HITACHI
Inspire the Next

Configuring Pentaho with Integrated Windows Authentication (IWA)

HITACHI
Inspire the Next

Change log (if you want to use it):

Date	Version	Author	Changes
11/27/2017	1.0	Carlos Lopez	

Contents

This page intentionally left blank.

Overview

Pentaho can be configured to use many mechanisms for authentication and authorization, such as the lightweight directory access protocol (LDAP) or database-based authentication (JDBC authentication) from Microsoft Active Directory.

This document aims to work through the steps needed to set up Pentaho to authenticate using Integrated Windows Authentication (IWA) with a pre-configured Microsoft (MS) Active Directory.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	8.0
Other Software/Applications	Microsoft Active Directory installed on MS Windows Server 2012 R12 Internet Information Services 8.5 as provided with MS Windows Server 2012 R2

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

Prerequisites

This document assumes that you have knowledge of Pentaho and have already installed it and configured it to authenticate using Microsoft Active Directory on MS Windows Server 2012 R2.

Use Case: Single Sign-On

> *Janice administers an environment where users can access single sign-on (SSO) to use their Windows networking credentials to get into most web applications they use. She knows that unlike other authentication mechanisms, Integrated Windows Authentication does not prompt the user to enter a username and password to log into Pentaho through the web browser. Instead, the current Windows user authentication on the computer is passed to Pentaho through the web browser. In this way, if the user does not have access to Pentaho as configured in the Microsoft Active directory configuration for Pentaho, then the user will be prompted to enter his or her username and password.*

Intro to Authentication and Authorization in Pentaho

To understand how to configure Pentaho to use a database-based authentication scheme, you will need to understand what authentication and authorization are in terms related to Pentaho. Pentaho uses the Spring Framework for authentication and authorization purposes.

Authentication happens when the user logs in. The user is checked for validity and activity before being able to log in.

Once the user is granted permission, we check the user's roles to determine what the user is authorized to do in the server. The roles are assigned when the user's identity has been verified. Roles give the user operational permissions on the server, such as **Manage Security**, **Schedule Content**, and **Manage Data Sources**.

Keep in mind that a user may be able to open a report, but not allowed to see the contents; this is not to be confused with authorization. Being able to see the contents of a report is controlled through Mondrian roles in analyzer reports. These are security-constrained accesses and are beyond the scope of this document.

Installing and Configuring Tomcat IIS Connector

You can download the `tomcat_iis_connector` at Apache's site, and then configure it. You can find details on these topics in the following sections:

- Install Tomcat IIS Connector
- Configure Tomcat IIS Connector

Install Tomcat IIS Connector

The downloaded zip file contains the configuration files necessary for the Internet Server Application Programming Interface (ISAPI) filter to run and communicate with Pentaho's Tomcat.

1. Extract the downloaded zip file and place the contents in a convenient folder in your `C:\` drive, such as `C:\tomcat_iis_connector`.
2. If you extracted the Apache JServ Protocol (AJP) Connector to a directory other than the default (`C:\tomcat_iis_connector`):
    a. Edit the `isapi_redirect.properties` file and make sure that the `log_file`, `worker_file`, `worker_mount_file` and `rewrite_rule_file` properties point to the correct locations.
3. Open `/tomcat_iis_connector/conf/workers.properties.minimal` and change the AJP port number from `8009` to `8023`.

It is currently configured to this port in Pentaho's `/tomcat/conf/server.xml`. If you wish to use a different port number, make sure to change both files accordingly.

4. Make sure the following Windows accounts have full access to the `C:\tomcat_iis_connector` folder and to the folder where the Pentaho Server is installed:
    ```
    IIS_IUSRS
    IUSR
    ```

Configure Tomcat IIS Connector

To get the connector working, you must first set up a registry value for the `tomcat_iis_connector`:

1. Make sure you back up your registry.
2. Run `regedit` as a Windows administrator.
3. Find the key: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Apache Software Foundation\`
    a. Create one new key entry there called `Jakarta Isapi Redirector`.
    b. Create another new key entry named `1.0`. Under this key, create six new string values as shown in this table:

Table 1: String Values for Key Entry 1.0

	Registry Key Name	Value
1	@	(Leave blank)
2	extension_uri	/jakarta/isapi_redirect.dll
3	log_file	C:\tomcat_iis_connector\ logs\isapi_redirect.log
4	log_level	error
5	worker_file	C:\tomcat_iis_connector\conf\workers.properties
6	worker_mount_file	C:\tomcat_iis_connector\conf\uriworkermap.properties

The result of creating these string values should look like this:

