Administrator’s Guide

Planon Software Suite

Version: L105

Planon

Building Connections

About this Document

Intended Audience

This document is intended for *Planon Software Suite* users.

Contacting us

If you have any comments or questions regarding this document, please send them to: support@planonsoftware.com.

Document Conventions

Bold
Names of menus, options, tabs, fields and buttons are displayed in bold type.

Italic text
Application names are displayed in italics.

CAPITALS
Names of keys are displayed in upper case.

Special symbols

ℹ	Text preceded by this symbol references additional information or a tip.
❗	Text preceded by this symbol is intended to alert users about consequences if they carry out a particular action in Planon.

Table of Contents

About Planon Software Suite Administrator’s Guide

This document describes the additional configuration steps for Planon Software Suite. It applies to on-premise installations only and provides several configuration file examples.

For Cloud-specific configuration, see Cloud configuration.

Important information:

- When copying examples from this document, paste them in a text editor first so that no additional characters are copied which may corrupt the configuration files. Make sure that for example CLI commands are on one line.
- As part of the installation/configuration, a number of configuration files need to be amended manually. Use only a UTF-8 compliant text editor if not the files will get corrupted and the installation/configuration will fail.
- In the code snippets, parameters that need to be replaced with your own values are displayed in bold and italic.

WildFly Command line interface

The Command Line Interface (CLI) is the management tool for the WildFly application server. It allows users to connect to the application server and execute management operations available through the detyped management model. The CLI is launched using the jboss-cli.bat located in the WildFly bin directory.

For WildFly, all configuration is in the standalone-full.xml, located in …\Server\wildfly-*\standalone\configuration. Use the CLI to update the configuration. The tool can connect online or offline. If you run the tool online, it will indicate when a restart is needed to get the changes applied. The tool also has a GUI interface. Do not forget to set JAVA_HOME, the CLI needs this variable.

Connect offline:

Start the jboss-cli.bat without parameters. Enter the following command:

```
embed-server -c=standalone-full.xml
```

If connected start your configuration. Tab completion is supported for all commands and options, i.e. node types and node names, operation names and parameter names.

For connecting online, you need to know the management http port. For more information on this subject, refer to Port set configuration. You can also find the port in the installation_summary.txt located in \.install4j

```
jboss-cli.bat --connect --controller=localhost:port
```

Connect online in GUI modes:

```
jboss-cli.bat --connect --controller=localhost:port --gui
```

For details on how to use the command line interface check the WildFly administrator’s guide, see Command Line Interface

Configuring the login language

When logging on to the application, the login screen is displayed to the user. Typically, this is displayed in the language of the server's locale. If required, you can configure the language that is initially displayed in the login screen.

> This is only possible for on-premise installations.

Procedure

1. In Windows explorer, go to the following location in your installation folder: ../server/tanuki/webserver/conf.
2. Open tomcat-wrapper-default.conf in a text editor and locate the **# Java Additional Parameters** section.
3. Add the following parameters (and increment the sequence number):

   -Duser.language=nl

   -Duser.country=NL

> - You can change the language and country as required.
> - For a list of locales, see .
> - By using this configuration, you are setting the default locale!

   The effect of this setting is that during login, the language configured here will be displayed to the user.

4. Restart the server to make your changes effective.

   The login screen will be displayed in the configured language. Once logged on, users will see the application in the language that is configured for them.

Server Hardening

By default, Planon Software Suite installation installs a secure application- and a web server. To further improve the server security, the following precautions can be adopted. Note that some of these configurations can change the functional behavior of the Planon Software Suite.
Depending on the infrastructure of your organization, you may configure the following options:

Configure SSL connections

You can further improve the server security by configuring the following options:

HTTPS communication

HTTPS with the Suite Installer

By default (if not disabled during installation) the Suite installer will install HTTPS communication between the clients and the web server. The installation will follow the secure configuration as delivered by default with Tomcat. For details on used ciphers and SSL protocol see: https://tomcat.apache.org/tomcat-9.0-doc/config/http.html. If you need a more secure setup, update the ciphers (see Selecting SSL cipher for details) or the sslProtocol manually to the desired values.

For information about certificates and keystores, see Handling certificates and keystores. Use the FQN for the CN when creating a key, keystore, and/or certificate.

Selecting SSL cipher

It is possible to select the ciphers that are used by the web server's SSL HTTP connector.

To select the ciphers used by the SSL HTTP connector, proceed as follows:

1. Go to ..\Server\tomcat-*\ conf\server.xml
2. Locate the connector that has SSLEnabled="true", and add the attribute ciphers="".

   The values are JSSE cipher names of the ciphers, separated by commas. The best way to configure this depends on your security policy, we cannot advise on that on up front.

> By default, the default ciphers for the JVM will be used less those considered to be insecure.

A good example of secure configuration can be found at the OWASP website see the following link: https://www.owasp.org/index.php/Securing_tomcat#Encryption It can be that a configuration described in the OWASP setup does not support all platforms supported by the Planon software. Check the *Supported configurations* for your setup to verify if it fits.

HTTPS for communication between the application server and the web server

First configure the SSL connector for the application server.

1. Stop the application server.
2. Start the CLI offline.
3. Update the parameters and run the following command:

By default we expect the keystore in the WildFly configuration directory (...\Server\wildfly-*\standalone\configuration), so you only need to provide the keystore name in the keystore-path parameter. If you want to use another location, provide the full path in the keystore-path parameter and remove the keystore-relative-to parameter from the command.

```
/core-service=management/security-realm=PlanonRealm/server-identity=ssl:add(keystore-
path="yourKeystore.jks", keystore-relative-to="jboss.server.config.dir", keystore-
password="changeit", alias="yourAlias", key-password="changeit")
```

4. Run the following command:

```
/subsystem=undertow/server=default-server/https-listener=HTTPs/:add(security-
realm=PlanonRealm, tcp-keep-alive=true, disallowed-
methods=[PUT,DELETE,OPTIONS,HEAD,TRACE,CONNECT,PATCH], max-post-
size=1024000000, socket-binding=https)
```

5. Run the following command to update the remoting connector to use the HTTPs connector:

```
/subsystem=remoting/http-connector=http-remoting-connector/:write-attribute(name=connector-
ref,value=HTTPs)
```

6. (Re)start the application server.

Client side changes

Next update the web server.

Procedure

1. You must change the remoting.xml located in ...\Server\tomcat-*\conf". Change the URL property to reflect the changes in protocol and port number. Make sure you fill in the correct hostname and port number.

```
<url>https://hostname:port/rest</url>
```

2. Add your certificate to the client cacerts keystore if needed (for example self-signed certificate). To do so use the following command:

The keytool can be found in the following location ...\Server\jdk-*\bin

```
keytool -cacerts -import -trustcacerts -noprompt -file path\to\your\certificate.crt -alias yourAlias -storepass changeit
```

Disabling HTTP for the application server

1. Stop the application server service.
2. Execute the following commands to remove the HTTP listeners:

```
/subsystem=undertow/server=default-server/http-listener=default/:remove
```

3. Remove the HTTP connector:

```
/socket-binding-group=standard-sockets/socket-binding=http/:remove
```

4. Start the application server service.

Troubleshooting

Question	Answer
I keep getting the following message in my logs, how do I fix this? javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target	Java cannot find your truststore file; make sure you supplied the correct VM-arguments to the web server).

Question	Answer
My (application server) server keeps complaining about the HTTPS hostname being incorrect! How do I fix this?	Your certificate & keystore do not have the FQN (Fully Qualified name) – <HostName>.<DomainName>. By default, the application server tries to perform a few "sanity" checks upon the SSL-connectors' URL before accepting a connection. When the FQN is part of the keystore, the application server will no longer complain.
The application server seems to work, but the client won't connect! It complains about the hostname of the server not being found!	Probably your servers' certificate has a common name (CN) that does not correspond to a hostname that can be resolved through DNS. Add the hostname of the application server to DNS.
How do I detect whether my application server uses a private hostname?	Try the command nslookup <hostname> in a command prompt. If the result of this command is the IP-address of your application server server, it probably has a public hostname. If it gives an error, like non-existent domain or such, your application server has a private hostname.

