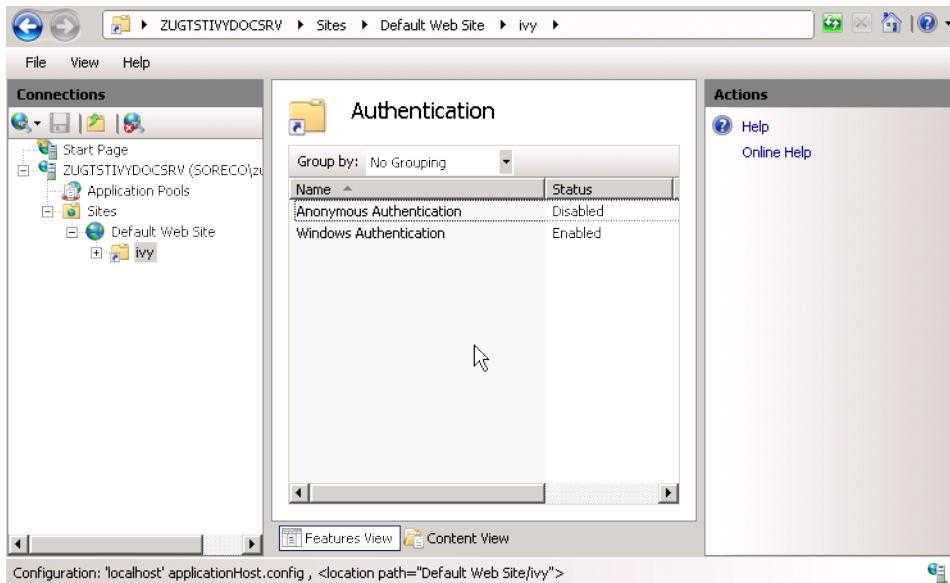
The following command automatically deactivates the Anonymous Authentication:

```
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 expect NTLM from Windows Authentication, otherwise Single Sign On may not work with the RIA clients.

Troubleshooting

See “Request Entity Too Large”

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_loadbalancer_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. At the top, there are navigation tabs for 'Server', 'Admins', and 'Cluster'. Below the tabs is a table titled 'Cluster Nodes' with the following data:

Name	Host	State	Communication	Is Master	Is Local	Start Time
ZUGPCFS.axonivy.wan1	ZUGPCFS.axonivy.wan1	RUNNING	UP	yes	yes	05.09.14 11:20
ZUGPCFS.axonivy.wan2	ZUGPCFS.axonivy.wan2	RUNNING	UP	no	no	05.09.14 11:20

A red circle highlights the name 'ZUGPCFS.axonivy.wan1' in the table, with a red arrow pointing to it labeled 'click'. Below the table, the 'Cluster Node Detail single node' page is displayed for the selected node. The 'JVM Route' field is highlighted with a red circle and a red arrow pointing to it, showing the value 'zugpcfs_axonivy_wan1'.

Figure 4.2. 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:

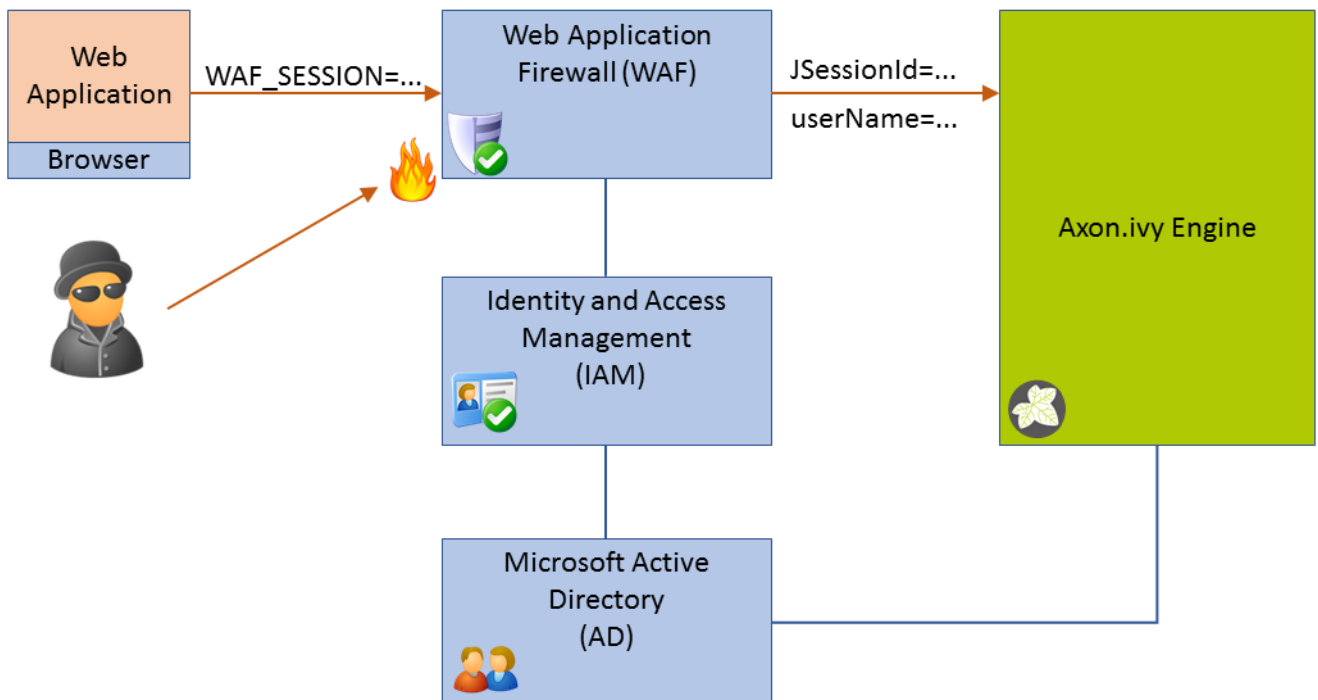


Figure 4.3. Single Sign On Infrastructure using a Web Application Firewall, Identity and Access Management and Active Directory

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 `webapps/ivy/META-INF/context.xml` from the Axon.ivy Engine installation directory and uncomment the following line:

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

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:

```
@Override
public void invoke(Request request, Response response) throws IOException, ServletException {
    String userName = getUserFromRequest(request);
    if (StringUtils.isBlank(userName))
    {
        getNext().invoke(request, response);
        return;
    }
    Principal userPrincipal = createUserPrincipalWith(userName);
    request.setUserPrincipal(userPrincipal);
    getNext().invoke(request, response);
}
```

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

Chapter 5. Administration

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:

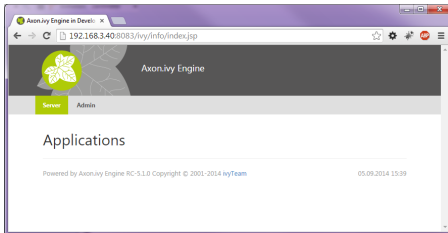


Figure 5.1. Axon.ivy Engine info page

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.

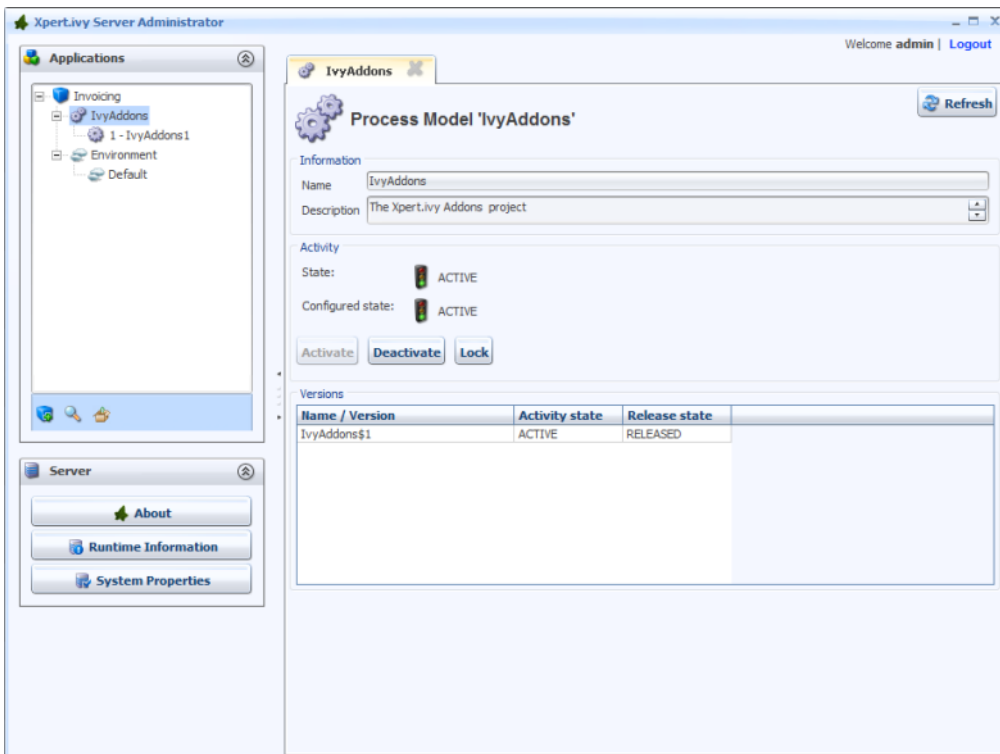



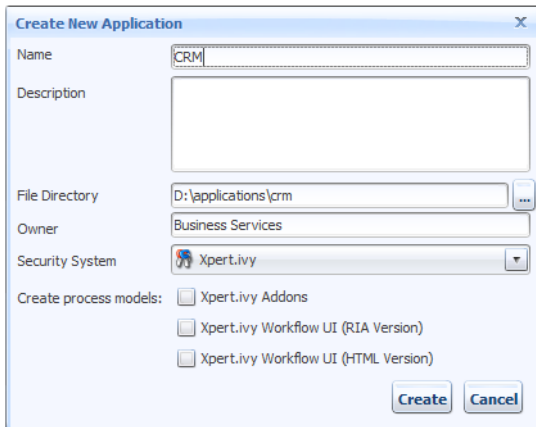
Figure 5.2. Admin Tool

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

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).

Configuring an External Security System

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 exist 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 exist 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 synchronization is executed:

- Once a day (see synchronization time).
- If one clicks 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.

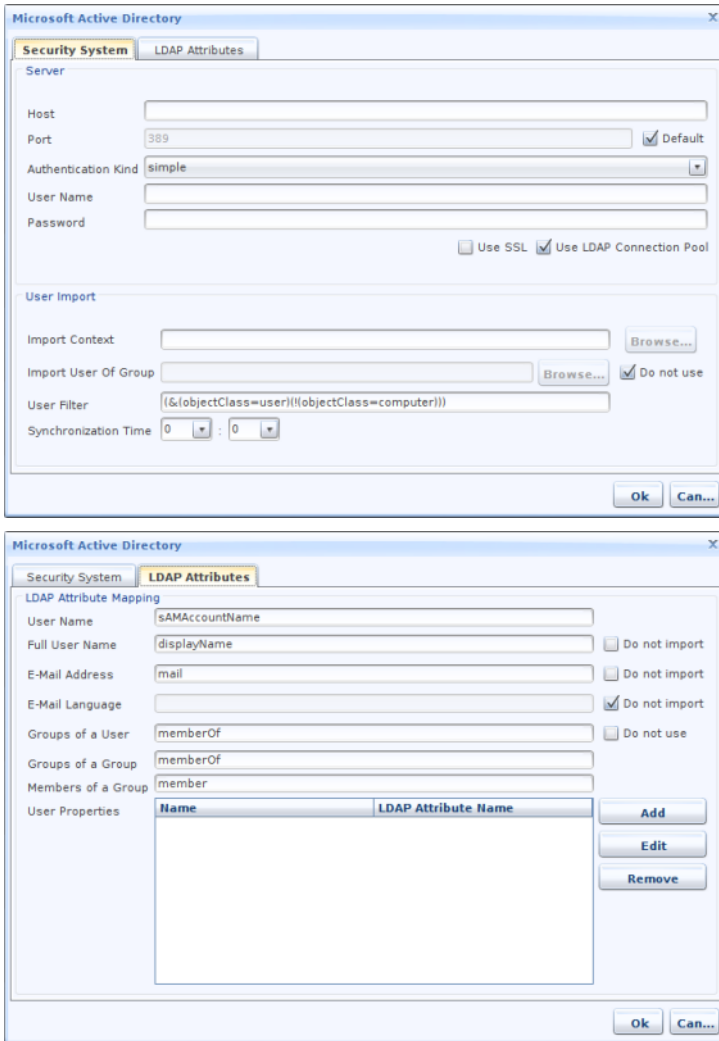


Figure 5.3. External security system configuration dialog

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. <p>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 the JRE keytool:</p>

```
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

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).

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.

Use the **Browse** button to select the import context from the server specified above (if the connection fails, check the connection data).

Import users of group

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.

Use the **Browse** button to select the user group from the server specified above (if the connection fails, check the connection data).

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

Default attribute mappings can be adjusted.

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.

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

The button **configure default settings** on the application configuration screen leads to the *Application Default Settings*.

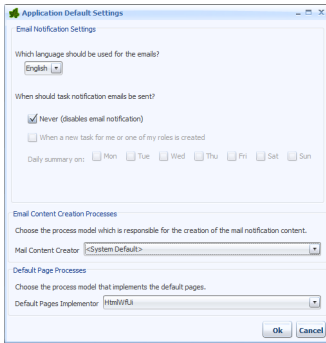


Figure 5.4. Application Default Settings

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. Notification processes start elements has to conform the criteria described in the section Email Notification.



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.

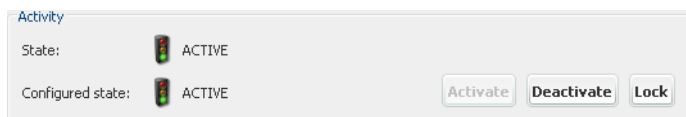


Figure 5.5. Managing the activation state

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.

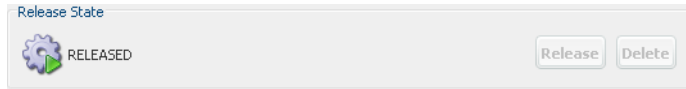


Figure 5.6. Managing the release state

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 5.1. 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 5.2. 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.