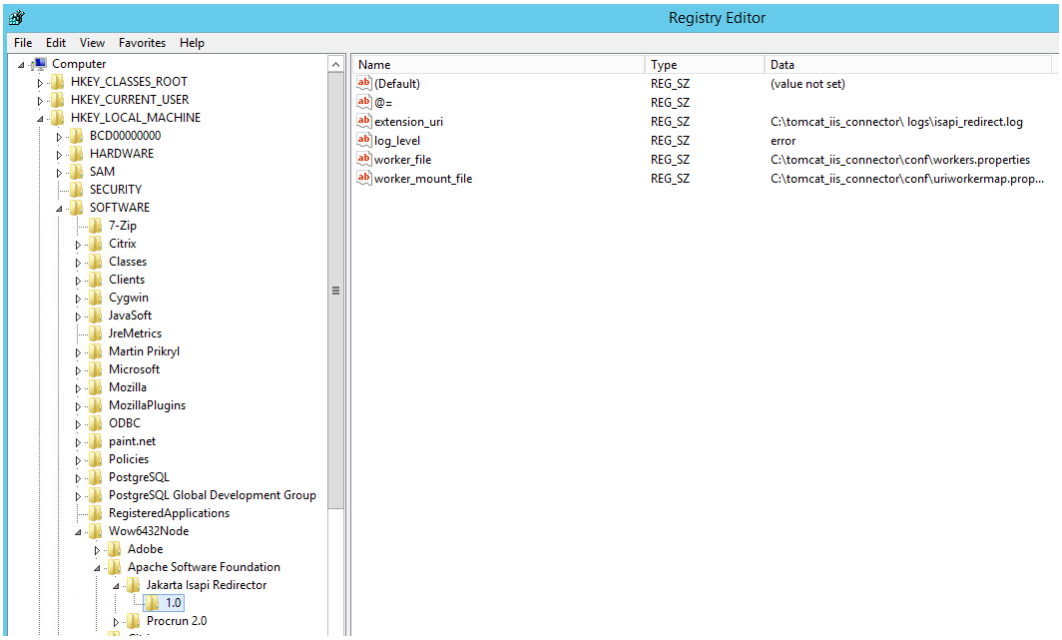

Figure 1: New String Values

The entries in the registry must match the entries in the configuration files in `C:\Tomcat_IIS_Connector`.

4. Configure the following files if you have one or more Tomcat instances (workers):
    a. In `C:\tomcat_iis_connector\conf\uriworkermap.properties`, find the entry `/*=worker1`.
        i. To enable more workers, add a `worker2` there, using a comma-separated list.
        ii. Add more workers as needed.
    b. In `C:\tomcat_iis_connector\conf\workers.properties`, change the following entries according to your environment:

```
#
# The workers that jk should create and work with. You can add more workers
accordingly
#
worker.list=worker1
# Defining a worker named worker1 and of type ajp13.
# Note that the name and the type do not have to match.
worker.worker1.host=localhost (ip address or dns name of the Pentaho
Server)
worker.worker1.port=8009
worker.worker1.type=ajp13
#worker.worker1.port=8009 this port is configured in tomcat/conf/server.xml
```

Installation and Configuration of IIS Role

The following instructions may vary based on your installation of IIS 8.5. Below, you will find the instructions to get Pentaho working with IIS 8.5.

You can find details on these topics in the following sections:

Add Server Roles

Here are the steps to add server roles:

1. Under the **Server Manager**, select the **Dashboard**.
2. Add a new **Server Role**.
3. Under **Roles**, select **Web Server (IIS)**.