Strict-Transport-Security

This is an additional security configuration when HTTPs is configured.

If enabled:

- Switching back to HTTP is not possible anymore. Once a supported browser receives this header that browser will prevent any communications from being sent over HTTP to the specified domain and will instead send all communications over HTTPS.
- Disable the HTTP connector

To enable Strict-Transport-Security, add the following valve to ... \Server\tomcat-*\conf \server.xml:

```
<Valve className="nl.planon.tomcat.AddHeaderValve" name="Strict-Transport-Security"
 value="max-age=31536000"/>
```

For more details about this header see: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security

Database server data encryption

Planon Software Suite supports:

- SQL Server transparent data encryption for versions 2008 and higher. Note that this is possible only for the Enterprise Editions of SQL Server.
- Transparent Database Encryption is supported for version 11.2 and higher. Note that this is only true for Oracle EE.

Secure communication with the database server

This section describes how to set up a secure communication between the application server and the database server.

You can do so by enabling an encrypted connection for the connection URL of the database server.

Procedure

1. Stop the application server.
2. Open JBoss CLI.
3. To enable encryption in the connection URL, run the following command:

```
/subsystem=datasources/data-source=PlanonDS/:write-attribute(name=connection-
url,value="jdbc:sqlserver://databasehostname.fullyqualifieddomainname\
\databaseserverinstancename;encrypt=true;trustServerCertificate=false")
```

4. Export the database server certificate without its private key to a certificate file.
5. Add the certificate to the client cacerts keystore if needed (for example: self-signed certificate). To do so, use the following command:

You can find the keytool in the following location ...\Server\jdk-*\bin

```
keytool -cacerts -import -trustcacerts -noprompt -file path\to\your\certificate.crt -
alias yourAlias -storepass changeit
```

You have now set up a secure communication with your database server. You can now safely restart your application server.

Protecting against clickjacking

With the upcoming deprecation of third party cookie support by Google (see Privacy Sandbox for the Web for more information), the Planon Self-Service integration support will become deprecated per January 1st 2024. To maintain the same functionality, the link should be opened in a new browser tab instead of using an iframe.

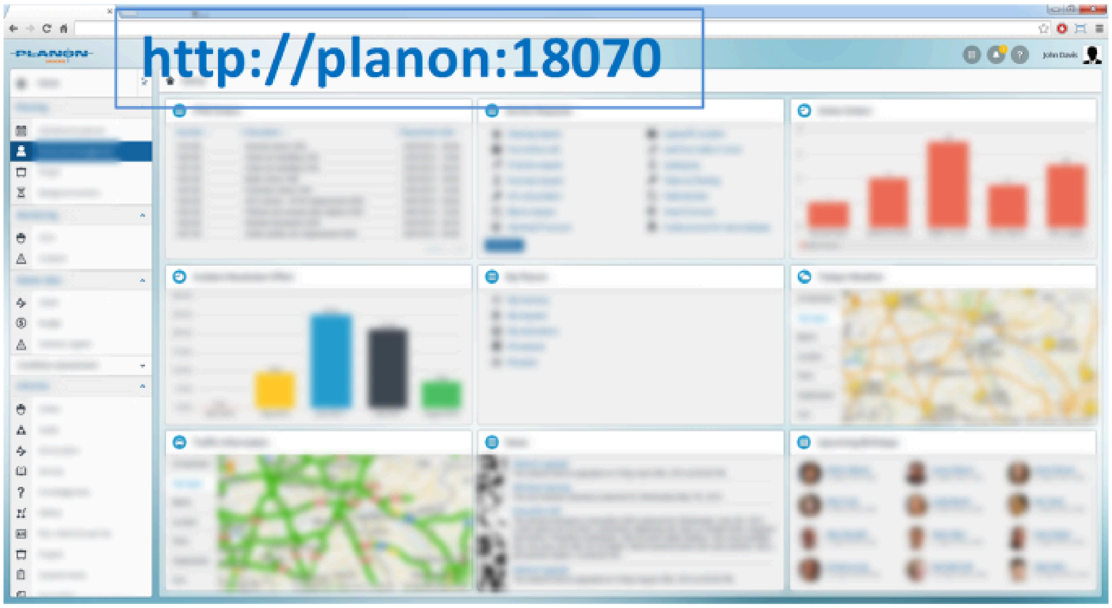
By default, the Planon Web Client is protected against clickjacking. Clickjacking is used to trick the user in clicking and performing an action on a site or application. This is usually done by hiding the real contents by putting a layer on top of the application.

There are two generic implementations to prevent clickjacking: "X-Frame-Options" and "CSP2 header frame-ancestor". They cannot be used out of the box, because both do not support the full set of browsers.

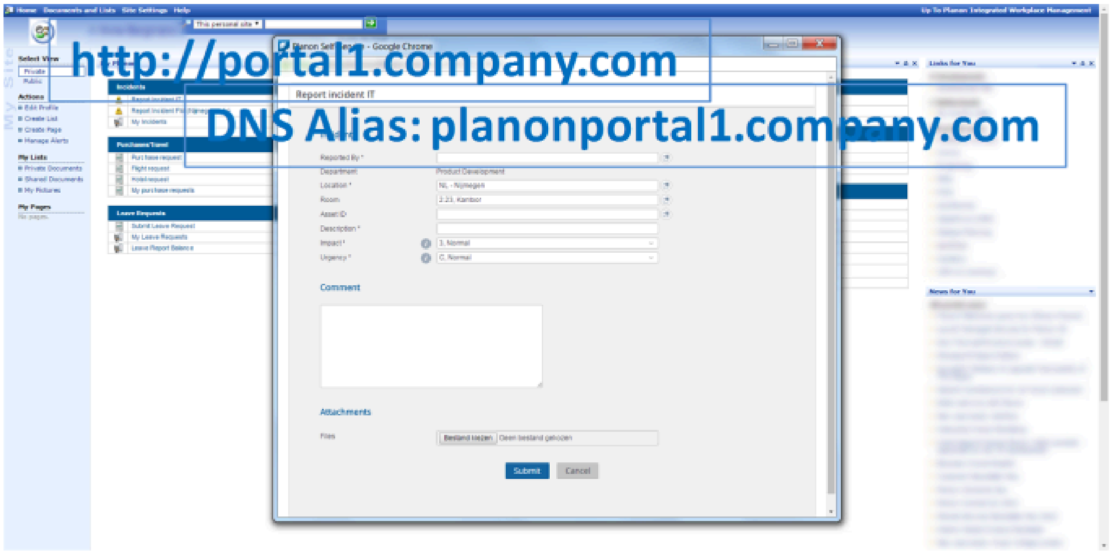
To solve this, Planon implemented a Valve on the Web server, the `ClickjackHostValve` valve. This valve is specified in the `server.xml` located in `\Server\tomcat-*\conf`. It sets runtime the "X-Frame-Options" and the "CSP2 header frame-ancestor" in the header with the correct properties based on the incoming URL.

By default Planon is installed with Planon Web Client setup, allowing only the Web server itself. For Planon Self-Service portal integration, additional configuration is needed. Every portal for which integration is needed requires a DNS alias.

Default installation: Planon Web Client setup



Alternative installation: Planon Self-Service integration setup


http://portal1.company.com
DNS Alias: planonportal1.company.com
Planon Self-Service - Google Chrome
Report incident IT
Comment
Attachments
Submit
Cancel

Reported By *

Department: Product Development

Location *: NL - Nijmegen

Room: 2.21, Kantoor

Asset ID

Description *

Impact *: 3. Normal

Urgency *: C. Normal

Files: Bestand kiezen Geen bestand gekozen

For more information about clickjacking in general see Clickjacking. For web applications the issue mainly concerns the IFrame elements.

Portal integration configuration

1. Define DNS aliases for every server for which portal integration is needed.
2. Add a file with the name "portals.txt in …\Server\tomcat-*\conf.

   The file must be filled with the DNS aliases and the corresponding URLs.

```
DNS Alias=Portal URL
```

For example:

```
planonportal1.company.com=http://portal1.company.com

planonportal2.company.com=http://portal2.company.com
```

If an incoming URL is in this file, Planon will allow framing from this location. All other URLs that come in should be from same origin, otherwise framing is blocked.