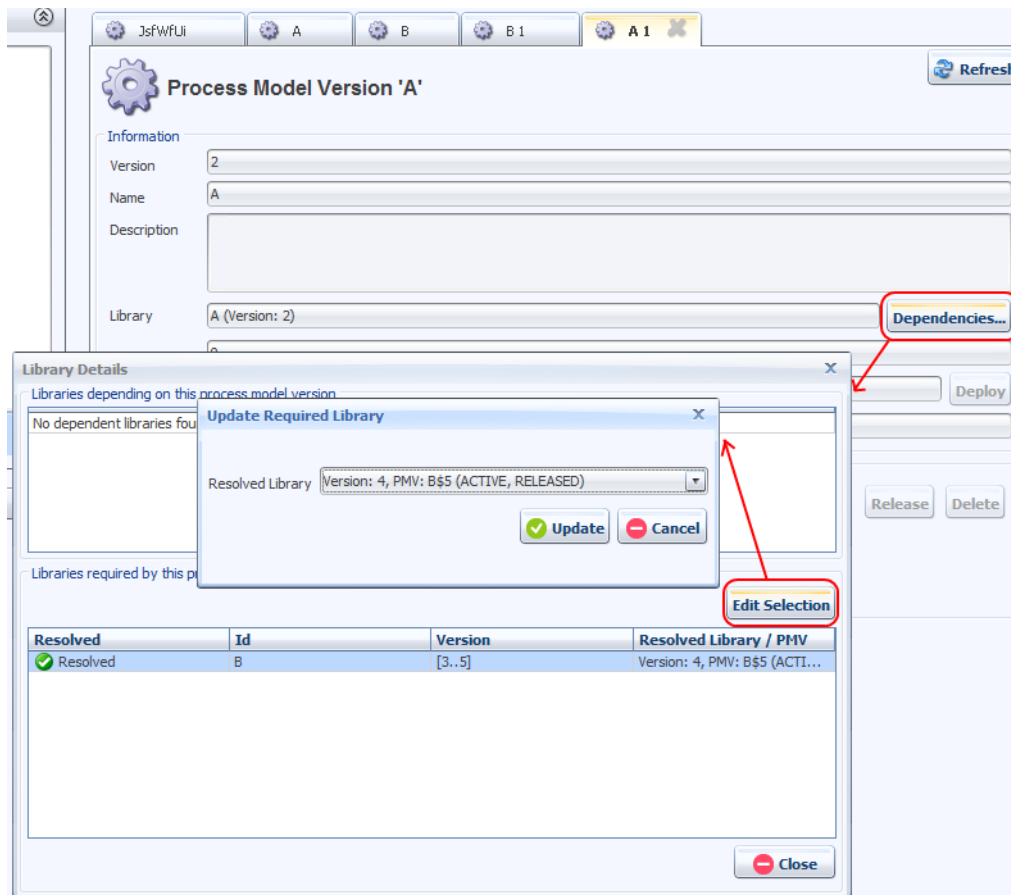


Figure 5.7. Project Dependencies

Project Deployment

Project deployment means to take a project created in the Axon.ivy Designer and to load it into a process model version, so that you can execute the project on the engine.

Before you begin with the deployment of a project make sure that you also have all required projects either already deployed to the engine or available in the same folder or in the same *.zip file as the major project.



Warning

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 project version (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 wizard

The deployment wizard can be started on the toolbar.

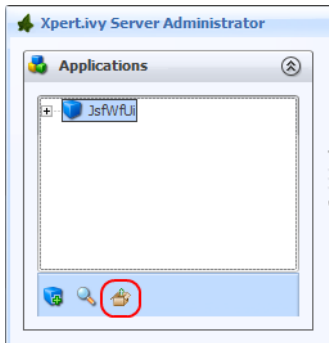


Figure 5.8. Command to Start the Deployment Wizard on the Toolbar

On the first page of the wizard select the projects that you want to 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 folder. 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 be 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.

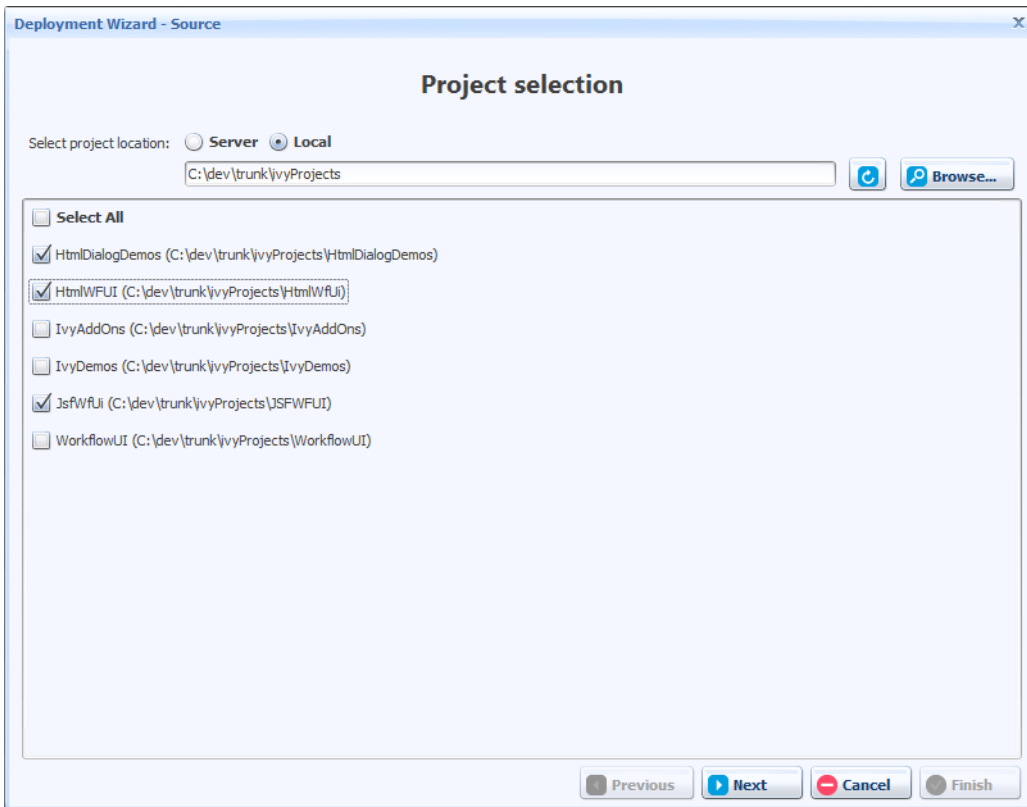


Figure 5.9. Deployment Wizard - Source

On the second page of the wizard select the applications to which you want to deploy your projects and press **Next**.

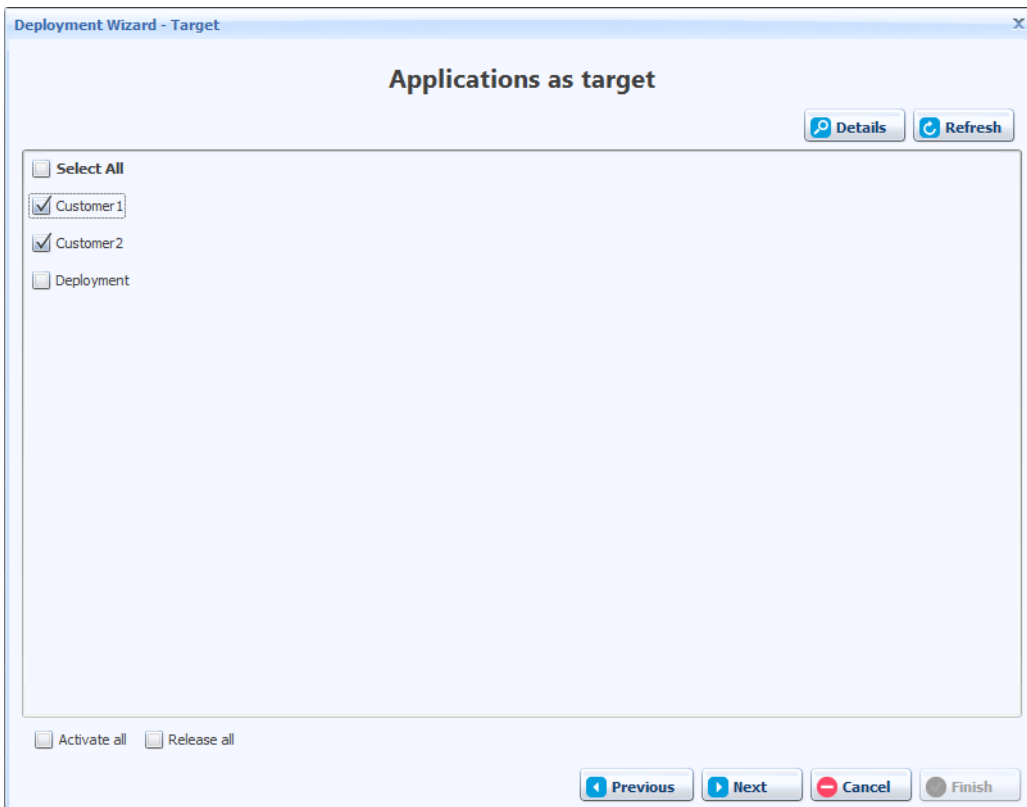


Figure 5.10. Deployment Wizard - Target

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.

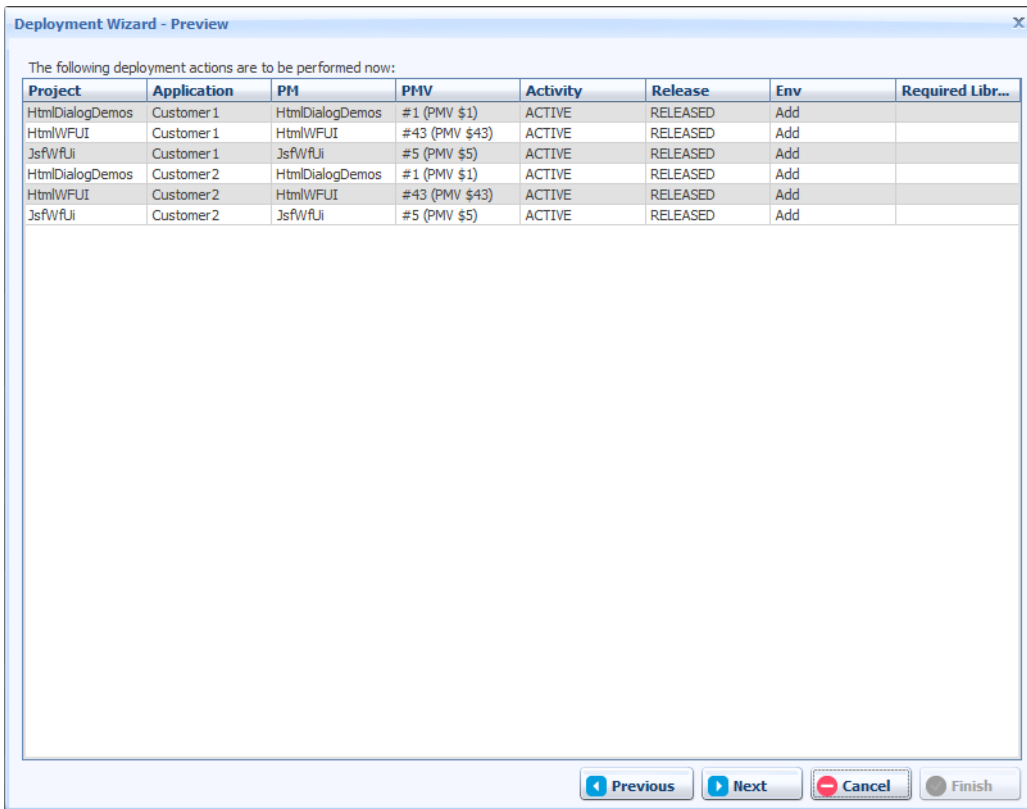


Figure 5.11. Deployment Wizard - Preview

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.

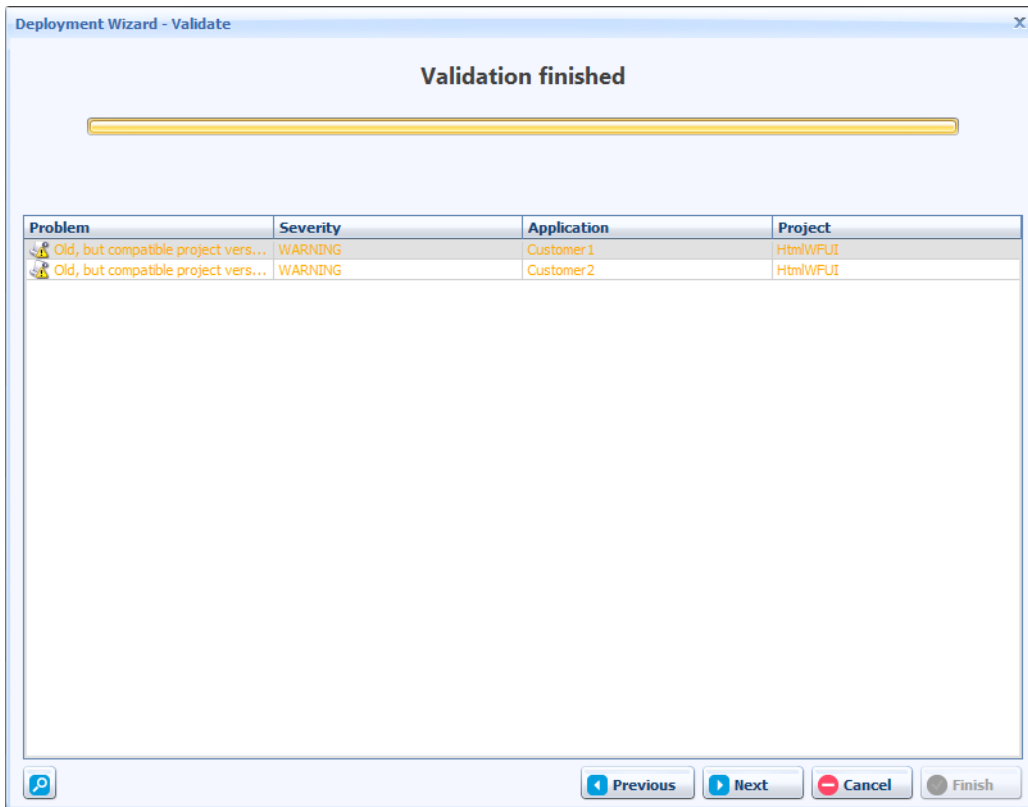


Figure 5.12. Deployment Wizard - Validate

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.