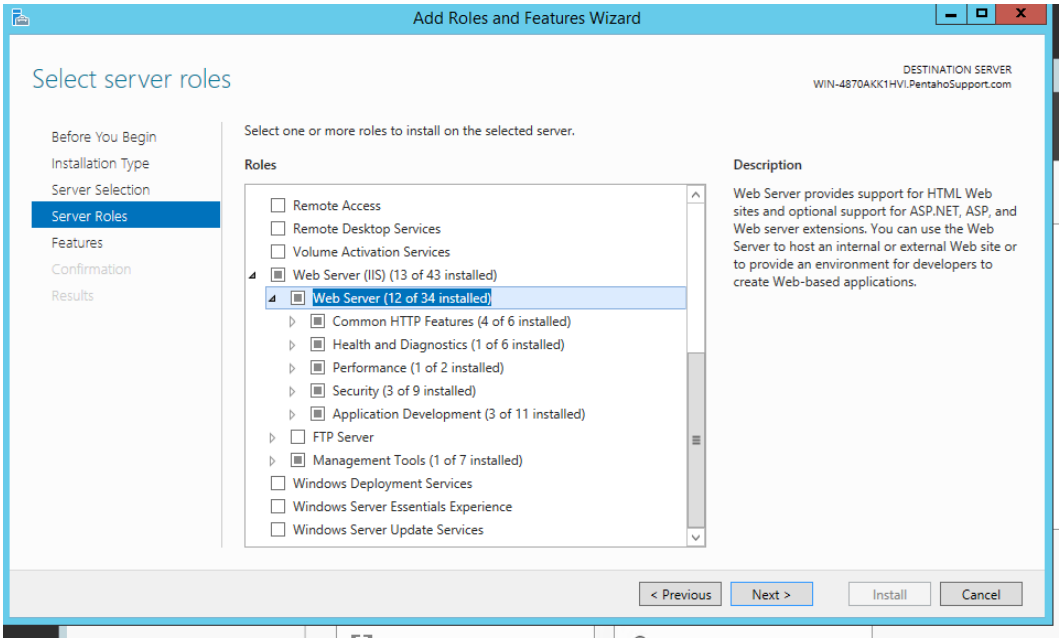

Figure 2: Add Roles and Features Wizard

4. Select **Application Development**, and then select **ISAPI Extensions** and **ISAPI Filters**.

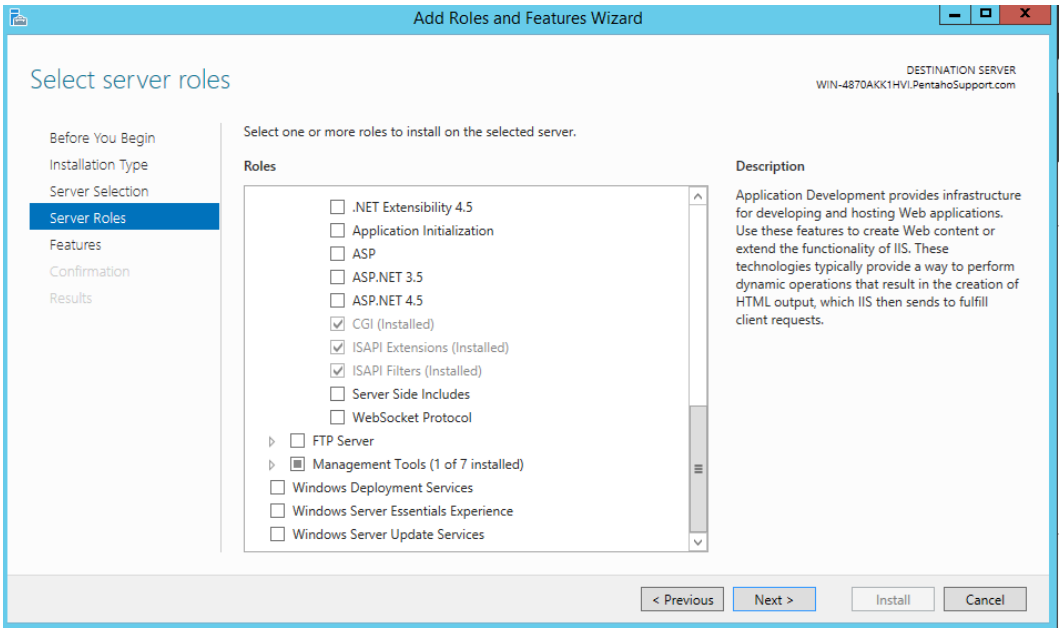

Figure 3: Add Roles and Features Wizard

5. Proceed to installation by clicking on **Next**, and then **Install**.

Set Up ISAPI and CGI Restrictions

Once installation is complete, open the IIS Manager. Follow these steps for setting up ISAPI and CGI restrictions:

1. Select your server.
2. Open **ISAPI** and **CGI Restrictions** from the features view.

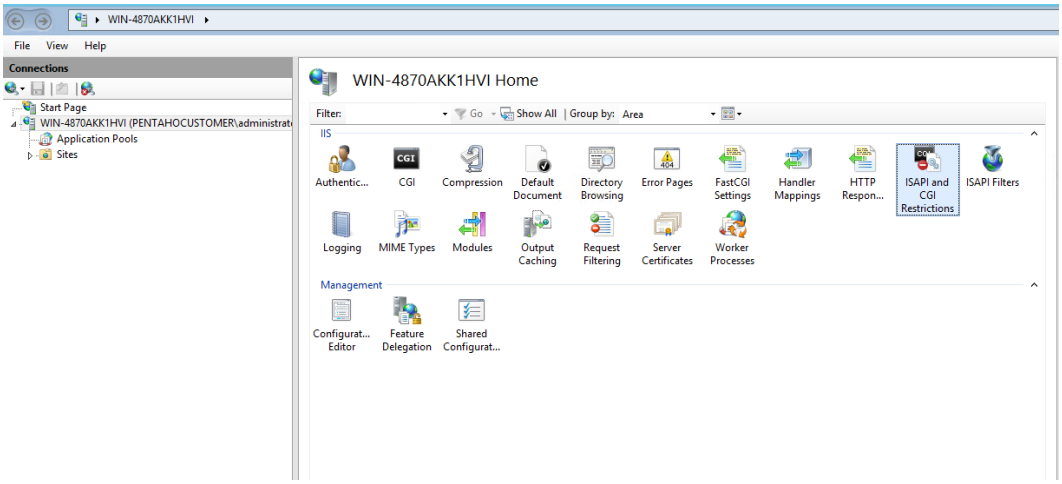

Figure 4: ISAPI and CGI Restrictions

3. For ISAPI or CGI path, click on the ellipsis (...) button and search for the `isapi_redirect.dll`.
4. In the description field, type `isapi_redirect`.

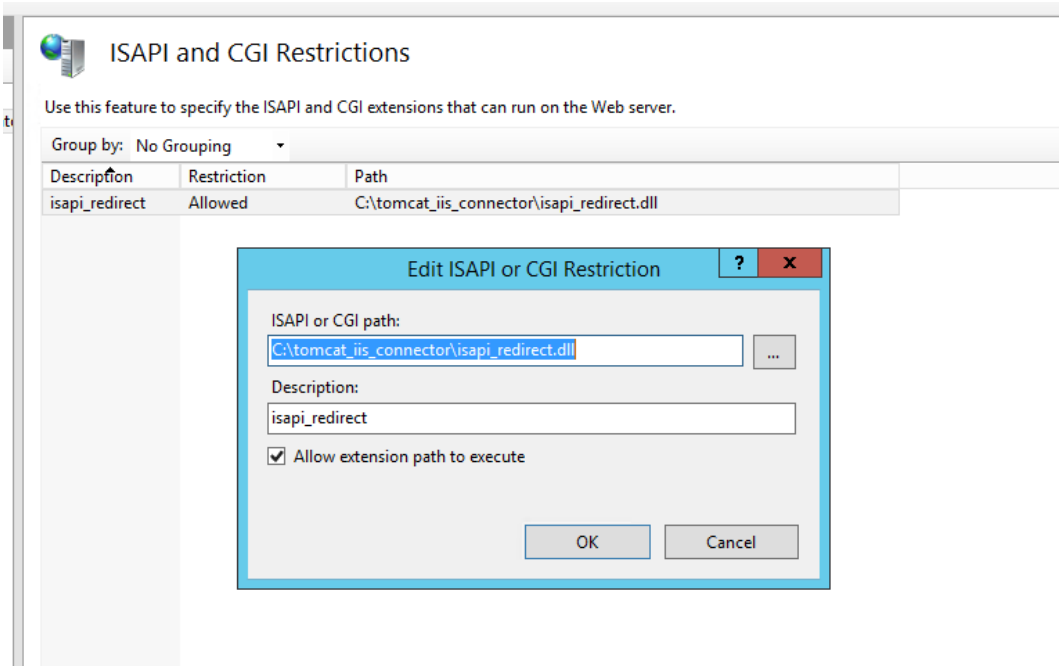

Figure 5: Edit ISAPI or CGI Restriction

Create a Virtual Directory and Handler Mappings

Follow these steps to create a virtual directory and handler mappings:

1. In **IIS Manager**, right-click on the default website and choose **Add Virtual Directory**.
2. Enter `jakarta` in the alias field and then choose the path of the `isapi_redirect.dll` as the physical path.