The file will be picked up automatically. A restart of the Web server is not needed.

Reverse proxy

In a simple web server deployment the HTTP request header is used to retrieve the host name. But when a reverse proxy is used, it may be necessary to set the host. If needed add the alternativeHostHeader parameter to the ClickjackHostValve valve. See the documentation of your reverse proxy manufacturer to find out which header is set.

Absolute URLs

If a Planon Self-Service page contains an absolute URL pointing to Planon (e.g. a redirect URL) and is needed in both Planon Web Client and Planon Self-Service portal integration, the web definition must be duplicated. It is not possible to mix the URLs. For example, in planonwebclient.company.com it is not possible to open an IFrame from planonportal.company.com. This is not allowed by the ClickJacking policies.

HTTPs

In case of multiple hosts, it is probable that multiple certificates are needed. But Tomcat only supports one certificate per HTTPs port, which means two web servers are needed in case of HTTPs. Or use one of the following alternatives:

- Front the Tomcat with a reverse proxy that is able to do SNI (Server Name Indication). SNI fixes this issue for you.
- Use different ports for the SSL configuration. So the customer would have one Tomcat server listening on two ports for HTTPS connections.
- Use a wildcard certificate.

Troubleshooting

Question	Answer
For portal integration, nothing is displayed in the popup.	Probably a misconfiguration. Check with the developer toolbar what the page is complaining about.

Hiding detailed login messages

By default, Planon Software Suite provides extra information when a login fails. For example:

- Account expired
- Account locked
- Server in upgrade mode

To hide the detailed login messages, add the following parameters to the Java Additional Parameters of the application server service wrapper configuration file located in ...\Server\tanuki\appserver\conf:

```
-DplanonDiscloseHTTPError=false

-DplanonDiscloseLoginFailureReason=false
```

ℹ If you do not have the above parameters, this is handled as if the parameters are set to **True**.

Encrypt database password

You can encrypt the database password in the data source with standard WildFly functionality using a security domain and picketbox. Execute the steps below:

1. Open the command prompt in the following location ...\Server\wildfly-*.
2. Encrypt the password using the following command, you can select any password.

   Make sure to replace the jdk-* (lookup \Server) and picketbox-* (lookup '...\wildfly-*\modules\system\layers\base\org\picketbox\main) with the versions of your installation:

   ```
   <java installation location ..\Server\jdk-*\bin\>java -cp modules\system\layers\base\org\picketbox
   \main\picketbox-*.Final.jar

   org.picketbox.datasource.security.SecureIdentityLoginModule Plan$QL
   ```
3. Start the CLI.
4. Add the EncryptDBPassword security domain:

   ```
   /subsystem=security/security-domain=EncryptDBPassword:add()
   ```
5. Add the login module to the security domain:

   ```
   /subsystem=security/security-domain=EncryptDBPassword/authentication=classic:add(login-
   modules=[


   {"code"=>"org.picketbox.datasource.security.SecureIdentityLoginModule", "flag"=>"required",
    "module-options"=>[("username"=>"YourUserName"), ("password"=>"EncryptedPassword"),
    ("managedConnectionFactoryName"=>"jboss.jca:service=LocalTxCM,name=PlanonDS")]}

   ])
   ```
6. Remove the following attributes using CLI; they will be replaced with the security domain.

   ```
   /subsystem=datasources/data-source=PlanonDS/:undefine-attribute(name=password)

   /subsystem=datasources/data-source=PlanonDS/:undefine-attribute(name=user-name)
   ```
7. Add the security domain:

   ```
   /subsystem=datasources/data-source=PlanonDS/:write-attribute(name=security-domain,

   value=EncryptDBPassword)
   ```

Encrypt passwords in Tomcat-users.xml

1. Open the server.xml located in …\Server\tomcat-*\conf\.
2. Add a CrendentialHandler to the following realm:

   org.apache.catalina.realm.UserDatabaseRealm. Should be as follows, replace the bold part with for example SHA-512, SHA-256 or MD5:

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
 resourceName="UserDatabase">
```

```
<CredentialHandler className="org.apache.catalina.realm.MessageDigestCredentialHandler"
 algorithm="SHA-256"/>
```

```
</Realm>
```

3. Start the command prompt in …\Server\tomcat-*\bin\ .
4. Execute the following command to generate the HASH for the password. Replace SHA-256 with the algorithm used in the CrendentialHandler and PASSWORD with the password in tomcat-users.xml (located in …\Server\tomcat-*\conf \):

```
digest.bat -a SHA-256 -h
 org.apache.catalina.realm.MessageDigestCredentialHandler PASSWORD
```

5. Replace the password for the user in question in the tomcat-users.xml with the encrypted password just generated.
6. Restart the web server.

For more configuration options about password encryption, see the Tomcat guide: https://tomcat.apache.org/tomcat-9.0-doc/realm-howto.html#Digested_Passwords.

Secure file locations

For extra security, set the **Allow user-defined path** system setting to **No**. This allows users to only store documents in the default file location folder or subfolders of that file location.

For details about system settings, please refer to the *System Settings* documentation.

Restricting the use of file types

For extra security you can restrict the use of file types. The following systems settings should be configured:

- For secure file location, enter the allowed types in the **File types** field.
- For file types Planon users are allowed to upload, enter the allowed types in the **Allowed file types** field.

For details about system settings, please refer to the System Settings documentation.

Configuring Planon Self-Service

Uploading documents can be additionally secured for file types and size.

For more information, refer to the Self-Service Configuration Guide, Chapter "Configuring Planon Self-Service", section "Uploading documents".

Performance

Monitoring performance

You can use visual VM to check the performance of the application:

Application server

1. Install visual vm: https://visualvm.github.io/.
2. Open the command prompt in the following location: ...\Server\tanuki\appserver\bin
3. To use VisualVM for WildFly, WildFly should be started from the desktop. VisualVM cannot be used with WildFly running as a service. Run the following command: wrapper -c ..\conf\jboss-wrapper-default.conf
4. Open a new command prompt.
5. Start ..\visualvm_*\bin\visualvm --jdkhome ..\Server\jdk-*

WilfFly will be automatically displayed in the list.

6. Check the statistics.

Webserver

1. Install visual vm: https://visualvm.github.io/.
2. Open the tanuki configuration file of the webserver.
3. Add the following parameters:

```
wrapper.java.additional.nr=-Dcom.sun.management.jmxremote=true

wrapper.java.additional.nr=-Dcom.sun.management.jmxremote.port=12346

wrapper.java.additional.nr=-Dcom.sun.management.jmxremote.ssl=false

wrapper.java.additional.nr=-Dcom.sun.management.jmxremote.authenticate=false
```

4. Restart the webserver.
5. Open a command prompt.

6. Start ..\visualvm_*\bin\visualvm --jdkhome ..\Server\jdk-*
7. Create a local JMX connection in jvisualvm to localhost:12346
8. Check the statistics.

Compression default enabled on the web server

Performance compression is enabled by default on the web server. See the HTTP connector in `server.xml` located in `\Server\tomcat-*\conf`.