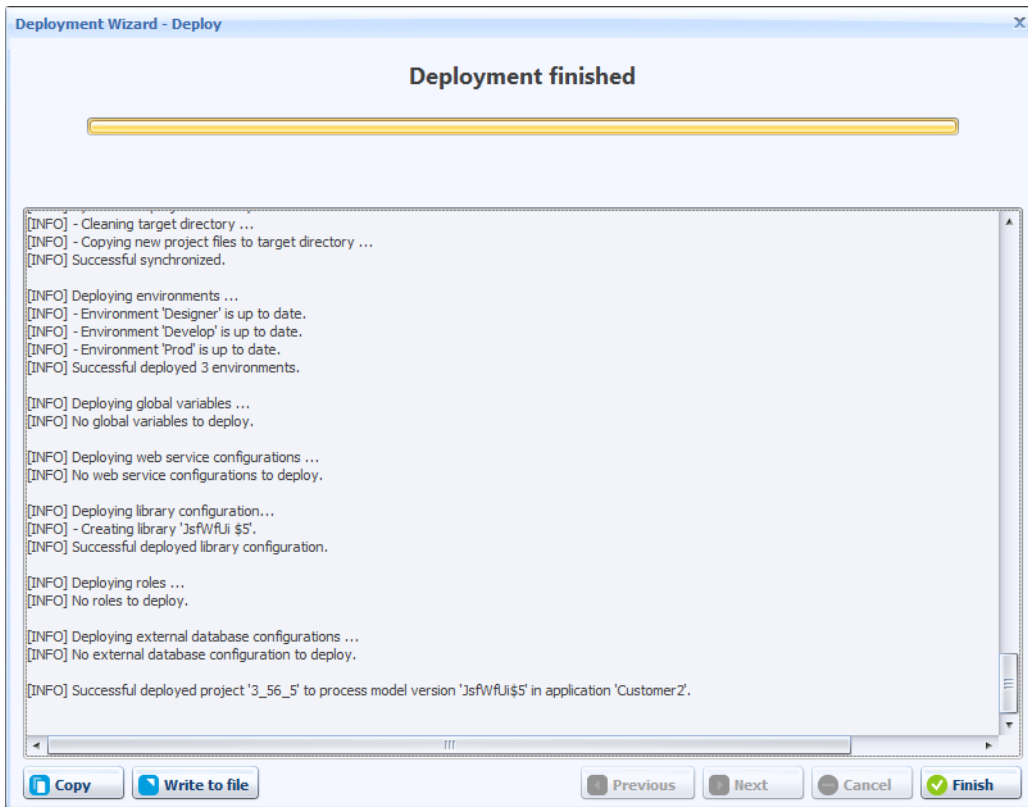


Figure 5.13. Deployment Wizard - Deploy

Deployment directory

Makes simple deployment of Axon.ivy projects to an Axon.ivy Engine possible by copying the project into the deployment directory.

The deployment directory is by default the directory with the name 'deploy' in the Axon.ivy Engine root directory. It can be altered through the system property 'Deployment.Directory'.

File based deployment

1. Ensure that the Axon.ivy Engine, to which the Axon.ivy project should be deployed, is up and running.
2. Navigate to the sub directory in the deployment directory which matches the name of an already created Application on the Axon.ivy Engine.
3. Copy an Axon.ivy project to deploy into the created application directory. The project must either be packed as ivy-archive (IAR) or ZIP or be contained in a file directory.
4. Initialize the deployment by creating a new empty file with the name of the project file or directory plus the suffix '.doDeploy'. E.g. If you the project /deploy/MyApp/JsfWorkflowUi.iar has been copied in the previous step, the creation of the file /deploy/MyApp/JsfWorkflowUi.iar.doDeploy initializes the deployment.
5. Wait until the file with the suffix '.doDeploy' disappears, it indicates that the deployment has finished.
6. Determine whether the deployment was successful. If no file with suffix '.deploymentError' was created, the deployment has finished successfully. The detailed log is available in the file with suffix '.deploymentLog'.

File suffix	Description
.doDeploy	Initializes the deployment and is deleted when the deployment process has finished
.deploymentLog	Contains the log output which is written during the deployment

File suffix	Description
.deploymentError	Contains the error cause and is only written when the deployment fails

Table 5.3. Deployment marker files

Maven deployment

The Maven project-build-plugin makes automated continuous deployment to an Axon.ivy Engine possible.