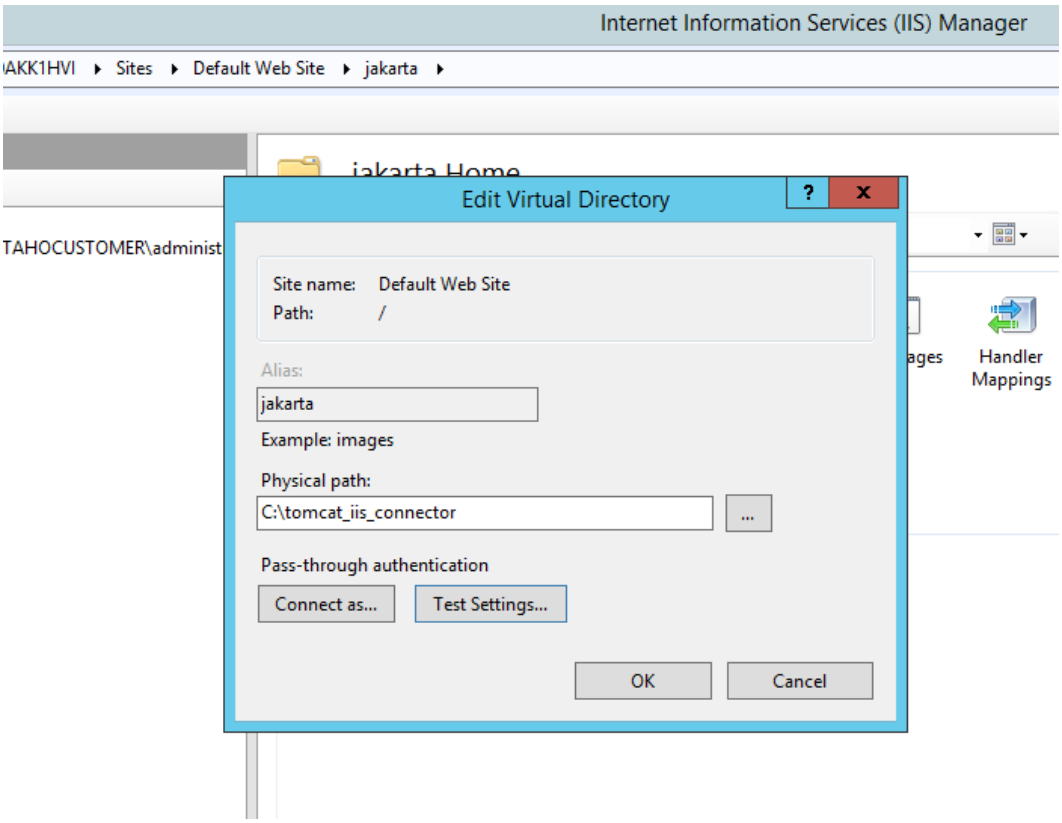

Figure 6: Edit Virtual Directory

3. Click **OK** then select the **Jakarta** virtual directory.
4. From the **Features** view, double-click on **Handler Mappings**, and select **ISAPI-dll**.
5. From the **Actions** bar, select **Edit Features Permissions**, and make sure that **Read**, **Script**, and **Execute** are selected.

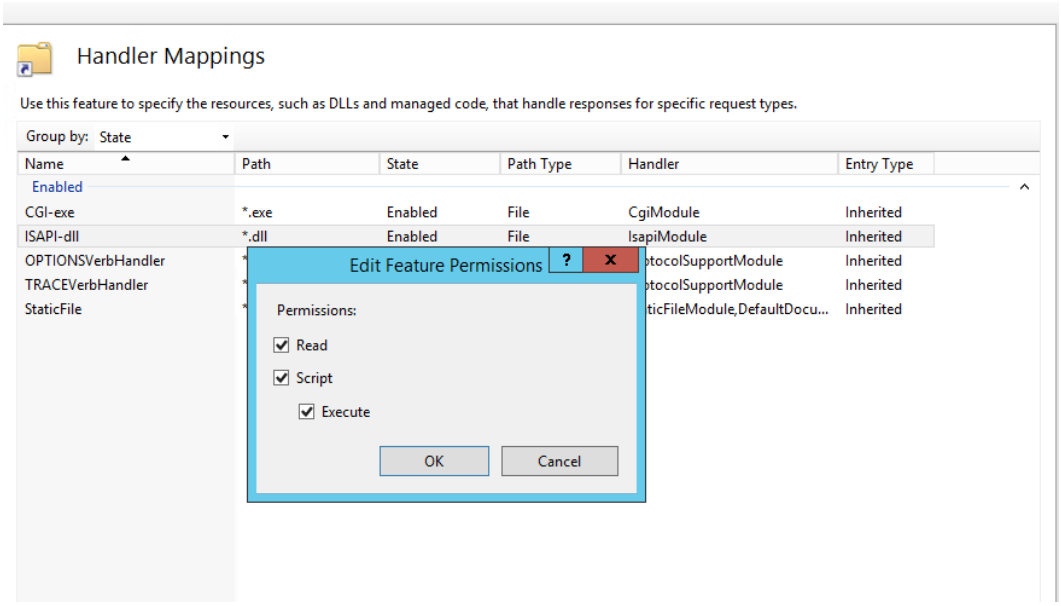

Figure 7: Handler Mappings

Enable Windows Authentication

Here are the steps for enabling Windows authentication:

1. On **IIS Manager**, select the default website.
2. Double-click on **Authentication**.
3. Make sure **Anonymous Authentication** is disabled.
4. Make sure **Windows Authentication** is enabled.