```
<Connector URIEncoding="UTF-8" compression="on" connectionTimeout="20000"
 maxHttpHeaderSize="16384" port="18170" protocol="HTTP/1.1" redirectPort="8443"
 server="webserver"/>
```

When you are close to the web server and CPU cycles are important, the compression slows down the application. You can disable it if required.

For more information, see Apache Tomcat 9 Configuration Reference.

Additional Scheduler settings

In addition to the basic Scheduler settings, the following parameters are available for fine-tuning Scheduler operations. We advise to run the Scheduler on a separate application server, if you want to deviate from the default settings. For a description of these fields, refer to the table below.

Parameter	Description
MaxCached Sessions PerAccount (nl.planon.maxcachedsessionsperaccount)	Sets the maximum number of cached sessions per account. The default value is 5 which can be extended to maximum 25.
MaxConcurrentJobs (nl.planon.maxconcurrentjobs)	The maximum number of concurrent jobs to be used by the scheduler. The default value is 10 which can be extended up to maximum 50 with 25 being an optimum value. If the value exceeds 25, it needs to be tested in the customer environment to check for issues, if any. The optimal value depends on the deployment and the usage of the products.

Applying the additional Scheduler settings

To apply these settings open the jboss-wrapper-default.conf file which is available in: ...\Server\tanuki\appserver\conf

1. Open jboss-wrapper-default.conf in a text editor
2. Under # Java Additional Parameters, add the parameters (digits are examples):

```
wrapper.java.additional.nr=-Dnl.planon.maxcachedsessionsperaccount=4

wrapper.java.additional.nr=-Dnl.planon.maxconcurrentjobs=3
```

Garbage collection

Planon uses the G1 Garbage collection. This is also the default value of the OpenJDK at this moment, but we do not fall back on the default, we specifically set it to G1.

The following java additional parameter is set in the Tanuki configuration files of the web server and the application server:

```
wrapper.java.additional.nr=-XX:+UseG1GC
```

For details see: https://wiki.openjdk.java.net/display/HotSpot/Garbage+Collection.

Creating a java heap dump for a java process

A heapdump is a snapshot of the memory of the application at a specific moment in time. It can be very useful as diagnostic information when memory issues occur. However, using a heapdump without knowing what to look for, is difficult. It is like searching for a needle in a haystack.

Automatically generating a heap dump

Add the following parameters to your Tanuki configuration file to automatically create a heap dump when Java goes out of memory:

```
wrapper.java.additional.xx=-XX:+HeapDumpOnOutOfMemoryError

wrapper.java.additional.xx=-XX:HeapDumpPath=path to folder to save heap-dump
```

For more information see: https://openjdk.java.net/groups/hotspot/docs/RuntimeOverview.html.

Manually generating a heap dump

1. Open Task Manager on the machine where the java process is running.
2. From the list of running processes, locate the java process related to Planon.

The process name is “OpenJDK Platform binary”. Filter on the Command Line column to find it.

3. If PIDs are not displayed, add the column to the Task Manager
4. Remember or note down the PID of the Planon Java process.
5. Open the command prompt as administrator.
6. Execute the following command at the location: ..\Server\\jdk-* bin to create a .hprof dump:

```
jmap -dump:format=b,file=YourFileName.hprof PID
```

***YourFileName*.hprof: can include a file path e.g. c:\dump\myfile.hprof**

This file can be analysed with Eclipse mat or jProfiler.

Creating a thread dump

Application server

1. Go to `\Server\tanuki\appserver\bin`.
2. Open a command prompt and execute the command `wrapper.exe -d ...\Server\tanuki\appserver\conf\jboss-wrapper-default.conf`.

   The thread dump will be in your `wrapper-default.log`.

Web server

1. Go to `\Server\tanuki\webserver\bin`.
2. Open a command prompt and execute the command `wrapper.exe -d ...\Server\tanuki\webserver\conf\tomcat-wrapper-default.conf`.

   The thread dump will be in your `wrapper-default.log`.

Restarting the server

Restarting the application server

Restarting the application server is nothing more than restarting the service. Under normal operations no other actions are needed. But it can happen the server gets polluted due to network hiccups or other interruptions, which require a complete clean startup:

1. Stop the service.
2. Remove the following folders located in ...\Server\wildfly-* \standalone\
    - data\bundlecache
    - data\tx-object-store
    - tmp
    - data\tmp
3. Start the server.

> ❗ Do not delete the other folders in the data folder. These folders contain unprocessed data. If removed, it may lead to data loss. Only if the application server will not start at all anymore, you can also clean up these folders. But keep in mind unprocessed work will be gone.

Restarting Web server

Restarting the web server is nothing more than restarting the service. Under normal operations no other actions are needed. But it can happen the server gets polluted due to network hiccups or other interruptions, which require a complete clean startup.

1. Stop the service.
2. Remove the following folders located in ...\Server\tomcat-*:
    - temp
    - work
3. Start the server.

Port set configuration

The ports selected during installation are logged to the file `installation_summary.txt` located in the folder `\.install4j`.

Application server

Reading and writing port information is done with the WildFly Command line interface (CLI). For more information on this subject, refer to WildFly Command line interface.

In WildFly, every port has a base value which can be configured. Also there is one single offset-value, which is applied to all ports. This offset can be configured as well. The port the server listens on is base port + offset.

Below you can find some example CLI commands for reading and writing port values.

Read offset

```
/socket-binding-group=standard-sockets:read-attribute(name=port-offset)
```

Read all ports

```
/socket-binding-group=standard-sockets:read-children-resources(child-type=socket-binding)
```

Set port offset to 10000

```
/socket-binding-group=standard-sockets:write-attribute(name=port-offset, value=${jboss.socket.binding.port-offset:10000})
```

Set management http base port to 9090 (this is the port you use for connecting with the CLI in on-line mode)

```
/socket-binding-group=standard-sockets/socket-binding=management-http:write-attribute(name=port, value=${jboss.management.http.port:9090})
```

Set management https base port to 9093

```
/socket-binding-group=standard-sockets/socket-binding=management-http:write-attribute(name=port, value=${jboss.management.http.port:9093})
```

Set http base port to 8080

```
/socket-binding-group=standard-sockets/socket-binding=http:write-attribute(name=port, value=${jboss.http.port:8080})
```

Set https base port to 8443

```
/socket-binding-group=standard-sockets/socket-binding=https:write-attribute(name=port, value=${jboss.https.port:8443})
```

Set port value using Java system properties

The general format for port-values is ${systemproperty:default-value}. This means that the default value is used, except if the system property is set. You can, for example, change the port offset to 22222 by adding the following to the jboss-wrapperdefault.conf

```
wrapper.java.additional.nr=-D jboss.socket.binding.port-offset= 22222
```

NoteIn the application server log you can search for the phrase *listening on* to find out the ports that it is actually listening on.

Web server

Planon uses the standard ports configuration:

- The Web client(s) ports are defined in the server.xml file located in the server\tomcat-*\conf directory.
- Port to connect to the Application is defined in the remoting.xml located in the server\tomcat-*\conf directory.

> You can deviate from the ports installed with the installer. But a Tomcat update will revert your manual changes.

JAAS login modules

Planon Software Suite uses the Java Authentication and Authorization Service (JAAS) to authenticate users. All login modules used are based on JAAS. The order in which the login modules are configured combined with their value for the property “flag” is relevant for proper functioning of Planon. Customers may configure the strictness of the login module based on the following values, see table below.

Keep in mind that misconfiguration of the authentication modules may negatively affect the performance of your system, harm your system’s security, cause unpredictable authentication behavior, or may prevent users from logging on at all. In the chapters that are about authentication, other login modules are added to the default set and the advised value for the flag is indicated.

Flag value	Description
“requisite”	The Login Module is required to succeed. If it succeeds, authentication continues down the Login Module list. If it fails, control immediately returns to the application (authentication does not proceed down the Login Module list).
“required”	The Login Module is required to succeed. Whether it succeeds or fails, authentication still continues to proceed down the Login Module list.
“optional”	The Login Module is not required to succeed. Whether it succeeds or fails, authentication still continues to proceed down the Login Module list.
“sufficient”	The Login Module is not required to succeed. If it succeeds, control immediately returns to the application (authentication does not proceed down the Login Module list). If it fails, authentication continues down the Login Module list.

Resetting the Supervisor password

In the event of forgetting or losing it, on-premise customers can reset the Supervisor password.

For Cloud customers, this feature is currently not available. To reset the supervisor password for Cloud, please contact Planon Support.

Proceed as follows to reset the password of the Supervisor/Application manager.

Procedure

1. To reset the supervisor password, the following bundle should be installed first: com.planonsoftware.passwordreset.jar
2. This bundle is located in the following location:

   ..\related_components\manual_installation_resources\tools \PasswordReset

3. Create a file named password_reset.txt (UTF-8) in Tanuki/bin directory (start-up directory of the application server.)
   The file should contain a single line:
   <username> = <newpassword>
   Example: Supervisor=secret
4. Copy the bundle com.planonsoftware.passwordreset.jar to the following location:
   ...\Server\wildfly-*\standalone\bundles\planon.
   The bundle will be automatically installed. It reads the text file on start and resets the password.
   The password_reset.txt is deleted and a password_reset.log is created containing log information whether or not the resetting was successful.
5. To uninstall the bundle:

   Remove com.planonsoftware.passwordreset.jar.

If the password of the user is of the type that expires, the new password will expire in a day.
If the start date of the user is after the current date, the start date is set to current date.
If the end date is before current date + 7 days, the end date is set to current date + 7 days.

Extending transaction timeout settings

In Planon Software Suite, the transaction time limit is set as 5400 seconds (90 minutes) to perform a specific action. Sometimes you may require additional time to complete an action. The existing time may cause impediment in completing your task.
For example, while importing multiple drawings using a single CAD import definition, an error message is displayed stating -The transaction is not active.
In such instances, you can extend the transaction timeout settings temporarily.

To extend the transaction timeout:

Procedure

1. Open the CLI and run the following command:

```
/subsystem=transactions/:write-attribute(name=default-timeout,value=5400)
```

2. Restart the application server. The transaction timeout settings are modified.

> It is not recommended to increase the default Timeout value. The transaction time should be modified only for an incidental action that takes longer than 5400 seconds. After completing the action, the time should be reverted to 5400 seconds.

DTAP environment

DTAP stands for **D**evelopment, **T**est, **A**cceptance and **P**roduction environment. Planon offers the possibility to mark your installation even if it is not your production installation. This will be visualized in the GUI.

In the application server service configuration file located in ...\Server\tanuki\appserver\conf, add the following parameters.

```
wrapper.java.additional.nr=-Dcom.planonsoftware.notproductionmode=true

wrapper.java.additional.nr=-Dcom.planonsoftware.notproductionmode.description=

"Test Environment"
```

notproductionmode:

- True: The installation is marked as not production. In this case the GUI will display an identifying text
- False or not available: The installation is marked as production. No identification
- This information can also be retrieved from the SDK

notproductionmode.description:

- The text that will be displayed in the GUI
- Will only be applicable if the notproductionmode=true
- If empty, a default text is displayed: “Not production”

> By default, Scheduled tasks, Platform apps and user extensions are disabled on first restart of DTA (Development, Test and Acceptance) environments. With next restarts, they will remain activated.

Restoring an environment

What happens when restoring an environment?

1. Typically, when you restore a Production environment to D, T, or A, all scheduled events are disabled:
   - Platform apps
   - TMS

- Scheduled tasks

  Disabling these events is done to ensure that Production data is not compromised.
  However, a second verification takes place.

2. When restoring an environment in D, T, or A, the Environment description is verified.

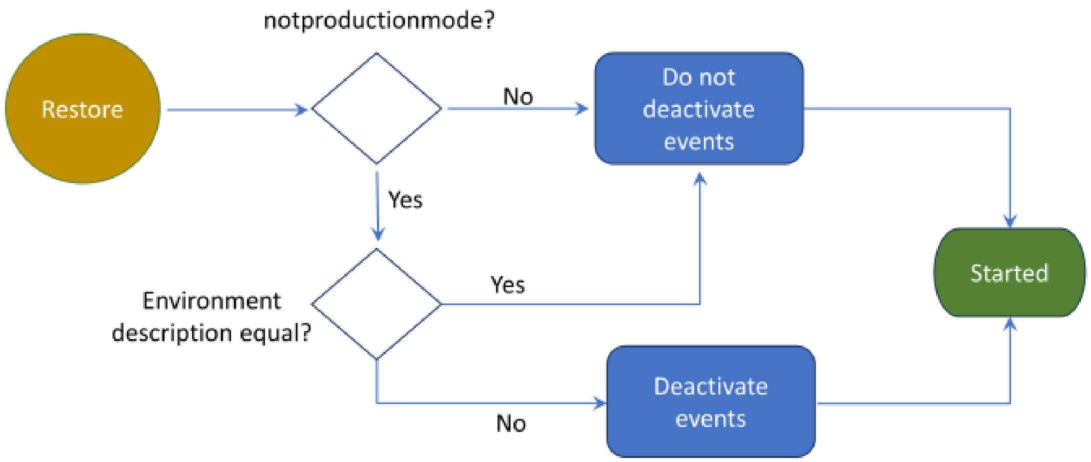
   - If the Environment mode is *not production* and if the Environment description is *equal*, the events are not deactivated.

     This implies that for the sake of convenience, these events persist and remain functional even after actions such as restarting, rebuilding, restoring, or upgrading the system.

> When restoring an environment with the same environment mode from another URL (environment name differs), the apps are set to *Inactive*.

   - If the Environment mode is *not production* and if the Environment description is *not equal*, the events are deactivated.

Graphically, this is illustrated as follows:



In a table, this is illustrated as follows:

Restore	notproductionmode	notproductionmode description	Deactivate?
Backup to Production	No	Equal	No
Backup to Production	No	No equal	No
Backup to DTA	Yes	Equal	No

Restore	notproductionmode	notproductionmode description	Deactivate?
Backup to DTA	Yes	Not equal	**Yes**

WildFly cluster

On-premise customers who have installed a WildFly cluster must ensure that the *JGROUPSPING* table is emptied when a database is copied from one environment to another. This needs to be done - for example - to prevent the ACC environment connecting to the Prod environment.

For Cloud, this is done automatically when restoring Production to ACC, TEST or DEV, or ACC to TEST etc.

Multiple application servers - Cache refresh

If multiple application servers are installed, the cache is automatically kept in sync. So if you make configuration changes (FieldDefiner, report definitions, TSI etc.), the configuration will be automatically adopted by all application servers. By default, the servers check for changes once every 60 seconds.

It is possible to change this value if issues are encountered. In the application server service configuration file located in ...\Server\tanuki\appserver\conf, add the following parameter to the Java additional parameters:

```
wrapper.java.additional.nr=-Dcom.planonsoftware.cacheversioncheck.interval=60
```

The interval for the timer is in seconds.

To disable the timer, set it to a value <= 0. This is only useful for single server environments. If set to a value between 0 and 60, the interval will be set a minimum of 60 seconds. The interval can be set to a value higher than 60 seconds.

Database

Reconnecting to the database automatically

If the database connection is lost for a long time, the connection is restored by default. This is achieved by configuring the database connection to do an automatic retry by including the following parameters to the standalone-full.xml.

Parameter	Explanation
check-valid-connection-sql Oracle	This query is executed on the database to validate that the connection still works. select 1 from dual MSSQL: select 1
background-validation	Validate the database connections in the background (when they are idle), default is true.
background-validation-millis	Validates the connection every X milliseconds. (Default is 10000)
query-timeout	The maximum amount of seconds that a query can run. By default this is 50 minutes (3000 seconds).
set-tx-query-timeout	The query-timeout is applied to the whole transactions instead of on individual queries. The default is true.
idle-timeout-minutes	Indicates the maximum time in minutes that a connection may be idle before it is closed. The default is 15 minutes, setting this to 0 disables it.
blocking-timeout-millis	The amount of milliseconds to wait before a connection from the connection pool becomes available. If no new connection is found in X milliseconds (default is 5000) an exception is thrown.

Parameter	Explanation
	When using the CLI to update the attribute blocking-timeout-millis, use blocking-timeout-wait-millis

Read using CLI

To know what all module options are available under PlanonDS

```
/subsystem=datasources/data-source=PlanonDS/:read-resource(recursive=false)
```

To read a particular attribute

```
/subsystem=datasources/data-source=PlanonDS/:read-attribute(name=blocking-timeout-wait-millis)
```

Write / modify the value of a particular attribute