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

Command line deployment

The 'deploy-iar' goal can be executed 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:6.1.0-SNAPSHOT:deploy-iar -Divy.deploy.iarFile
```

Build goal execution

To deploy an ivy-archive (IAR) during its Maven build lifecycle configured an execution which binds the 'deploy-iar' 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-iar</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.

Business Calendar

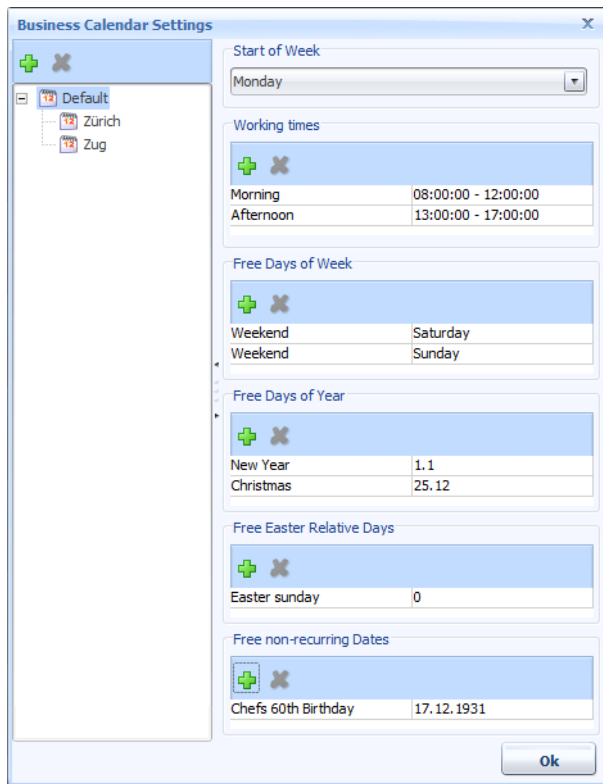
A business calendar defines the following three things:

- 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

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 ore 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**

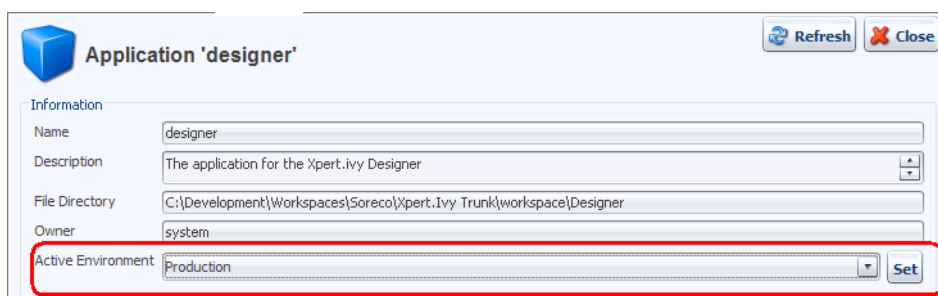


Figure 5.14. Set active environment for an application

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.

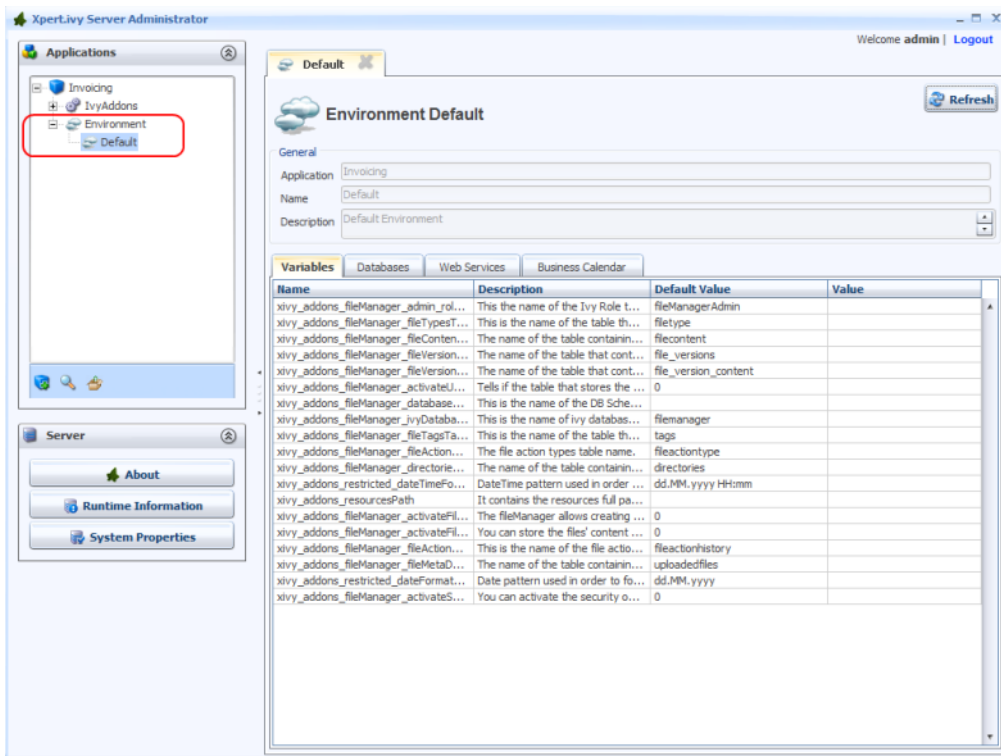


Figure 5.15. Environment details

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.

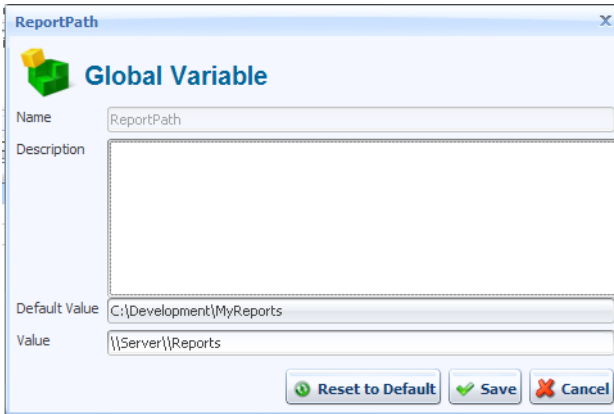


Figure 5.16. Details of a global variable

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**.

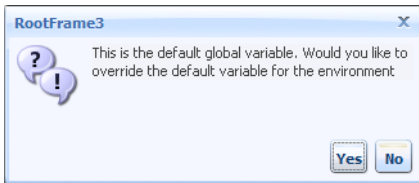




Figure 5.17. Question to override the default value of the global variable

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**.

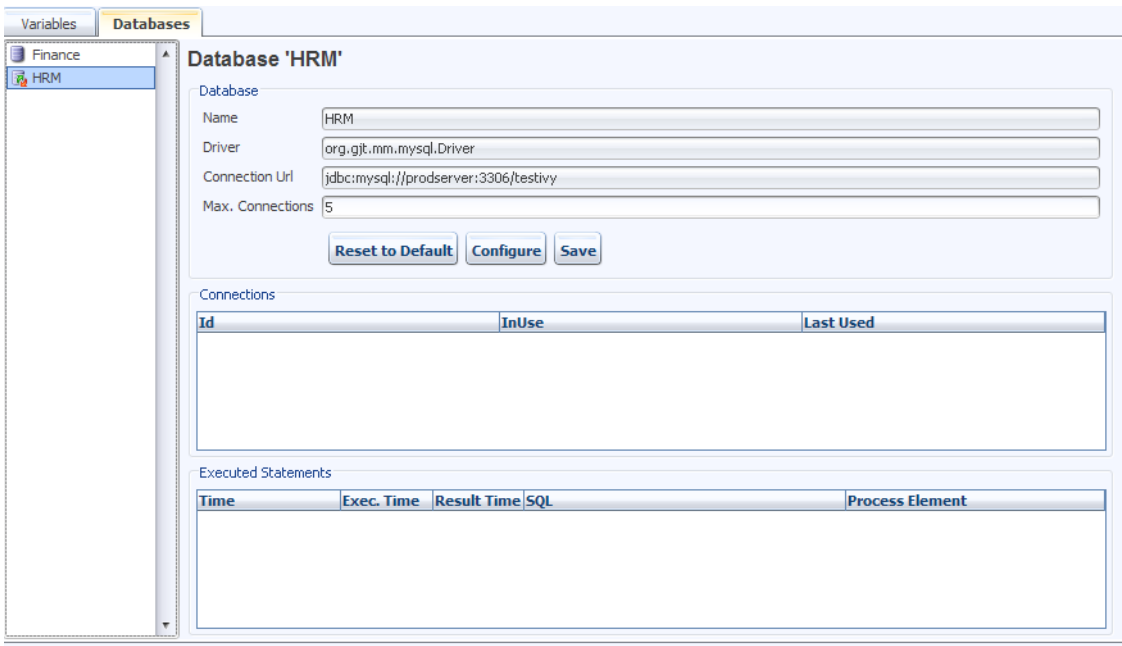


Figure 5.18. Databases

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.

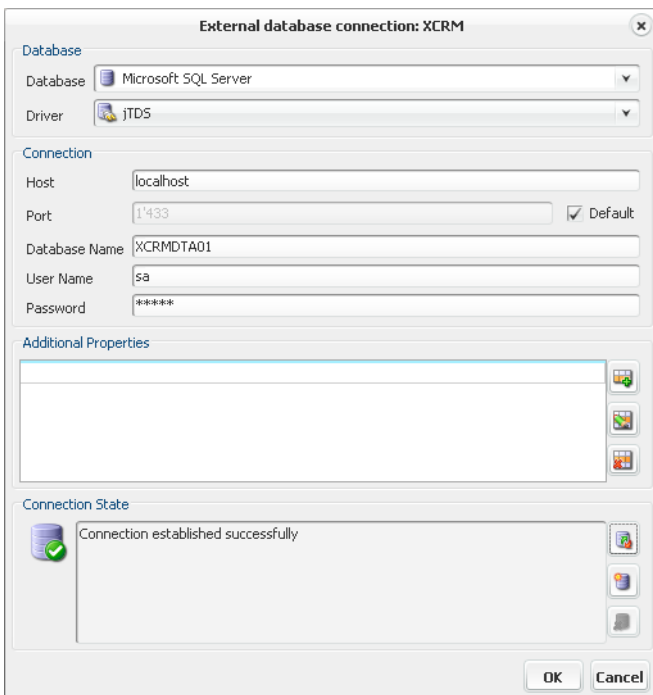


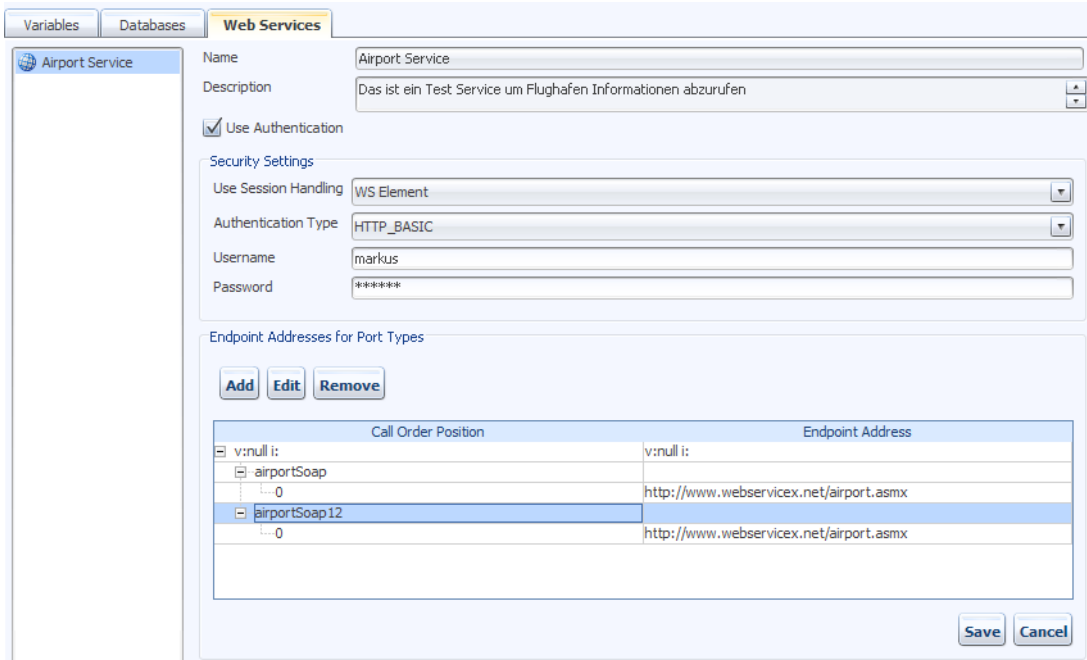


Figure 5.19. Configuring the connection of an external database

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 .

The screenshot shows the 'Rest Client Details' configuration window. On the left, a search bar and a list of REST Clients are visible, with 'twitter' selected. The main panel displays the following details:

- UUID:** 066a1b8c-f787-4540-9496-f1f2ecc9a1b1
- Name:** twitter
- Description:** (empty)
- URI:** https://api.twitter.com/1.2
- Authentication:** Type: HTTP Basic HTTP Digest; Username: (empty); Password: (empty)
- Feature:** JSON
- Class:** com.axonivy.connectivity.rest.client.handler.GsonMessageBodyHandler (Add), com.axonivy.connectivity.rest.client.filter.TwitterOAuthFilter (Remove)
- Properties:**

Name	Value
consumerSecret	*****
consumerKey	y3B1HDgeXsBh2oJdrFd6ji2fR

Buttons at the bottom include 'Set to default', 'Save', 'Cancel', 'Add', 'Remove', and 'Add Password'.


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

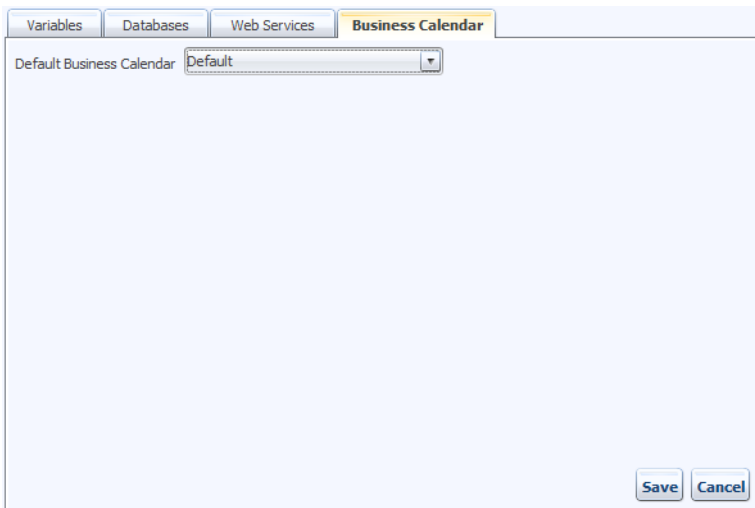
The name of the REST Client. The REST Client can be referenced by this name. Can be modified. Note that references using the name will break if you change it.

Description	Description of the REST Client.	
URI	<p>The base URI under which the remote service publishes its resources (e.g. <code>https://api.twitter.com/1.1</code>).</p> <p>The URI can contain template placeholders which are resolved later by the client user (e.g. <code>https://api.twitter.com/{version}</code>).</p> <pre>ivy.rest.client("twitter").resolveTemplate("version", "1.1").get()</pre>	
		<p>Tip</p> <p>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: <code>{ivy.engine.host}</code>, <code>{ivy.engine.http.port}</code>, <code>{ivy.engine.context}</code>, <code>{ivy.request.application}</code>.</p> <p>E.g. <code>http://{ivy.engine.host}:{ivy.engine.http.port}/{ivy.engine.context}/api/{ivy.request.application}/my/service</code></p>
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	<p>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>.</p> <p>The properties configured here are registered in the <code>WebTarget</code> using the method <code>property(String, Object)</code>.</p>
	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

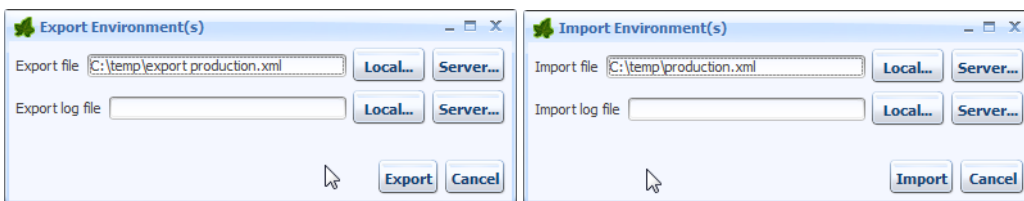
Each environment can define its own default business calendar which is going to be used to calculate time periods in working time if the environment is active. By default the global default business calendar is set.

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.

The screenshot shows a 'Security System' configuration panel. It contains three input fields: 'Name' with the value 'Xpert.ivy', 'Users' with the value '2', and 'Roles' with the value '1'. To the right of the 'Name' field are 'Edit ...' and 'Synchronize' buttons. To the right of the 'Users' field is a 'Show ...' button. To the right of the 'Roles' field is a 'Show ...' button.

Figure 5.20. 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.

The screenshot shows a 'Users' dialog box with a table of users and a list of action buttons. The table has columns for Name, Full Name, and E-Mail Address. The buttons include Create ..., Edit ..., Delete ..., Roles ..., Properties ..., Permissions ..., System Permissions ..., and Ok.

Name	Full Name	E-Mail Address
testuser1	Test User 1	testuser1@ivyteam.com
testuser2	Test User 2	testuser2@ivyteam.com

Figure 5.21. 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 5.22. 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.

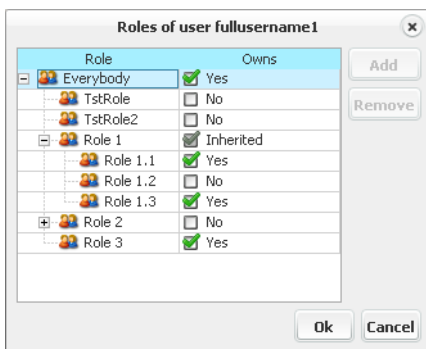


Figure 5.23. Manage roles of a user

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.

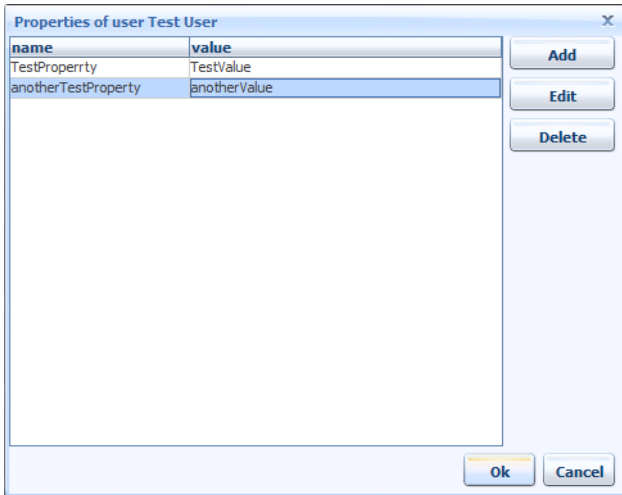


Figure 5.24. User properties

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.

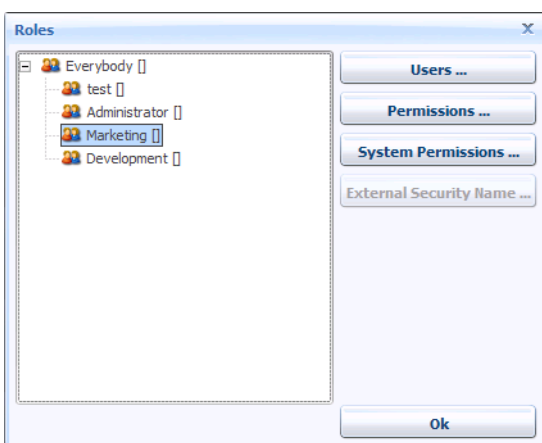


Figure 5.25. List of all roles

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 5.26. 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.

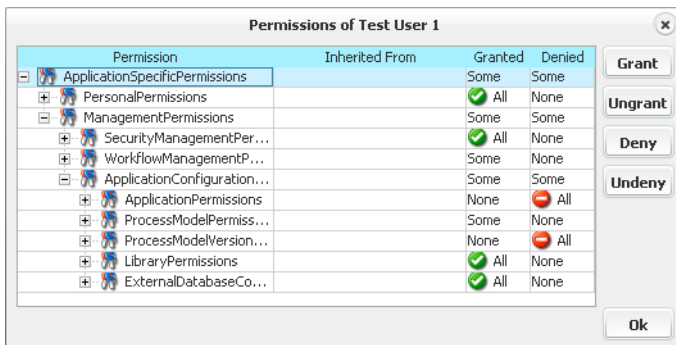


Figure 5.27. Editing the permissions of a user

System Properties

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.

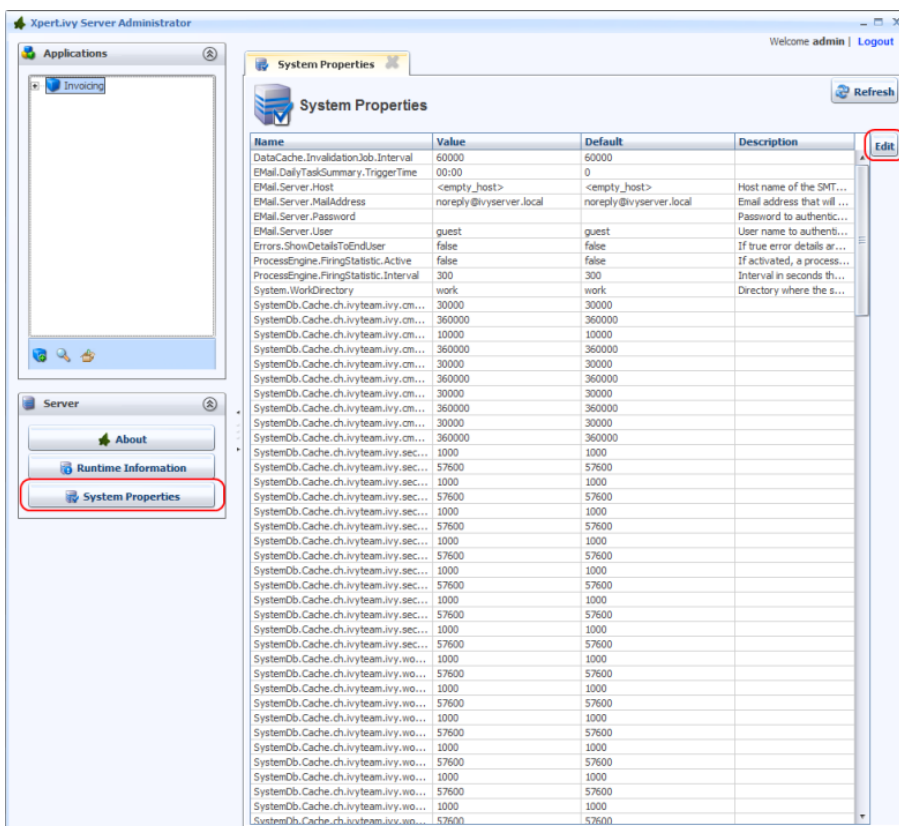


Figure 5.28. Overview of system properties

To change any properties, select the corresponding row in the table and press the **Edit** button or simply double-click on the row.

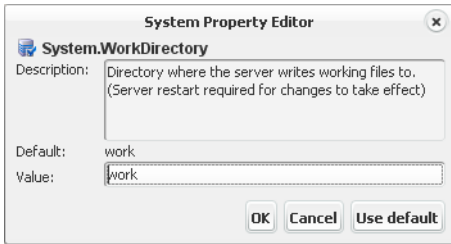


Figure 5.29. Editing a system property



Note

Some settings may not take effect until you restart the engine.

Data Cache

Use the system property `DataCache.InvalidationJob.Interval` to configure the time in seconds between two executions of the job that invalidates expired data cache entries.

Email Server

Use the system properties `EMail.Server.*` to configure a SMTP mail server that is used to send mails. The available properties are similar to the Email preferences of the designer. For more information see [Designer Guide > Introduction > Axon.ivy Preferences \(Workspace Preferences\) > Email Settings](#).

Errors

Use the system property `Errors.ShowDetailsToEndUser` to configure whether details about an error should be shown to an end user. For security reasons no details should be shown. For debugging reasons this option could be turned on temporary. For more information see [Show Error Details to End Users](#).

Process Engine Firing Statistic

Use the system properties `ProcessEngine.FiringStatistic.*` to configure the process element performance statistic. For more details see [Process Element Performance Statistic and Analysis](#).

SSL Client

Use the system properties `SSL.Client.*` to configure the key and trust store used for the client side of SSL connections. The available properties are similar to the SSL Client preferences of the designer. For more information see [Designer Guide > Introduction > Axon.ivy Preferences \(Workspace Preferences\) > SSL Client Settings](#).

System Work Directory

Use the system property `System.WorkDirectory` to configure the directory where temporary work files are stored.

System Database Cache

Use the system properties `SystemDb.Cache.*` to configure how many and how long entities from the system database are cached in memory.

System Tasks

Use the system property `SystemTask.SearchJob.Interval` to configure the time in seconds between two executions of the System Task Search Job. The job searches system tasks that were not executed because of failures and initiate their execution.

Use the system property `SystemTask.Failure.Behaviour` to configure the behaviour in case a System Task execution fails. A system task is a task that is executed by the system.

Thread Pool

Use the system property `ThreadPool.*.CorePoolSize` to configure the number of threads that are at least used in the thread pool.

Use the system property `ThreadPool.*.MaximumPoolSize` to configure the maximum number of threads that are used in the thread pool. This property is not available for fixed sized thread pools.

Update Checker

Use the system property `UpdateChecker.Enabled` to configure if the update checker is enabled or disabled. The update checker checks if a new update release is available. It also sends some statistic information to the vendor. For more details see Update Notification.

Web Server

Use the system properties `WebServer.AJP.*` to configure the AJP protocol used for the communication between IIS or Apache web server and the Axon.ivy Engine. For more information see Integration.

Use the system properties `WebServer.HTTP.*` to configure the HTTP protocol used for the communication with Web Browsers and Rich Dialog clients.

Use the system properties `WebServer.HTTPS.*` to configure the HTTPS protocol used for the communication with Web Browsers and Rich Dialog clients.

Use the system properties `WebServer.External*` and `WebServer.IvyContextName` to configure the external host name, port and protocol used to provide links in mails. For more information see Update external base URL.

Use the system property `WebServer.WellKnownServerPorts` to configure additionally well known external web servers that forwards requests to Axon.ivy engine. Format is `host:protocol=port, host:protocol=port, ...`. Example: `developer.axonivy.com:http=80, developer.axonivy.com:https=443`.

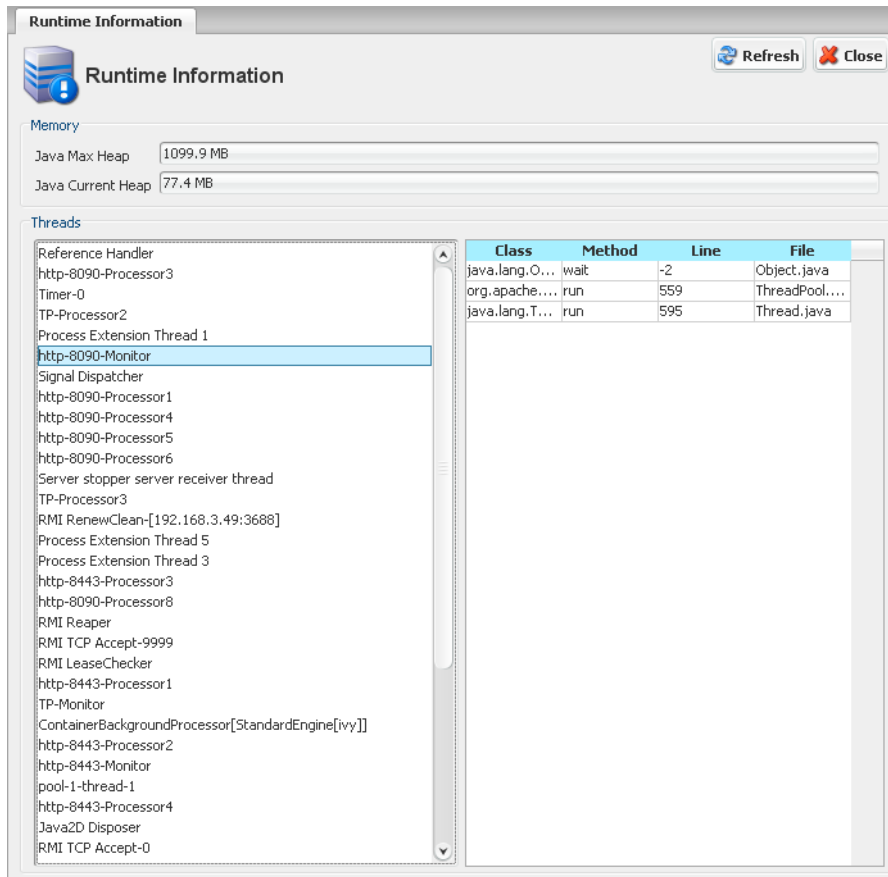
Elasticsearch configuration

Use the system properties `Elasticsearch.*` to configure the bundled or external Elasticsearch server installation. For more details, see Elasticsearch installation.

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.



The screenshot shows a 'Runtime Information' window with a title bar containing a refresh icon and a close button. The window is divided into two main sections: 'Memory' and 'Threads'.

Memory Section:

- Java Max Heap: 1099.9 MB
- Java Current Heap: 77.4 MB

Threads Section:

A list of threads is shown on the left, with 'http-8090-Monitor' selected. The right pane displays a table of the selected thread's stack frames.

Class	Method	Line	File
java.lang.O...	wait	-2	Object.java
org.apache....	run	559	ThreadPool....
java.lang.T...	run	595	Thread.java

Figure 5.30. 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.

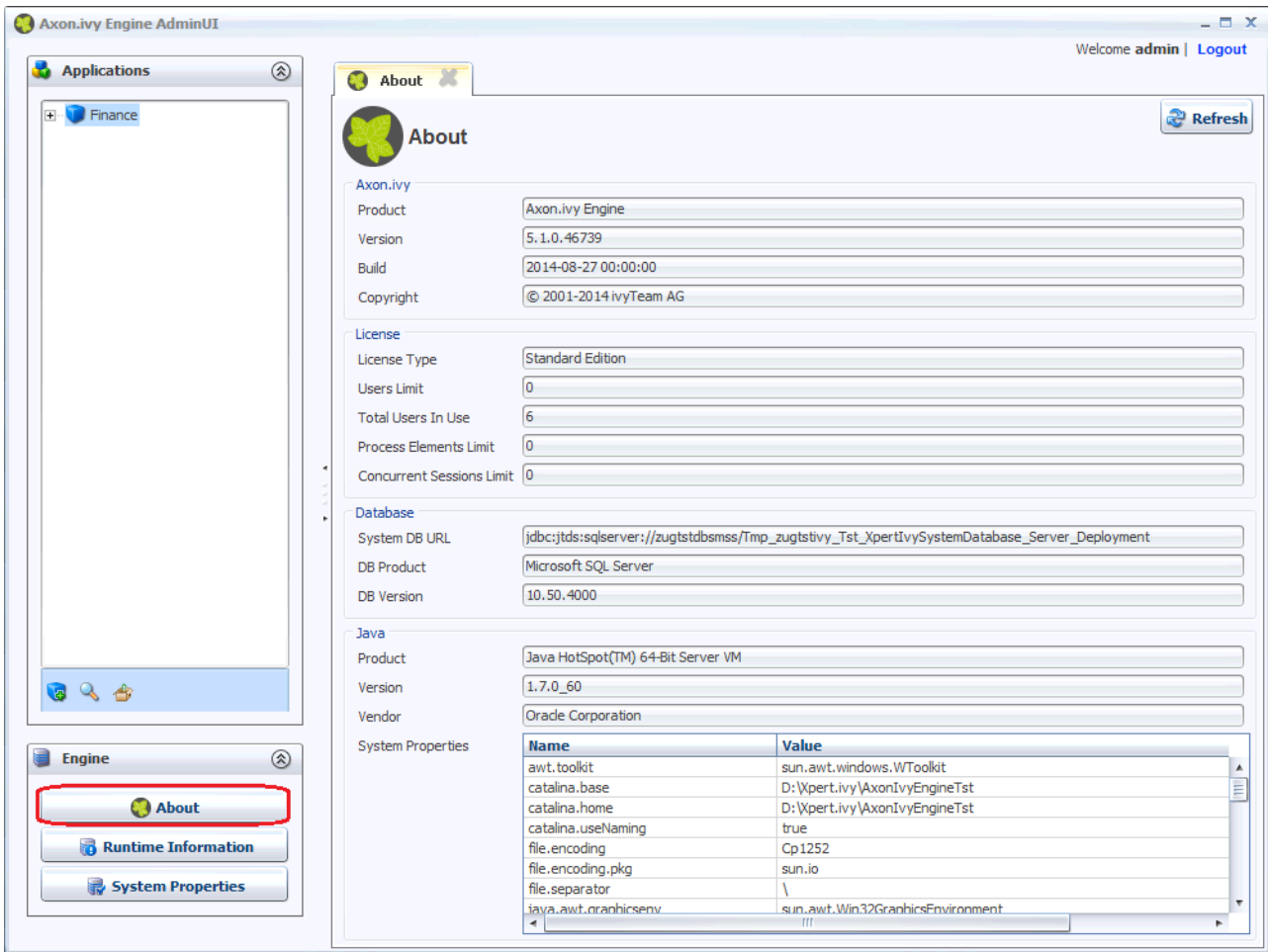


Figure 5.31. About

HTML Workflow UI

Axon.ivy does not provide a User Interface to manage tasks. There are APIs to access and manipulate the tasks of an application. But the user interface to access the tasks must be implemented according to the needs of the application.

This chapter describes the possibilities which can be used in a HTML based Axon.ivy web application. For RIA based task management please consult the RIA Workflow UI documentation.

Replacement Project

In a HTML based workflow application there are certain pages displayed depending on the state of your application and/or workflow. You can override these default pages for each of your applications, to adopt it to your needs.

For each kind of page there is a predefined process start signature, which allows you to render the page in your own process instead of using the default page. The following table shows which signature is needed to override each type of page.

Page	Description	Process Start Signature
Application Home	Home Page of the application.	DefaultApplicationHomePage()
Task List	Contains a list of all Tasks the currently logged in user can work on.	DefaultTaskListPage()
Process Start List	Contains a list of all Processes the currently logged in user can work on.	DefaultProcessStartListPage()

Page	Description	Process Start Signature
End	Is displayed whenever a Task or Process ends	DefaultEndPage(Number endedTaskId)
Login	Is displayed whenever an authenticated user is required and none is available	DefaultLoginPage(String requestedPage)

Table 5.4. Default Pages

How to override the default pages:

1. To override the pages create one project containing all processes with the overriding signatures.
2. Deploy this project to the application for which you want to override the behaviour.
3. Configure the application to use the deployed project to override the default pages. For more details see Application Default Settings.

Application Home Page replacement

By default Axon.ivy shows a page which simply says hello to the user.

Use a Request Start with the signature *DefaultApplicationHomePage()* to overwrite this behaviour.

Start Signature

Signature `DefaultApplicationHomePage()`

Name `DefaultApplicationHomePage`

Definition of the input parameters

Name	Type

This Request Start must either lead to an Web Page Element or an End Page Element.



Task List Page replacement

By default this page displays all tasks the currently logged in user can work on. You can override this page to adopt it to your look and feel or to display your own list of tasks.

Use a Request Start with the signature *DefaultTaskListPage()* to overwrite this behaviour.