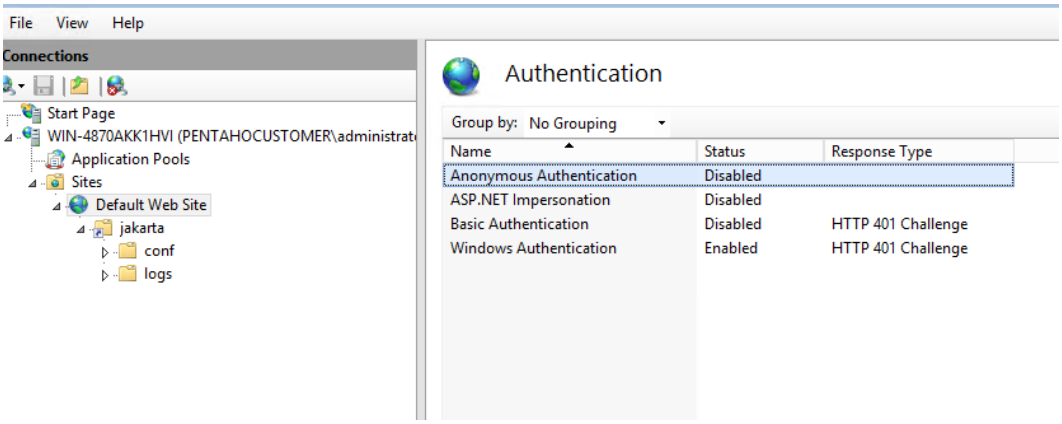

Figure 8: Default Website Authentication

5. On **IIS Manager**, select the **Jakarta** virtual directory.
6. Double-click on **Authentication**.
7. Make sure **Anonymous Authentication** is disabled.
8. Make sure **Windows Authentication** is enabled.

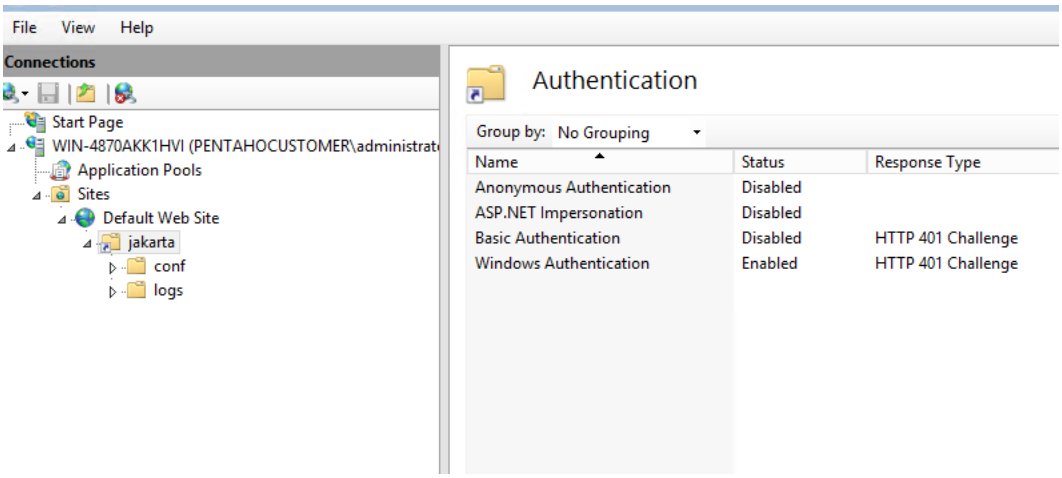

Figure 9: Jakarta Authentication

Configure IIS to Handle Special Characters

Here are the steps to configure IIS to handle special characters:

1. Open **IIS Manager**.
2. Click on the default website.
3. Click on **Actions**, and then select **Explore**. It should open in the `C:\inetpub\wwwroot` folder where you will find the `web.config` file.
4. Add the following entry under `<system.webServer>`:

```
<security>
<requestFiltering allowDoubleEscaping="true"></requestFiltering>
</security>
```

5. Add the following entry outside `<system.webServer>`:

```
<system.web>
<httpRuntime requestPathInvalidCharacters="" />
</system.web>
```

6. Save the `web.config`.
7. Restart IIS on the default website.

Pentaho Configurations

This section contains the configuration steps for your Pentaho installation. You can find details on these topics in the following sections:

Set Tomcat Authentication to False

Follow these steps to set `tomcatAuthentication` to `False`:

1. Locate the following file: `Pentaho-server/tomcat/conf/server.xml`
2. Locate the `AJP connector` entry and change it from:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

    <Connector URIEncoding="UTF-8" port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

   To this:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->

    <Connector URIEncoding="UTF-8" port="8009" protocol="AJP/1.3"
enableLookups=”false” tomcatAuthentication=”false” redirectPort="8443" />
```

Enable Pre-Authenticated Processing Filter

These steps will help you enable `preAuthenticatedProcessingFilter`:

1. Locate the following file: `\pentaho-server\pentaho-solutions\system\applicationContext-spring-security.xml`
2. Locate the `filterChainProxy` and replace the patterns using this text:

```
        <sec:filter-chain pattern="/api/repos/dashboards/print"
filters="securityContextHolderAwareRequestFilter,httpSessionPentahoSessionC
ontextIntegrationFilter,httpSessionContextIntegrationFilter,preAuthenticate
dSecurityFilter,httpSessionReuseDetectionFilter,logoutFilter,authentication
ProcessingFilter,basicProcessingFilter,requestParameterProcessingFilter,ano
nymousProcessingFilter,sessionMgmtFilter,exceptionTranslationFilter,filterI
nvocationInterceptor" />

        <sec:filter-chain pattern="/webservices/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,preAuthenticatedProcessingFilter,basicProcessingFilter,
anonymousProcessingFilter,sessionMgmtFilter,securityContextHolderAwareReque
stFilterForWS,exceptionTranslationFilterForWS,filterInvocationInterceptorFo
rWS" />

        <sec:filter-chain pattern="/api/repos/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,preAuthenticatedProcessingFilter,basicProcessingFilter,
requestParameterProcessingFilter,anonymousProcessingFilter,sessionMgmtFilte
r,securityContextHolderAwareRequestFilterForWS,exceptionTranslationFilterFo
rWS,filterInvocationInterceptorForWS,preFlightFilter" />

        <sec:filter-chain pattern="/api/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,preAuthenticatedProcessingFilter,basicProcessingFilter,
requestParameterProcessingFilter,anonymousProcessingFilter,sessionMgmtFilte
r,securityContextHolderAwareRequestFilterForWS,exceptionTranslationFilterFo
rWS,filterInvocationInterceptorForWS" />

        <sec:filter-chain pattern="/plugin/reporting/api/jobs/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,preAuthenticatedProcessingFilter,basicProcessingFilter,
requestParameterProcessingFilter,anonymousProcessingFilter,sessionMgmtFilte
r,securityContextHolderAwareRequestFilterForWS,exceptionTranslationFilterFo
rWS,filterInvocationInterceptorForWS,preFlightFilter" />

        <sec:filter-chain pattern="/plugin/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,preAuthenticatedProcessingFilter,basicProcessingFilter,
requestParameterProcessingFilter,anonymousProcessingFilter,sessionMgmtFilte
r,securityContextHolderAwareRequestFilterForWS,exceptionTranslationFilterFo
rWS,filterInvocationInterceptorForWS" />

        <sec:filter-chain pattern="/**"
filters="httpSessionPentahoSessionContextIntegrationFilter,httpSessionConte
xtIntegrationFilter,httpSessionReuseDetectionFilter,logoutFilter,preAuthent
icatedProcessingFilter,authenticationProcessingFilter,basicProcessingFilter
,requestParameterProcessingFilter,anonymousProcessingFilter,sessionMgmtFilt
er,securityContextHolderAwareRequestFilter,exceptionTranslationFilter,filte
rInvocationInterceptor" />
```

Configure Authentication Manager

Follow these steps to configure `authenticationManager`:

1. Locate the following file: `\pentaho-server\pentaho-solutions\system\applicationContext-spring-security.xml`
2. Replace the following `authenticationManager`:

```
  <bean id="authenticationManager"
class="org.springframework.security.authentication.ProviderManager">
    <constructor-arg>
      <util:list>
        <pen:bean
class="org.springframework.security.authentication.AuthenticationProvider"/
>
        <ref bean="anonymousAuthenticationProvider" />
      </util:list>
    </constructor-arg>
    <property name="authenticationEventPublisher">
      <ref bean="defaultAuthenticationEventPublisher" />
    </property>
  </bean>
```

3. Add the following `authenticationManager`:

```
<!-- ================ AUTHENTICATION ======================= -->
  <bean id="authenticationManager"
class="org.springframework.security.authentication.ProviderManager">
    <constructor-arg>
      <util:list>
            <pen:bean
class="org.springframework.security.authentication.AuthenticationProvider">
       <ref bean="preAuthAuthenticationProvider" />
       <pen:attributes>
       <pen:attr key="providerName" value="${security.provider}"/>
       </pen:attributes>
       </pen:bean>
<ref bean="anonymousAuthenticationProvider" />
      </util:list>
    </constructor-arg>
  </bean>
```

Configure Pre-Authenticated Processing Filter

In the same file, add the following bean for `preAuthenticatedProcessingFilter`:

```
<!--- IWA --->
      <bean id="preAuthenticatedProcessingFilter"
      class="org.pentaho.platform.web.http.security.UsernameSubstringPreAuthenticatedProcessingFilter">
            <property name="authenticationManager">
                  <ref bean="authenticationManager"/>
            </property>
            <property name="regex" value=".+\\(.+)"/>
      </bean>
```

Configure Pre-Authentication Provider

In the same file, add the following bean for `preAuthAuthenticationProvider`, just below the `preAuthenticationProcessingFilter`:

```
bean id="preAuthAuthenticationProvider"
class="org.springframework.security.web.authentication.preauth.PreAuthenticatedAuthenticationProvider" >
      <property name="preAuthenticatedUserDetailsService">
            <bean id="userDetailsServiceWrapper"
class="org.springframework.security.core.userdetails.UserDetailsByNameServiceWrapper">
                  <property name="userDetailsService"
ref="ldapUserDetailsService" />
            </bean>
      </property>
  </bean>
```

Configure Exception Translation Filter

Here are the steps for configuring `exceptionTranslationFilter`:

1. Locate the following file: `\pentaho-server\pentaho-solutions\system\applicationContext-spring-security.xml`
2. Locate the following bean, `exceptionTranslationFilter`:

```
  <bean id="exceptionTranslationFilter"
class="org.springframework.security.web.access.ExceptionTranslationFilter">
    <constructor-arg ref="authenticationProcessingFilterEntryPoint"/>
    <property name="accessDeniedHandler">
      <bean
class="org.springframework.security.web.access.AccessDeniedHandlerImpl" />
    </property>
  </bean>
```

3. Replace it with the following bean declaration:

```
<bean id="exceptionTranslationFilter"
class="org.springframework.security.web.access.ExceptionTranslationFilter">
      <constructor-arg ref="preAuthenticatedProcessingFilterEntryPoint" />
    <property name="accessDeniedHandler">
      <bean
class="org.springframework.security.web.access.AccessDeniedHandlerImpl" />
    </property>
  </bean>
```

4. Add the following bean, below the `exceptionTranslationFitler`:

```
<bean id="preAuthenticatedProcessingFilterEntryPoint"
class="org.springframework.security.web.authentication.Http403ForbiddenEntr
yPoint" />
```

Configure Exception Translation Filter for WS

These steps will help you configure `exceptionTranslationFilterForWS`:

1. Locate the following bean:

```
<bean id="exceptionTranslationFilterForWS"
class="org.springframework.security.web.access.ExceptionTranslationFilter">
    <constructor-arg ref="basicProcessingFilterEntryPoint"/>
    <property name="accessDeniedHandler">
      <bean
class="org.springframework.security.web.access.AccessDeniedHandlerImpl" />
    </property>
  </bean>
```

2. Replace it with the following bean:

```
<bean id="exceptionTranslationFilterForWS"
class="org.springframework.security.web.access.ExceptionTranslationFilter">
      <constructor-arg ref="preAuthenticatedProcessingFilterEntryPoint" />
    <property name="accessDeniedHandler">
      <bean
class="org.springframework.security.web.access.AccessDeniedHandlerImpl" />
    </property>
  </bean>
```

3. Save your changes.
4. Restart Pentaho Server.
5. Test the changes. You should be able to log into the URL served by IIS. Go to Internet Explorer and type the following URL: `http://localhost/pentaho/`.

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- **Apache Tomcat**
  - Download Tomcat Connectors
- **Pentaho Documentation**
  - Components Reference
  - Switch to Integrated Windows Authentication
- **Spring**
  - Spring Framework

Finalization Checklist

This checklist is designed to be added to any implemented project that uses this collection of best practices, to verify that all items have been considered and reviews have been performed.

Name of the Project:_____

Date of the Review:_____

Name of the Reviewer:_____

Item	Response	Comments
Did you install and configure the Tomcat IIS Connector?	YES_____ NO_____	
Did you configure the IIS role?	YES_____ NO_____	
Did you configure your Pentaho software?	YES_____ NO_____	