```
/subsystem=datasources/data-source=PlanonDS/:write-attribute(name=background-validation-millis,

value=20000)
```

Changing the Oracle driver

The Suite installer takes care of configuring the Oracle driver for the Oracle version.

In the Suite installer the Oracle version you want to use can be provided. If you want to move to another Oracle version you have to reinstall Planon to update the Oracle driver.

If you do not want to do a fresh install, you can update the driver manually. But keep in mind that this is a manual change.

> In case of a major update installation in a future update installation, the old version will be prefilled. You can just enter your new version and the installer will install the proper Oracle driver for you. This manual change is no longer needed.

If you want to change the Oracle version manually you need to change the driver as follows:

- standalone-full.xml

1. Start the jboss-cli.bat
2. Run the following command:

/subsystem=datasources/data-source=PlanonDS/:write-attribute(name=driver-name,value=***oraclex***)

- Oracle module.xml

  The location of this file is: ..\Server\wildfly-*\modules\system\layers\base\com \oracle\main

1. Update: <module name="com.***oraclex***" services="export" export="true"/>
2. Save and close the file.

   Where ***oraclex*** is one of the following values:

   - oracle7 for Oracle 12.1
   - oracle8 for Oracle 12.2 and Oracle 18
   - oracle10 for Oracle 19

LDAPS configuration

In order to use LDAPS authentication, you must configure the LDAPS settings in the standalone-full.xml to match the LDAPS configuration to be used by the installation customer.

> For security reasons Planon recommends to use LDAPS; e.g. LDAP via SSL.

You can choose the LDAPS solution that suits your requirement. The following section only serves as an example.

> The configuration below is just an example, you need to change the values of the parameters according to your LDAPS setup. For more information on LDAPS (parameters, procedures or user guides), see http://docs.jboss.org/jbportal/v2.7.1/referenceGuide/html/ldap.html.

If you intend to use the LDAPS login module, make sure that the default Planon login module is not in your standalone-full.xml file. Execute the following command in the JBoss CLI to remove the default Planon login module:

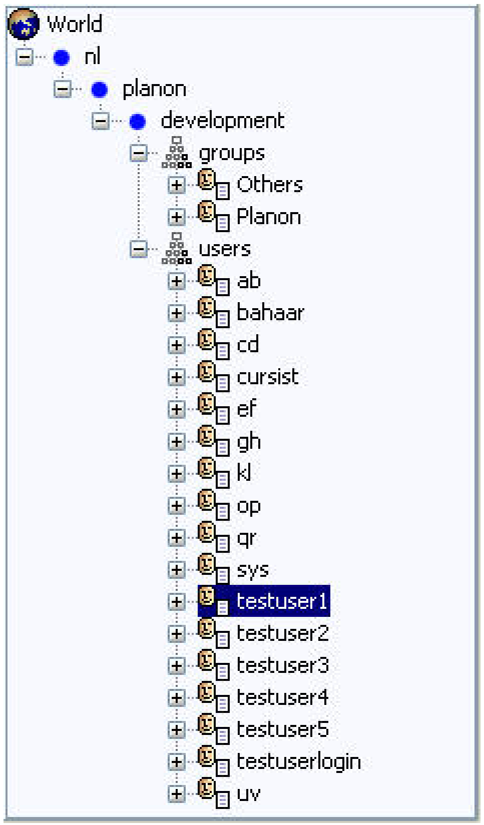
```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication =classic/login-
module=
 com.planonsoftware.jboss.login.artemis.server.jboss.JBossServerLoginModule/:remove
```

Execute the following command in the JBoss CLI to remove the CommitPlanonUserLoginModule. This module will be recreated in a later step, we delete it because the CLI only supports adding new login modules. The order of the login module must be updated, this one must always be executed last.

```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication =classic/login-
module=
 com.planonsoftware.jboss.login.artemis.server.CommitPlanonUserLoginModule/:remove
```

Ensure your LDAPS server is configured.

> The following screenshot an LDAPS server configuration example that is used as a model in the following configuration.



Login configuration with plain password

For LDAPS, use only the login module described below.

1. Open JBoss CLI. Update the parameters in the CLI command to match your LDAPS server's hierarchical organizational structure.

```
/subsystem = security/security-domain = PlanonSecurityDomain/authentication =
 classic/login-module = org.jboss.security.auth.spi.LdapExtLoginModule /:add(code
 = org.jboss.security.auth.spi.LdapExtLoginModule,flag = required,module-options
 = ["java.naming.provider.url" => "ldaps://host:port/", "throwValidateError"=>"true",
 "baseCtxDN"=>"dc=development, dc=planon, dc=nl", "bindDN"=>"uid=testuser1, ou=users,
 dc=development, dc=planon, dc=nl", "bindCredential"=> "mypassword", "baseFilter"=>"(uid=


{0})", "rolesCtxDN"=>"ou=users, dc=development, dc=planon, dc=nl", "roleFilter"=>"(uid={0}

)", "roleAttributeID"=>"memberOf", "roleAttributeIsDN"=>"true", "roleNameAttributeID"=>"cn",
 "searchScope"=>"ONELEVEL_SCOPE", "allowEmptyPasswords"=>"false"])
```

2. You must grant the Planon ‘role’ to the authenticated user in the server login configuration. Execute the following in JBoss CLI:

```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication=classic/login-
module =com.planonsoftware.jboss.login.artemis.server.CommitPlanonUserLoginModule/:add

(code=com.planonsoftware.jboss.

login.artemis.server.CommitPlanonUserLoginModule,flag=required,module-
options="roles"=>"Planon")
```

3. Add your certificate to the client cacerts keystore if needed (for example self-signed certificate). To do so use the following command:

The keytool can be found in the following location ...\Server\jdk-*\bin

```
keytool -cacerts -import -trustcacerts -noprompt -file path\to\your\certificate.crt -
alias yourAlias -storepass changeit
```

4. Restart your service for the changes to take effect.
5. When a user logs on to Planon Software Suite, the user credentials are first authenticated against LDAPS and subsequently it is verified whether the user name exists in Planon Software Suite. If both tests succeed, the user is logged on. If either test fails, the user is not granted access.

> The user name in LDAPS and in Planon must be the same.

Login configuration with encrypted password

By default the LDAPS password is plain text, but if needed it can be encrypted.

For security reason the JMX console is disabled in the application server. Because of this, it is not possible to use the standard configuration as provided by the security guide from JBoss.

Procedure:

1. Create a JCEKS keystore with a secret key. Run the keytool located in ...\Server\jdk-*\bin\

```
keytool -genseckey -alias vault -storetype jceks -keyalg AES -keysize 128 -storepass

vault22-keypass vault22 -dname "CN=vault, O=planon, C=NL" -keystore /path/to/
vault.keystore
```

2. Create a Vault directory

```
mkdir /path/to/vault-data-dir
```

3. Encrypt your LDAPS password. Run the following command with the vault tool located in ...\Server\wildfly-*\bin. Save the outcome.

```
vault.bat -a passa -b LdapLogin -e /path/to/vault-data-dir -i 22 -k /path/to/vault.keystore -
p vault22 -s 87654321 -v vault -x mypassword
```

4. Open JBoss CLI. Update the parameters in the CLI command to match your LDAPS server's hierarchical organizational structure.

```
/subsystem = security/security-domain = PlanonSecurityDomain/authentication =
 classic/login-module=org.jboss.security.auth.spi.LdapExtLoginModule /:add(code
 = org.jboss.security.auth.spi.LdapExtLoginModule,flag = required,module-options
 =["java.naming.provider.url"=>"ldaps://host:port/", "throwValidateError"=>"true",
 "baseCtxDN"=>"dc=development, dc=planon, dc=nl", "bindDN"=>"uid=testuser1, ou=users,
 dc=development, dc=planon, dc=nl", "bindCredential"=>"${VAULT::LdapLogin::passa::1}",
 "baseFilter"=>"(uid={0})", "rolesCtxDN"=>"ou=users, dc=development, dc=planon, dc=nl",
 "roleFilter"=>"(uid=0})", "roleAttributeID"=>"memberOf", "roleAttributeIsDN"=>"true",
 "roleNameAttributeID"=>"cn", "searchScope"=>"ONELEVEL_SCOPE",
 "allowEmptyPasswords"=>"false"])
```

5. You must grant the Planon ‘role’ to the authenticated user in the server login configuration. Execute the following in JBoss CLI:

```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication=classic/login-
module =com.planonsoftware.jboss.login.artemis.server.CommitPlanonUserLoginModule/:add

(code=com.planonsoftware.jboss.