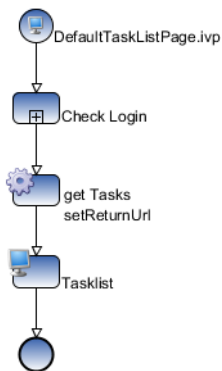
Signature DefaultTaskListPage()

Name DefaultTaskListPage

Definition of the input parameters + -

Name	Type

This Request Start must either lead to an Web Page Element or an End Page Element.



Process Start List Page replacement

By default this page displays all Processes the currently logged in user can work on. You can override this page to adopt it to your look and feel or to display your own list of process starts.

Use a Request Start with the signature *DefaultProcessStartListPage()* to overwrite this behaviour.

Start Signature

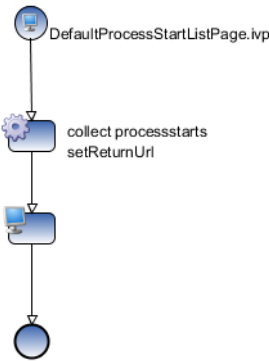
Signature DefaultProcessStartListPage()

Name DefaultProcessStartListPage

Definition of the input parameters + -

Name	Type

This Request Start must either lead to an Web Page Element or an End Page Element.



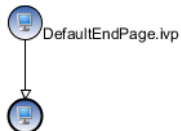
End Page replacement

By default Axon.ivy will show a page which says "You have successfully finished the task", when a task is finished and the process does not define a specific End Page.

Use a Request Start with the signature *DefaultEndPage(Number endedTaskId)* to overwrite this behaviour

Start Signature	
Signature	DefaultEndPage(Number)
Name	DefaultEndPage
Definition of the input parameters + -	
Name	Type
endedTaskId	Number

This Request Start must either lead to an Web Page Element or an End Page Element.



Login Page replacement

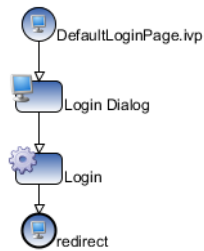
By default Axon.ivy will show a simple login page, whenever a login is required. The page allows you to login with username and password and then redirects you to the original URL requested.

Use a Request Start with the Signature *DefaultLoginPage(String requestedPage)* to overwrite this behaviour

Start Signature	
Signature	DefaultLoginPage(String)
Name	DefaultLoginPage
Definition of the input parameters + -	
Name	Type
requestedPage	String

This Request Start must lead to an activity which does login a user. Normally this is done using a dialog which asks for username and password. But you are free to do anything to identify the user.

Normally the process should lead to an End Page which redirects to the original requested url. But you are free to do anything after the user is logged in.



Sample HTML Workflow UI Project

Axon.ivy Engine is delivered with a sample HTML Workflow UI project. You can find it in the installation directory in the folder *projects/HtmlWfUi*.



Tip

We suggest to adopt the sample project to your needs and use it as described above instead of creating a new project from scratch.

If you want to use the sample project as it is, consult the section create a new application.

Email Notification

Overview

Axon.ivy Engine can send out email notifications on user task changes. These emails contain information on tasks that are available for users to work on. Notification subscriptions are configurable and the email content can be highly customized. In order to understand how it generally works it is recommended to read this section. This feature works only on engine versions, not in the Axon.ivy Designer. There are two types of notification emails:

New Task	One email is sent when a new task is available (created, inherited, assigned) for a given user
Daily Summary	One email is sent daily with a summary of all open tasks are available (created, inherited, assigned) for given user

Whether a user receives email notifications or not depends on application and user configurations. Engine administrators can set up initial settings for both notification types at user creation (see Application Default Settings). Later, users can specify their own notification settings using WorkflowUI (see WorkflowUI documentation). Email notification messages are created by the system and sent as email messages. Therefore make sure, that SMTP settings are set up correctly, check it in the System properties of your application.

Axon.ivy supports own email notification processes to influence the content of the emails. In case you set up such a notification process, then this process will be executed before the notification email is sent. There are two types of email notification processes:

New Task Mail Notification Process	This process provides the email content for new task notification emails.
Daily Summary Mail Notification Process	This process provides the email content for daily summary notification emails.

Mentioned notification processes will be executed, and their HTML result will be used as email content.

Process Design

Email Notification Processes are used to customize the design and content of the mails sent if the standard mail notification is not proper for your company. It is advised to prepare the important details you want to display in the notification mails and prepare some screen designs as well.

Quick overview how to create notification processes:

Open designer	Start Axon.ivy designer
Create mail notification process	Create new process(-es) which provide email content for your notification mail.
Deploy	Deploy project library as process model and PMV on your Axon.ivy Engine
Configure notification	Configure Axon.ivy Engine to use notification process(-es) for mail notification. For more details see Application Default Settings.
Backup	Make sure that your process models and PMV-s are backed up, just like your System DB, so that in case of some hardware failure your Axon.ivy Engine can be restored easily.

More detailed description of non trivial steps are documented in the following section.

Create mail notification process

Create a normal process within your fresh Axon.ivy project. Place some process start element(-s), and put a simple process together. This process should display an HTML page, preferably using an Endpage process element to save engine resources. The HTML page defines the content of the notification mail while the <title> tag in the <header> section will be used as mail subject.

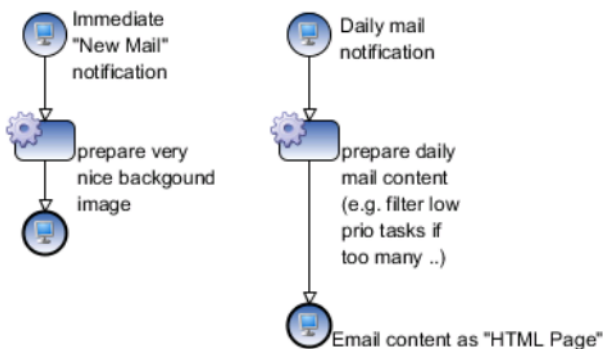


Figure 5.32. Example for both email notification process types

Notification process start elements must follow some naming and parameter conventions in order to be recognised by Axon.ivy Engine as potential email notification processes. Please create process starts as follows:

New Task Mail Notification
Process Start

This process has to accept the new task you want to notify about and the user to which the notification will be sent to.

The name of the Process start must be:

- MailNotification_NewTask

Process start parameters:

- notificationUserId : Number (java.lang.Number) - The ID of the recipient that the new task notification is created for
- notificationTaskId : Number (java.lang.Number) - The ID of the task that is ready to be processed by the user

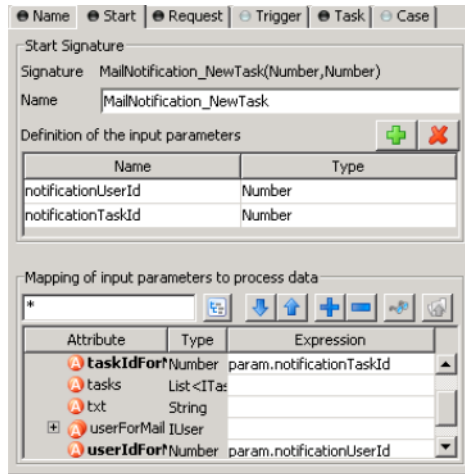


Figure 5.33. Correct declaration of process start element for a "New task" email notification



Tip

You can get a user object (IUser) using a userID calling the method `findUser(int)` on `ivy.session.getSecurityContext()` (see `ISecurityContext` interface in the *PublicAPI* documentation for more details)



Tip

You can get a task object (ITask) using a taskID using the method `findTask(int)` on `ivy.wf` (see `IWorkflowContext` interface in the *PublicAPI* documentation for more details)

Daily Summary Mail Notification
Process Start

This process has to accept only the user we want to send the daily summary to, therefore define the process start element as follows:

The name of the Process start must be:

- MailNotification_DailyTaskSummary

Process start parameters:

- notificationUserId : Number (java.lang.Number) - The ID of the recipient that the new task notification is created for

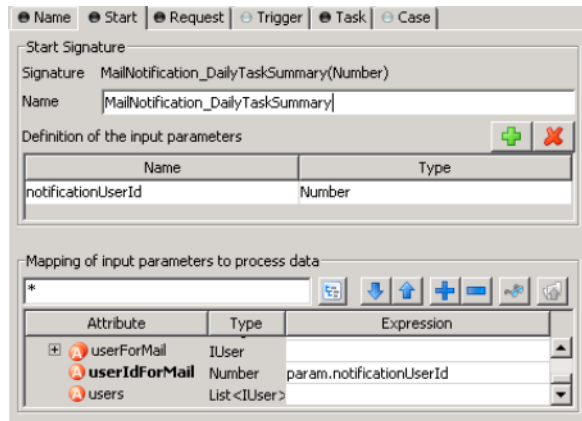


Figure 5.34. Correct declaration of process start element for a "Daily summary" email notification

It is not supported to create multiple processes or process starts within the same project (PMV) for the same notification type (daily/new task). In case you create multiple matching process starts then it cannot be forecasted which process start will be used within the project (PMV library).

Created notification processes will be executed, and their HTML result - the page they display - will be used as content in the notification email.

Deploy

Start the Engine Administrator, deploy your project library as a process model and a PMV. It is important to deploy your notification project within the application you want to use it for.

Configure notification

Having deployed your notification process model in the application where you want to customize the notification make sure that the PMV is active. Go to the Application Default Settings, refresh GUI if necessary, and select the library ID of the PMV that implements the notification process you want to use for email notification. In case you implemented the process start properly, and the project including the notification process is deployed properly, you should see the library ID.



Tip

Should you have difficulties implementing an email notification process, open a WorkflowUI project. It contains two example email notification processes.



Tip

In a Daily notification mail you might need a task list for a user that she may activate and work on. The method `findWorkTasks(...)` on `ivy.wf` does this job. It uses even substitutions and absences to collect the correct task list (see `IWorkflowContext` interface in the *PublicAPI* documentation for more details)

Email Notification Summary

Having customized your email notification process(-es) users of specific application should receive their email notification prepared by your notification process.

You might still have some questions left:

How can a process provide email content?

Defined process should look like a simple process with an HTML dialog end. This process will be executed and the HTML dialog will be taken as content for notification email. In case you use normal HTML Page Step process element(s), then the process

will be executed only up to the first HTML Page and this page will be sent as notification email.

What can be customized exactly? These processes going to "display" an HTML page. Whatever is included in this HTML page, will be sent as content of the specific notification email.

How often is this process executed? Processes for "new task" notification are executed whenever a new task is created (e.g. with a trigger), an existing task expires and needs to be escalated, or the creator of an existing task is changed. On mentioned events the process is executed separately for each user that is responsible for the given task directly, indirectly by his roles or by substitutions (e.g. if a task is created for the role *secretary*, then the process is executed for each user that own the role *secretary*, or substitutes an absent secretary over her role).

Processes for "daily summary" notification are executed once a day for each user that is subscribed for notification on that day.

When is the daily summary process executed? You can setup the execution time in the engine's System properties, under the property: `Email.DailyTaskSummary.TriggerTime`

What happens with images of the HTML page used for email notification? In case this page includes some local images (e.g. CMS, ivy file), then they will be downloaded by Axon.ivy Engine, attached as MIME parts, and re-linked to the email to look just like the HTML page would look in an Internet browser.

You may also include images from remote hosts. If you do so, then Axon.ivy does not download and attach these images, but leaves the links as they are. Some Email clients will block these images (protecting their user from phishing attacks) until the user explicitly says to download external images.

How notification processes can be set? Notification processes can be set for each application independently. First of all you have to deploy a notification process, then select notification process in the Application Default Settings dialog.

Can I cancel a notification? Yes, in case your process terminates normally without displaying any HTML page.



Tip

Use EndPage process element to reduce resource usage of your email notification process(-es).

Chapter 6. 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, to log files and on Windows (x86) to the Windows Event Log. 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 6.1. 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**log 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/ch.ivyteam.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 [...ner.OpenEditor.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.

Configure ULC logging on server and client

It is possible to log ULC communication and status messages both on client and server side. Both a default log level and a log level per known Ivy user can be configured, as well as a log file path/name where the log of an ULC session should be stored permanently.

To configure ULC logging, the files *configuration/jnlpcnfig.any* and *configuration/ulclogcnfig.any* within the Axon.ivy Engine installation folder must be edited. The former one controls the client-side logging, the latter one the server-side logging. The log configurations are similar in both cases and edited with *Anything* notation.

To modify the *client logging* properties locate the following section in *jnlconfig.any*:

```
...
/log-level {
  /user-specific {
    /johndoe FINER
  }
  /default WARNING
  /log-to-file "C:/temp/xpertivy_ulc_client.log"
}
...
```

To modify the *server logging* properties locate the following section in *ulclogconfig.any*:

```
{
  /user-specific {
    /johndoe FINER
  }
  /default WARNING
  /log-to-file "xpertivy_ulc_server.log"
}
```

To enable user-specific logging on server side and/or client side, enter the known name of the Ivy user (or AD user) to log for in the `/user-specific` section, followed by a valid log level (one of SEVERE, WARNING, INFO, FINE, FINER, FINEST) just as specified in the example configurations with the user *johndoe*. Use double quotes around the Ivy user name if it contains special (i.e. non-ASCII) characters or white space.

To set the default log level (which will be used for all sessions) set the log level for the key `/default`.

Finally, to enable logging to a file, enter a filename **in double quotes** for the key `/log-to-file` as shown above. If the `/log-to-file` slot is omitted altogether or if its value is a `*`, then no logfile will be created.



Note

Changes in log configurations take place immediately for all new sessions or client application starts; a restart of the engine is not necessary. However, you have to restart any running client applications for the new log settings to take effect.

The created log files will automatically include the user name (if user-specific logging is enabled), the HTTP session ID and an ID for the application. The two ID's mentioned are part of the client and the engine log file, so it is easy to find the two logs related to a client-server communication.



Tip

For reasonable results a default level of WARNING or INFO is recommended. For error analysis, FINER on the client and FINER on the server is recommended, on a per-user basis.

If you enable ULC logging, it is recommended to specify the `/log-to-file` parameter. This way the log output is redirected to a file, otherwise the logs are written to the console.



Warning

Please bear in mind that the logfile path that you specify may not be in correct format for every client platform (Linux/Windows). If this should be the case then the logfiles will be created in the client system's default temporary directory.

Be aware that log files can become very large (up to several hundred MB per day and user if FINE /FINER /FINEST is used as log level).

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`)

All configurations are administered by the following System Properties:

Property	Description
<code>ProcessEngine.FiringStatistic.Active</code>	<p>True for activation, otherwise false. An activation may slow down the engine performance!</p> <p>A restart is not required after de-/activation. The setting take effect, maximum after the duration of the defined interval (System Property: <code>ProcessEngine.FiringStatistic.Interval</code>).</p>
<code>ProcessEngine.FiringStatistic.Interval</code>	<p>Interval in seconds the statistic is written to the log directory. Default value is 300 seconds (5 minute). After each interval the statistic informations are cleaned after the dump out.</p> <p>A restart is required after a value change.</p>

Table 6.2. System properties to configure the firing statistic

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.

Process Element	Internal Category	External Category
Program Interface		The execution of the defined Java-Class.

Table 6.3. 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.
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 6.4. 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 uncommenting the following line in an ivy launch control file **.ilc* (See chapter Windows Program Launcher Configuration) to activate remote access:

```
ivy.management.port=9003
```

On other operating systems the below options must be added to the launch configuration script of the engine (*AxonIvyEngine.conf*):

```
-Dcom.sun.management.jmxremote.port=9003  
-Dcom.sun.management.jmxremote.login.config=jmx  
-Djava.security.auth.login.config=configuration/jaas.config  
-Dcom.sun.management.jmxremote.ssl=false  
-Djava.rmi.server.hostname=<IP of the machine>
```

Auto Discovery (JDP)

Some monitoring tools can auto discover running JMX servers in the network. If so the user does not have to know the server host and JMX IP port. Instead he can simple click on the auto discovered server.

Axon.ivy Engine supports auto discovery by default if JMX is activated on Windows. However, you can disable this feature by uncommenting the following line in the **.ilc* file::

```
ivy.management.autodiscovery=false
```

On other operating systems the following option must be set to the launch configuration script of the engine (*AxonIvyEngine.conf*) to activate it:

```
-Dcom.sun.management.jmxremote.autodiscovery=true
```

Provided MBeans

The Axon.ivy Engine provides performance and management information by a set of MBeans. These allow to monitor 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.

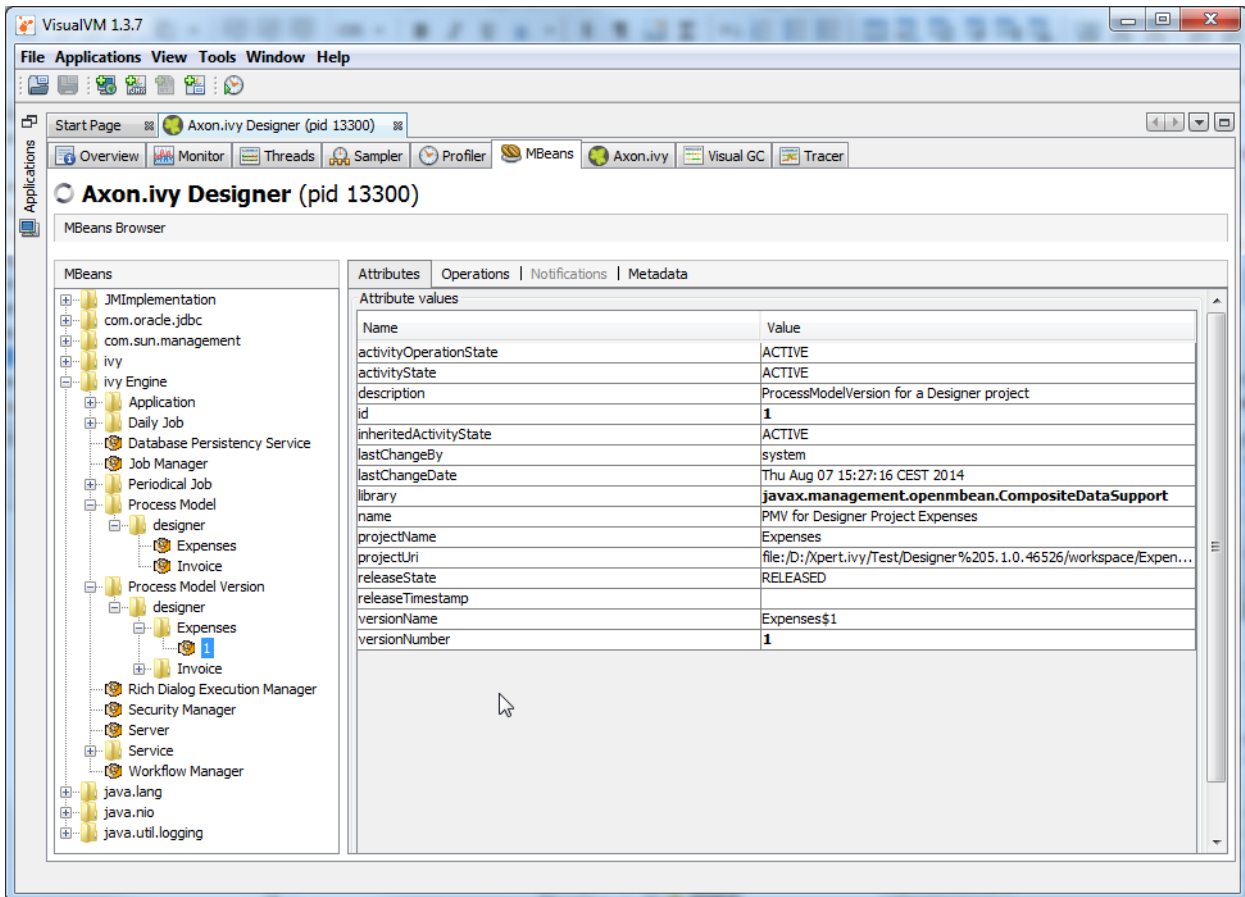


Figure 6.1. MBeans Tree of Axon.ivy shown in MBeans Plugin of VisualVM

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:

- Axon.ivy Engine:type=External Web Service,application=MyApplication,environment=Default,name=Echo (43838347ABCD)
- Axon.ivy Engine:type=Job Manager
- Axon.ivy Engine:type=Process Start Event Bean,application=MyApplication,pm=MyProcessModel,pmv=1,name="MyStartEventBean (3485471349/start.ivp)"

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, maxCharactersToCache]

ivy Engine type=CacheClassPersistencyService,name=*,strategy=CacheAllRemoveUnused [maxBytesToCache, maxCharactersToCache, countLimit, usageLimit]

ivy Engine type=CacheClassPersistencyService,name=*,cache=LongBinaries [readHits, readMisses, writes, cachedLongValues, clearCache()]

ivy Engine type=CacheClassPersistencyService,name=*,cache=LongCharacters [readHits, readMisses, writes, cachedLongValues, clearCache()]

ivy Engine type=CacheClassPersistencyService,name=*,cache=ObjectsAndAssociations [objectReadHits, objectReadMisses, objectWrites, cachedObjects, associationReadHits, associationReadMisses, associationWrites, cachedAssociations, clearCache()]

- Memory (Java Heap, Perm Gen)

java.lang:type=Memory

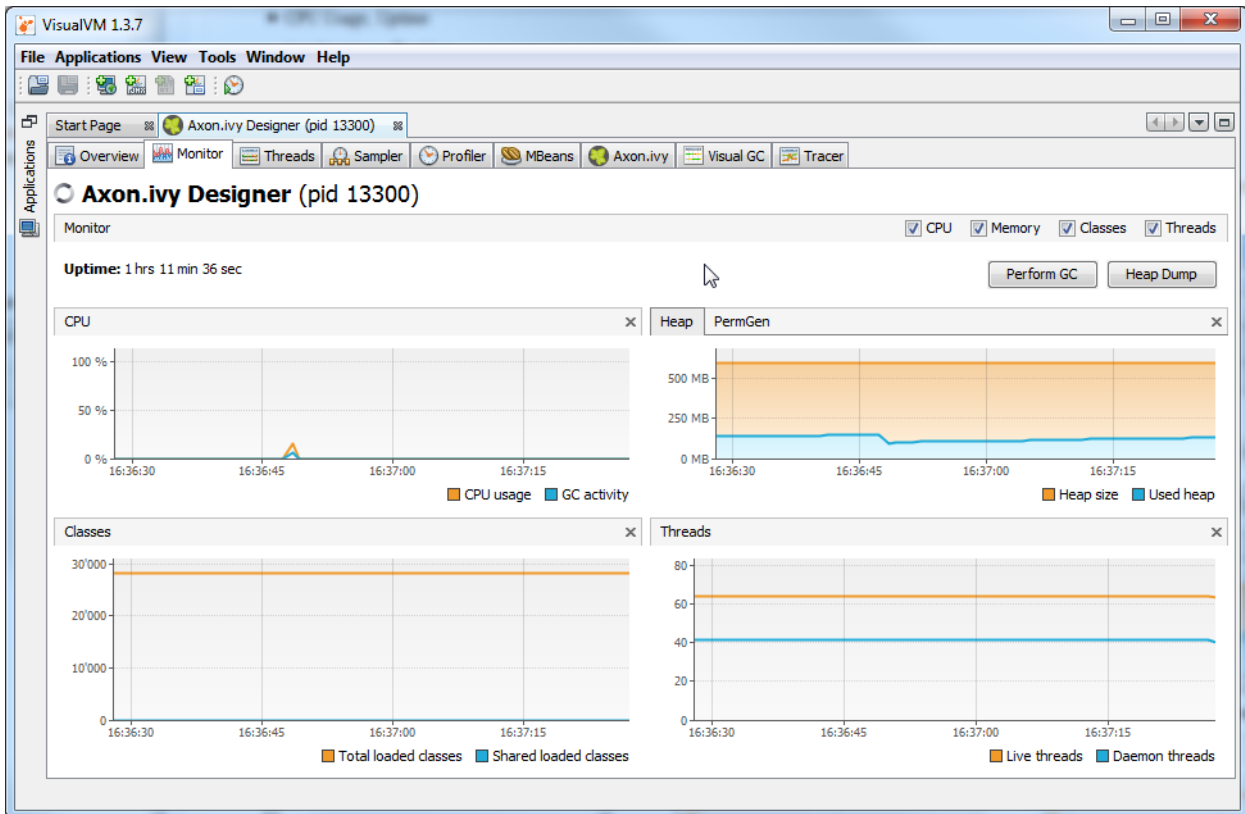
- CPU Usage, Uptime

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 JDK (Java Developer Kit).

Axon.ivy Plugin for VisualVM

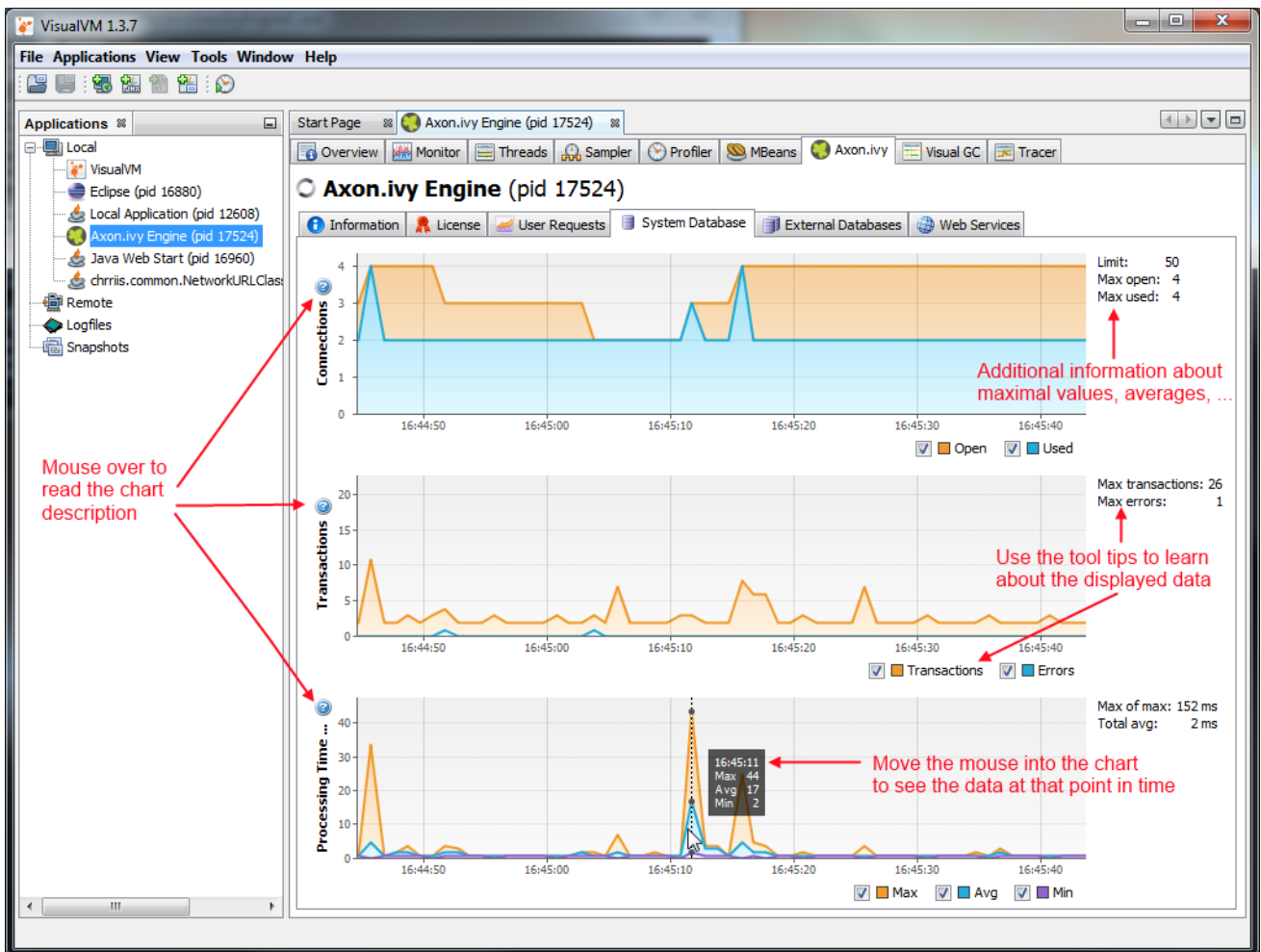
In the delivery of Axon.ivy we provide a dedicated plugin for VisualVM that allows you to monitor some of the technical aspects of an Axon.ivy Engine or Designer. For example you can observe the current transactions on the System Database, whether you violate the licence or how many requests are running 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.

Engine Administration

In the Engine Administration application you can use the Runtime Information page to view information about the java heap memory and all running java threads with their current call stack. Another important page is the About page where you find information about the Axon.ivy Engine version, Java version and the Java system properties.

For each external database you can find information about the last SQL statements that have been executed on the database and the open connections to the database.

Chapter 7. Miscellaneous

Rich Dialogs

Provide Custom ULC Widgets

The chapter *Concepts > Extensions > Rich Dialog Clientside Libraries > Provide custom ULC Widgets* in the *Axon.ivy Designer Guide* explains how to provide Custom ULC Widgets.

Provide Customer Rich Dialog Client Certificate

If a client starts the first time a Rich Dialog a security information dialog box pops up showing information about the publisher of Axon.ivy and its certificate. For some clients it may be confusing that a certificate from the provider of Axon.ivy is displayed instead of the certificate of the web site or the company they working for.

The chapter *Extensions > Rich Dialog Clientside Libraries > Provide Customer Certificate* in the *Axon.ivy Designer Guide* explains how to provide customer Rich Dialog client certificate.

Rich Dialog Session Timeout

The normal session timeout has no effect to Rich Dialogs because they send recurring requests to the server in shorter intervals than the session timeout. Therefore you can configure a special timeout feature for Rich Dialogs. It detects if an open Rich Dialog was not touched for the configured time and stops that Rich Dialog application. The user will receive a message so that he knows why the Rich Dialog has been closed. You can choose whether you want to have a Rich Dialog message box or if you want to see the message in your browser. With the former, the Rich Dialog and the session is closed only after the user confirmed the message. With the latter, the Rich Dialog and the session is closed immediately when the timeout happens thus freeing the server side memory. In this case you can even adapt the HTML file that is used. It is the file *webapps/ivy/info/time_out.jsp*.

To configure the Rich Dialog session timeout just go to the file *configuration/jnlconfig.any* and adapt the settings for it.

Rich Dialog Look and Feel

The look and feel that is used for Rich Dialogs can be configured per applications. To change it open in the AdminUI the application and change the configuration property *ria.lookandfeel*.

Compressed client libraries

Rich Dialog clients download many jars during the first startup. If the network connection of the client is poor, it is possible to speed-up the download by sending compressed pack200 jars. It will speed-up the download time, but the client will require more resources to unpack the downloaded jars.

In order to send compressed jars they must be provided within the */clientlib* directory of the engine. The following sample script will create these pack.gz files. The script must be run within the engines clientlib directory. Note that the pack200 binary comes from a JRE or JDK.

On Windows:

```
for /R %f in (*.jar) do pack200 "%f.pack.gz" "%f"
```

On Linux:

```
for f in `find signed -name '*.jar'`; do
  echo "packing $f"
  pack200 $f.pack.gz $f
```

done

If the script runs successfully the clientlib directory contains files ending with pack.gz next to the normal jars.

ULC Load Tests

ULC Load is a product from Canoo Engineering AG to make load tests on ULC Applications (such as Rich Dialogs are). This chapter describes how to use ULC Load together with an Axon.ivy Engine.

Requirements

- Axon.ivy Designer installation
- Axon.ivy Engine installation
- ULC Load 3.0.4
- Java Runtime Environment (JRE) 1.6



Note

ULC Load is not shipped with Axon.ivy and needs a separate licence. It can be obtained from Canoo Engineering AG (<http://www.canooo.com>).

IvyTeam AG does not provide any support for ULC Load. Please contact Canoo Engineering AG to get support for ULC Load.

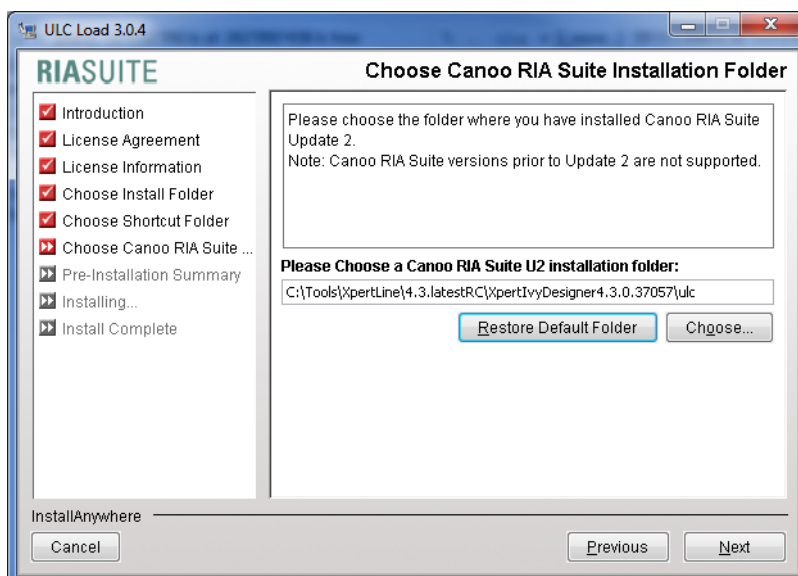


Note

If you run ULC Load with JRE 1.7 you cannot save your test plan. Ensure you have installed a JRE 1.6. If it is not possible to uninstall JRE 1.7 reconfigured the property `lax.nl.valid.vm.list=1.5+` to `lax.nl.valid.vm.list=1.6` in the property file `bin/ULC Load 3.0.4 UI.lax` of your ULC Load installation directory.

Install ULC Load

- Start the installation of ULC Load.
- At the Step Named *Choose Canoo RIA Suite* select the *ulc* directory of your Axon.ivy Designer Installation.





Note

The Axon.ivy Designer must have been started at least one time. Otherwise the ulc directory needed for the ULC Load installation is not available.

- Finish the installation of ULC Load
- Edit the File *ULC Load 3.0.4 UI.lax* in the *bin* directory of your UlcLoad installation.

Look for the line which starts with *lax.class.path=*.

Add an entry to the end of this line which points to the *ivyUlcExtension-*.jar* file in the *clientlib/signed* folder of your engine installation.

Add an entry to the end of this line for each library in the *clientlib/signed/windows_64_native* folder of your engine installation. If you are using an other operating system than Windows 64 bit, use the libraries of the operating system you are running ULC Load with.

Prepare Engine

ULC Load will only work correctly with Axon.ivy Engine (Designer) if you change the following line in the *context.xml* file in the directory *webapps/ivy/META-INF/* of the engine/designer installation directory:

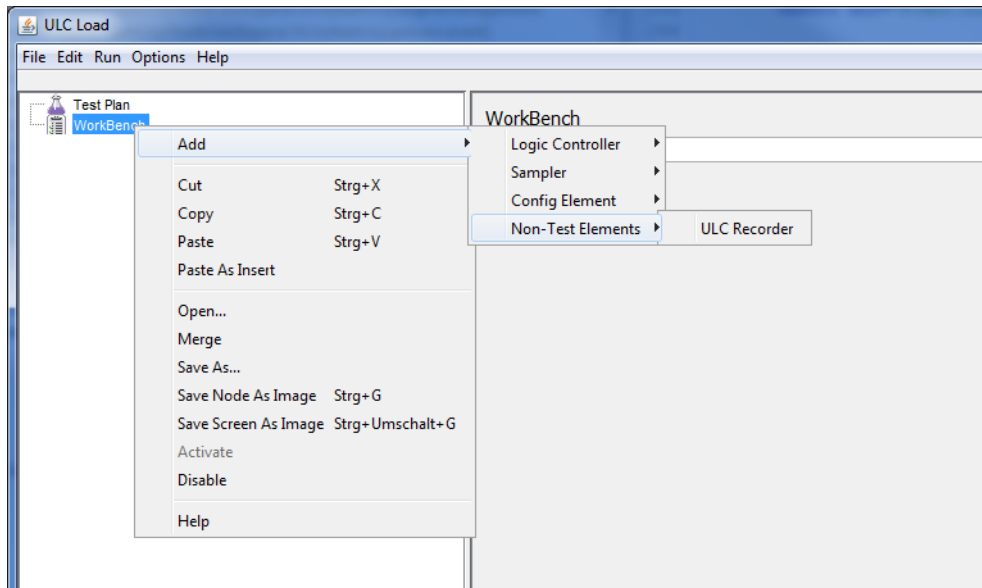
```
<Context antiResourceLocking="false" privileged="true">
```

change to:

```
<Context antiResourceLocking="false" privileged="true" useHttpOnly="false">
```

Record a Test Scenario

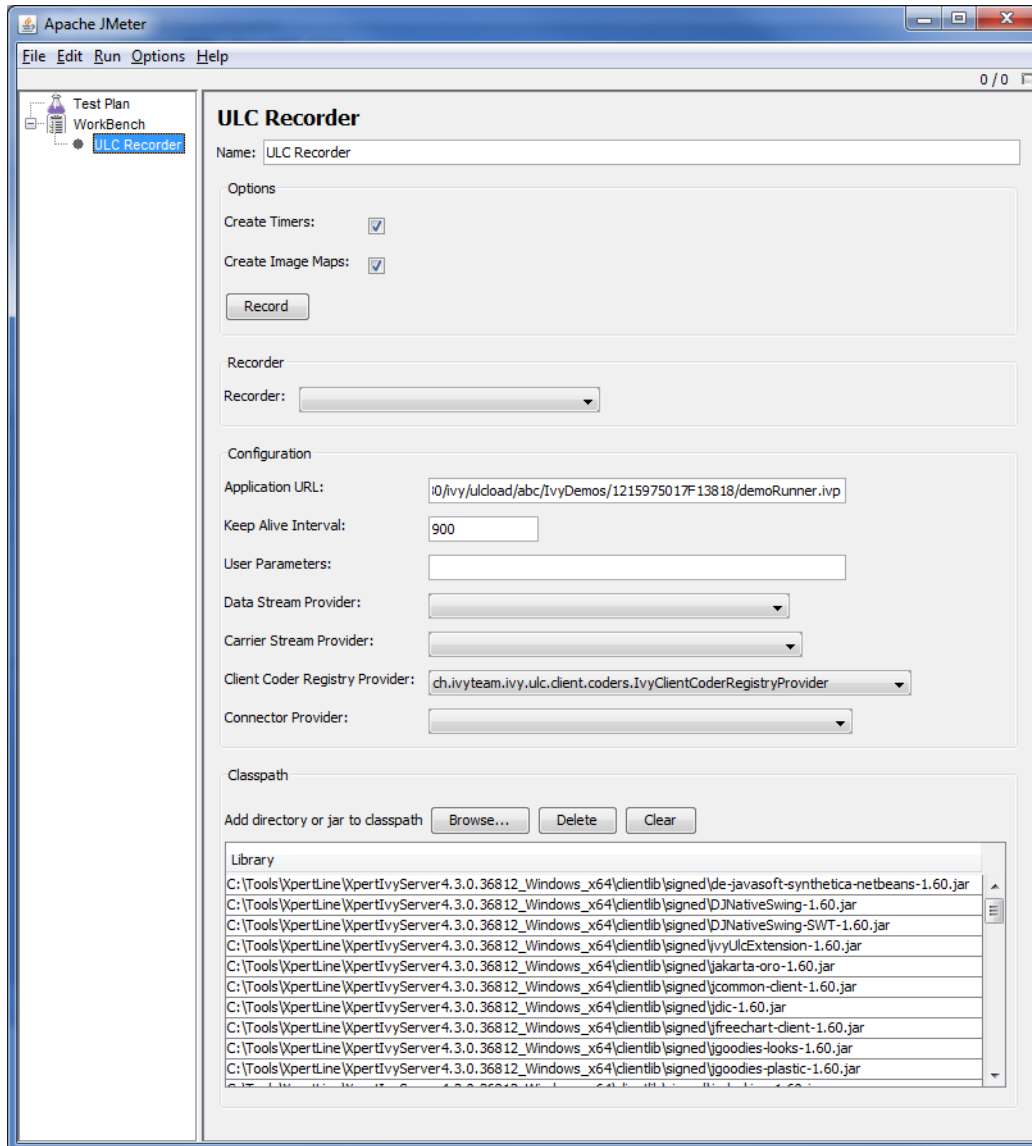
- Start ULC Load
- Create a Recorder



- Configure the Recorder
 - Add all jars in the *clientlib/signed* folder of your Axon.ivy Engine installation to the classpath.
 - Choose the *IvyClientCoderRegistryProvider* as *Client Coder Registry Provider*

- Copy the URL of the process you want to start to the Application URL Field and replace the prefix */ivy/pro* with */ivy/ulcloud*.

Example: If you want to test the process with the URL *http://localhost:8080/ivy/pro/abc/IvyDemos/1215975017F13818/demoRunner.ivp* you have to use the Application URL *http://localhost:8080/ivy/ulcloud/abc/IvyDemos/1215975017F13818/demoRunner.ivp*



- Click the Record button. This starts the application and records every action you do. The recording automatically stops when the application is closed.



Note

It's not possible to change the look and feel for recording tests. The application will always start with the default swing look and feel. As ULC Load does not run any client side code when executing the tests, the look and feel wouldn't have any effect on the testruns anyway.



Warning

It's not possible to test processes which use Html Pages. When the process comes to an Html Page step ULC Load will display an error.

Run Load Test

- There are no special configuration necessary to run a Load Test.
- When recording a test, ULC Load creates a *Thread Group* which contains all recorded actions. With the menu Run -> Start these actions can be replayed.
- On the *Thread Group* itself its possible to configure how often the actions are replayed and with how many concurrent users.

HTML Dialogs

Primefaces Theme

The default Primefaces theme that is used in Html Dialogs can be configured in the web.xml. It can also be configure per application. To change it open in the AdminUI the application and change the configuration property `jsf.primefaces.theme`. Moreover, the theme can also be set per session. See Java class `ch.ivyteam.ivy.jsf.primefaces.theme.IvyPrimefacesThemeResolver` in Public API for details.

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.

Update Notification

When newer Axon.ivy versions are available a message will be displayed on the Axon.ivy Engine main web page. The message contains information about the new versions and where those can be downloaded.



Note

While checking for new versions the following statistic information are sent to the update server. These information are only used to improve the product.

- Engine (version, up time)
- Configuration (number of: cluster nodes, users, licenced users, applications, process model, process model version, deleted process model version, running cases, running tasks)
- 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 the host without being able to read the original IP address and MAC address itself)



Note

If you do not want that update notification messages are shown or that statistic information are sent to the update server then you can set the system property `UpdateChecker.Enabled` in the Admin UI to false. This will turn of the whole update checker service.

Tool Reference

AxonIvyEngine

Axon.ivy Engine program. This program starts an instance of the Axon.ivy Engine.



Tip

On Windows Axon.ivy Engine can also be started as a Windows Service. More information about Axon.ivy Engine Windows Service can be found in the section `AxonIvyEngineService` tool reference chapter.



Tip

On Linux an instance of Axon.ivy Engine can also be started as a systemd service. More information how to install Axon.ivy Engine as a systemd service can be found in the section `InstallService` of the tool reference 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.	no
-status	Prints the current status of the engine.	no

Table 7.1. 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 7.2. AxonIvyEngine Launchers

AxonIvyEngineConfig

AxonIvyEngineConfig

Axon.ivy Engine Configuration program. The program is used to configure the Axon.ivy Engine. The program supports two modes:

- graphical user interface mode and
- headless mode which needs no graphical user interface on the server machine.



Note

More information can be found in the chapter Configuration

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 7.3. AxonIvyEngineConfig Options

Launchers

The following program launchers are available for the Axon.ivy Engine Configuration program:

Platform	Console support	Launcher
Windows	no	bin/AxonIvyEngineConfig.exe
Windows	yes	bin/AxonIvyEngineConfigC.exe
Linux	yes	bin/AxonIvyEngineConfig

Table 7.4. AxonIvyEngineConfig Launchers

AxonIvyEngineService

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.

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 7.5. AxonIvyEngineService Options

Launchers

The following program launcher is available for the Axon.ivy Engine Service program:

Platform	Launcher
Windows	bin/AxonIvyEngineService.exe

Table 7.6. AxonIvyEngineService 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.
<command> help	Get help to a specific command e.g. EngineConfigCli config-db help

Table 7.7. EngineConfigCli Options

ControlCenter

The Control Center program. With Control Center program you can start and stop the different Axon.ivy Engine types. The Control Center allows you to change the configuration of the Axon.ivy Engine and on Windows even the configuration of Axon.ivy Engine Windows Service.



Note

More information can be found in the section Control Center of chapter Configuration

Launchers

The following program launchers are available for the Control Center program:

Platform	Console support	Launcher
Windows	no	bin/ControlCenter.exe

Platform	Console support	Launcher
Windows	yes	bin/ControlCenterC.exe
Linux	yes	bin/ControlCenter

Table 7.8. ControlCenter Launchers

InstallService

The install service program helps to install the Axon.ivy Engine as a systemd Linux daemon. To install the service:

1. `cd` to the `bin/` directory of your Engine
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 7.9. InstallService Launchers

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
ivy.heap.max.ratio	yes	The maximum heap size (-Xmx) in percentage of the physical memory of the machine.
ivy.heap.max.size	yes	The maximum heap size (-Xmx) in megabytes.
ivy.heap.start.size	yes	Start heap size (-Xms) in mega bytes.
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 7.10. Ivy Launch Control Properties

The properties that are bold are the most used properties.

Unix Launcher Configuration

To change the Java virtual machine (JVM) options for the Engine launcher you can edit the *AxonIvyEngine.conf* file. To adjust JVM Options for any other launcher (e.g. ControlCenter or Engine stop) you have to edit *control.conf*.



Tip

When you want to optimize the JVM options for the engine you need to edit *AxonIvyEngine.conf*. Normally there is no need to edit *control.conf*.

Chapter 8. Troubleshooting

Introduction

Here you will find solutions to some of the most common problems related to Axon.ivy Engine. If you can't find your solution here there are some other sources which could help:

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 support@soreco.ch (support may be subject to charging, depending on your licence agreement).

Program / Engine start problems

If a program or engine does not start, execute it in a console and look at the console output. In most cases error messages are written to the console output giving you an idea about the problem.

JVM cannot allocate enough Memory

Effect

If the program or engine cannot be started and in the console the following error message is written:

```
>Error occurred during initialization of VM
Could not reserve enough space for object heap
Create registry entry 'SYSTEM\CurrentControlSet\Services\EventLog\Application\AxonIvyEngineConfig
Launchers' for event log if it does not exists
Error: Could not initialize java virtual machine (-4)
```

Cause

The JVM tries to allocate memory for the Java heap, but it does not succeed. The amount of memory the JVM tries to allocate is too large for the operating system. Consequently the JVM cannot be started.

Solution

On Windows use an Ivy launch control file to set the maximum heap size to a lower value (smaller or equal to 1 GByte).

On Linux modify the AxonIvyEngine.conf script to set the maximum heap size to a lower value. The maximum heap size can be configured with the Java option -Xmx.

Socket name not available on this system

Effect

If the Enterprise Edition engine cannot be started and in the console the following error message is written:

```
Error while starting Axon.ivy Engine org.jgroups.ChannelException:
    failed to start protocol stack
    ...
Caused by: java.net.SocketException:
```

```
The socket name is not available on this system
...
```

Cause

This happens if your system supports IPv6 and you try to bind a cluster node to a IPv4 address.

Solution

Set the following java option

```
-Djava.net.preferIPv4Stack=true
```

On Windows use an Ivy launch control file. Change the line

```
#ivy.vm.additional.options=
```

to

```
ivy.vm.additional.options=-Djava.net.preferIPv4Stack=true
```

On Linux modify the serverLauncher.sh script to set java option.

Request Entity Too Large

Effect

If the following error message appears in the browser:

```
Request Entity Too Large! - The HTTP method does not allow the data transmitted, or the da
```

Cause

If your Active Directory contains large user groups and dependencies among them then it may occur that single sign on for some users that are members in a lot of user groups does not work. This is because the packet size of the AJP13 protocol that is used for the communication between IIS and Axon.ivy Engine is too small to hold all the necessary security information.

Solution

The problem can be solved by increasing the packet size (default value 8192) of the AJP13 protocol. The packet size must be configured with the same value on the IIS and Axon.ivy Engine side. If one side is not configured to the same size as the other, then the communication may fail. On the IIS side the packet size can be configured in the *workers.properties* file by setting the worker attribute `max_packet_size`.

Example:

```
worker.AxonIvyEngine.max_packet_size=16384
```

On the Axon.ivy Engine side the packet size can be set by changing the value of the system property `WebServer.AJP.PacketSize` (See chapter System Properties to learn about how to change a system property).

Install a secure random provider

Effect

In the log the following warning appears:

```
Failed to generate random number within 10 seconds. This could slow down the JSF runtime c
```

Cause

The Engine is running on a highly virtualized cloud host and can not provide enough random data through hardware interrupts (e.g. keyboard input or mouse moves).

Solution

Install and run an alternative random provider like 'haveged'. See the installation instructions here <https://www.digitalocean.com/community/tutorials/how-to-setup-additional-entropy-for-cloud-servers-using-haveged>

Blocking Application/UI

If an application periodically does not answer for a couple of seconds, then the reason can be a long running full GC. Visit the chapter GC Optimization for possible solutions.