login.artemis.server.CommitPlanonUserLoginModule,flag=required,module-
options="roles"=>"Planon")
```

6. Add your certificate to the client cacerts keystore if needed (for example self-signed certificate). To do so use the following command:

The keytool can be found in the following location ...\Server\jdk-*\bin

```
keytool -cacerts -import -trustcacerts -noprompt -file path\to\your\certificate.crt -
alias yourAlias -storepass changeit
```

7. Using the JBoss CLI, execute the following:

```
/core-service = vault:add(vault-options = ["KEYSTORE_URL"=>"/path/
to/vault.keystore","KEYSTORE_PASSWORD"=>"Outcome_of_step_4",
 "KEYSTORE_ALIAS"=>"vault", "SALT"=>"87654321","ITERATION_COUNT"=>"22",
 "ENC_FILE_DIR"=>"/path/to/vault-data-dir"])
```

8. Restart your service for the changes to take effect.
9. When a user logs on to the Planon application, the user credentials are first authenticated against LDAPS and subsequently it is verified whether the user name exists in Planon Software Suite. If both tests succeed, the user is logged on. If either test fails, the user is not granted access.

> The user name in LDAPS and in Planon Software Suite must be the same.

Microsoft Active Directory configuration

For Active Directory, a login module is developed that provides information to the user concerning the reasons for failing to log on.

The following cases are supported:

- Invalid credentials.
- Not permitted to log on at this time.
- Not permitted to log on at this workstation.
- Account disabled.
- Password must be reset.
- Account is locked.
- Unable to log on.
- User name was not found.
- Your password is expired.
- Your account has expired.

In all other cases a generic error is displayed (“Unable to log on.”).

> Messages are only displayed if the login module flag is required. If you set it to another level, the module is not mandatory and no messages are displayed.

In order to use Active Directory authentication, you must configure the LDAPS settings first in the standalone-full.xml to match the LDAPS configuration to be used by the installation customer.

All steps are equal to the LDAPS configuration (please refer to the previous sections). The only difference between standard LDAPS and AD LDAPS is the login module.

1. Replace all occurrences of the LDAPS login module:

```
org.jboss.security.auth.spi.LdapExtLoginModule
```

   with the AD login module:

```
com.planonsoftware.jboss.login.artemis.server.PnActiveDirectoryLoginModule
```

2. Do this twice for 'Login configuration with plain password' in the command line of Step 1.
3. Do this twice for 'Login configuration with encrypted password' in the command line of Step 4.

WebDAV

Web based Distributed Authoring and Versioning, or WebDAV, is a set of extensions to the Hypertext Transfer Protocol (HTTP) that allows computer users to edit and manage files collaboratively on remote Webservers.

WebDAV is used for storing files on the web, so that the files can be accessed from anywhere. It allows you to create, update and delete files directly on the server.

Planon Software Suite uses the WebDAV protocol to access files located on a web server. You can set up Apache Tomcat as a WebDAV server.

> Files with the following special characters in their file names cannot be viewed (are invisible) in the file selection pop-up and also an error is displayed when uploading the file to the WebDAV location: **# ;** and **%**. This also applies to the files used by CAD Import.

Creating WebDAV folder using Apache Tomcat

If you want to use WebDAV on the web server, use the Suite Installer to install a web server for WebDAV.

Secure WebDAV

By default, the installer installs password protected WebDAV:

- BASIC authentication
- The tomcat-users.xml located in … \Server\tomcat-*\conf is updated accordingly.

WebDAV folder outside the web server

By default, the installer installs the WebDAV file location in the folder selected during the installation:

- A file webdav.xml with this file location configuration is added in … \Server\tomcat-*\conf\Catalina\localhost

Connecting Planon Software Suite to the WebDAV folder

Procedure

1. Set up the file locations to the WebDAV folder.
2. Arrange the login name and password for the WebDAV folder.
3. You can register a joint WebDAV login name and password for all users who are working within the same Planon property set in Launch Center > System Settings > WebDAV settings.
   These settings allow access to the file locations on the WebDAV server.
4. Set up the WebDAV folder in Launch Center > System Settings TSI > File locations.

ℹ	For more information on system settings, please refer to System Settings documentation.
ℹ	For security reasons it is strongly recommended to set the **Allow user-defined path** to **No**. This allows users to only store documents in the default file location folder or subfolders of that file location.

Single Sign-On Planon Web Client

This chapter describes how to configure Planon Web Client and the web server to support Single Sign-On (SSO) for on-premise customers on a Kerberos-enabled domain using:

- WAFFLE - only for Windows platform
- Apache SPNEGO implementation
- Tomcat Keycloak adapter - to be used to configure SSO via a Keycloak Server.

ℹ
- Single sign-on is enabled using a Kerberos domain, but it can also be another authentication mechanism. WAFFLE or the Apache SPNEGO implementation takes care of the selection of the accepted mechanism. WAFFLE or the Apache SPNEGO implementation may come with a login prompt because it is not Kerberos but another mechanism.
- This chapter is NOT an installation guide or manual for Kerberos security, WAFFLE authentication, SPNEGO authentication or Keycloak.

WAFFLE

WAFFLE (**W**indows **A**uthentication **F**unctional **F**ramework **L**ight **E**dition) is a native Windows authentication framework consisting of two C# and Java libraries that perform functions related to Windows authentication, supporting Negotiate, NTLM and Kerberos.

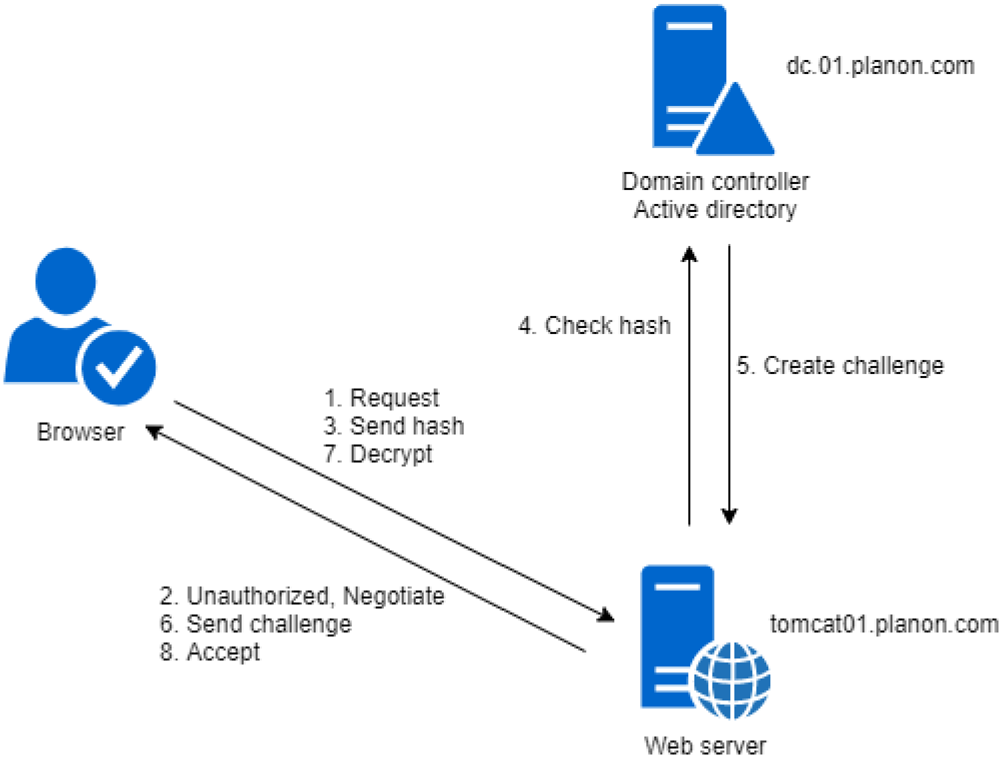
This solution only works for the Windows platform. The application server, web server and client must all be in the same domain.

ℹ
- For more information, for example about configuration options and enabling extra logging for Waffle, see: https://github.com/dblock/waffle/blob/master/Docs/tomcat/TomcatSingleSignOnValve.md
- For troubleshooting Waffle, see: https://github.com/Waffle/waffle/blob/master/Docs/Troubleshooting.md

How WAFFLE SSO authentication works

This image depicts the working of WAFFLE SSO authentication.

ℹ A precondition for WAFFLE SSO is that the user request comes from a device that is logged on to the domain.



1. The browser sends a request to the web server.
2. The web server replies with *unauthorized* and proposes negotiations.
3. The client browser gets the user's credentials that were used to log into Windows, takes its hash and sends it to the server.
4. When receiving the hash, the server looks up the user store and identifies the user.

> There is no keytab file needed as is the case for SPNEGO.

5. An unique and encrypted challenge is created.
6. The server sends the challenge to the browser. That challenge can be only decrypted using the user's password.
7. The browser decrypts the challenge with the user's credentials and sends the response back to the server.
8. The server checks whether the response for the challenge is correct and serves the user request if the answer is correct. If the answer is wrong, the server denies the access to the requested resources and sends the unauthorized message.

Configuring the web server

Adapt the following files located in: ..\Server\tomcat-*\conf\Catalina\localhost

1. Open Root.xml and remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator). Add the following realm and valve below the AccessKeyValve:

```
<Valve className="waffle.apache.NegotiateAuthenticator" />

<Realm className="org.apache.catalina.realm.CombinedRealm">

<Realm className="waffle.apache.WindowsRealm" />

<Realm appName="PlanonRealmLogin" className="nl.planon.tomcat.PnMessageJaasRealm"

  userClassNames="nl.planon.cerebeus.PnUser"

  roleClassNames="nl.planon.cerebeus.PnRole"

  allRolesMode="authOnly"/>

</Realm>
```

2. Open Webclient.xml and remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator). Add the following realm and valve above the trustedServiceKeystore:

```
<Valve className="waffle.apache.NegotiateAuthenticator" />

<Realm className="waffle.apache.WindowsRealm" />
```

3. Open **sc.xml** and remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator) or BasicAuthenticator. Add the following realm and valve above the trustedServiceKeystore:

```
<Valve className="waffle.apache.NegotiateAuthenticator" />

<Realm className="waffle.apache.WindowsRealm" />
```

The accounts used to log in to Planon should have the following user name:

- NetBios name

  For example: “planon\username” or “planon.com\username”

Use domain user for the web server service

If you want to use the service account for the server service in combination with WAFFLE the HTTP SPN for the web server should be set to the service account. Run the following command to set the SPN:

```
setspn -U -S http/planonserver.domain.ext serviceaccount@domain.ext
```

Note that you need to be a member of the Domain Admins, Enterprise Admins or have been granted the permission to set the SPN. By default the http SPN is not set, so Waffle will use the host SPN (default linked to Local System). After setting the SPN, you cannot use Waffle with Local System anymore until you remove the SPN again.

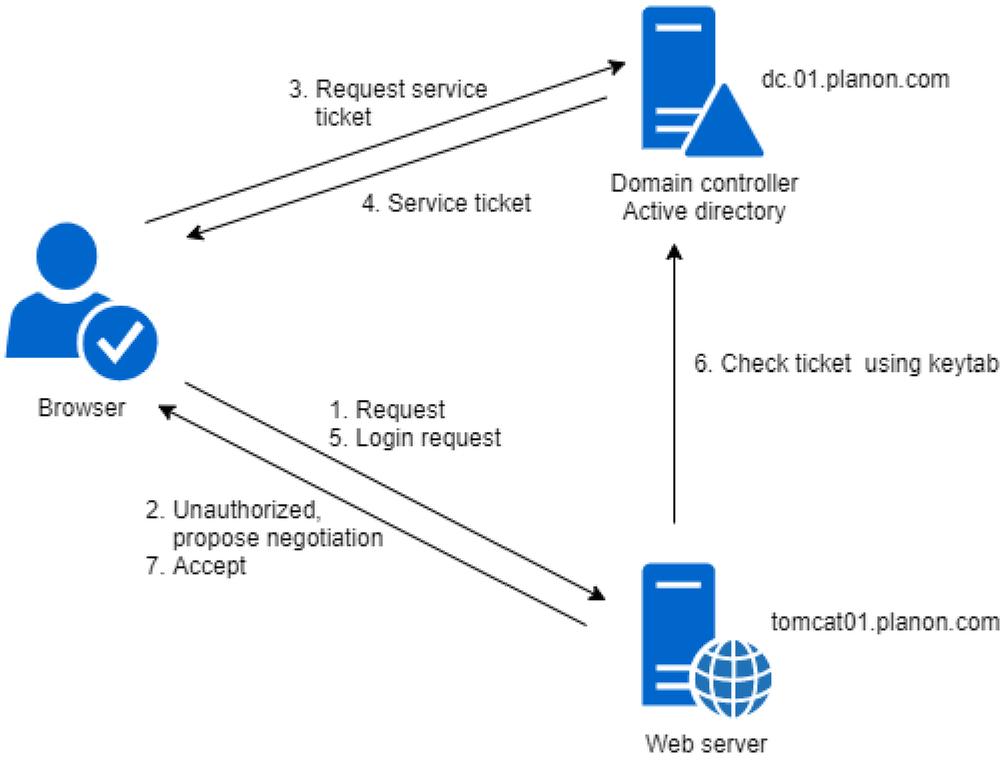
Apache SPNEGO implementation

For more information, refer to https://tomcat.apache.org/tomcat-9.0-doc/windows-auth-howto.html.

SPNEGO SSO using Kerberos does not work on localhost. You cannot try out your SSO configuration on the web server. Negotiation will take the unsupported NTLM authentication instead of localhost.

How SPNEGO SSO authentication works

The following image depicts the working of SPNEGO SSO authentication.



1. The browser sends a request to the web server.
2. The web server replies with *unauthorized* and proposes negotiations.
3. The browser decides to go with Kerberos (because configured). The browser takes the client ticket from the local ticket cache, and uses that ticket to request a service ticket for HTTP/tomcat01@PLANON.COM from the domain controller.
4. The domain controller validates the client ticket and returns the service ticket.
5. The browser sends a login request to the web server.
6. The web server verifies the ticket of the client against the keytab.
7. If the ticket is validated, the server accepts the login request.

Generating a key tab

Because the web server is responsible for authenticating the caller (the user), it needs to authenticate itself against the security domain. When such a service needs to authenticate itself, this is typically done by using a keytab.

A keytab is a file with trusted (private credentials) information associated with a domain user who is mapped to the service for which the keytab is valid. The user's name is then mapped to a SPN (Service Principal Name).

For the web server to be able to authenticate a Planon user, a keytab needs to be generated.

> Generating a keytab can only be done by a user with administrator’s privileges on a domain controller.

A keytab creation is done by executing the ktpass executable.

Given a domain realm PLANON.COM, the following steps have to be performed:

1. Create a Kerberos user for the web server. If you are using PPJC SSO, make sure you create different users for the application server and the web server.

> In the example below tomcat01 is used

2. Set the option the user does not need to change the password.
3. Set the option the password never expires.
4. Map the service principal name to the user account.

```
setspn -A HTTP/HostName.planon.com tomcat01
```

The HostName should be the FQDN of the web server, used in the browser to connect to the web server.
Do not use the CNAME record it must be the host name.

5. On the domain controller, open a command line and create a keytab:

```
ktpass /out tomcat.keytab /mapuser tomcat01@PLANON.COM /princ
 HTTP/HostName.planon.com@PLANON.COM /pass tomcat01pass /kvno 0 -ptype
 KRB5_NT_PRINCIPAL -crypto all
```

- Type the command in one line or create a .bat and run it.
- The password (/pass tomcat01pass) will override the password of the user created in Step 1.
- Make sure the service principal name is unique; otherwise Single Sign On will not work.
- We recommend to save this command, so you can refer to/reuse it later.
- About encryption limits, check the Server Hardening chapter, section .

6. Copy the file to the web server host (for example to …\Server\tomcat-*\).

> Because the file contains sensitive security information and is only needed by the web server, it is advisable to restrict access to the file to the user running the web server.

Setting the Kerberos environment

For the web server, the Kerberos environment needs to be set. This is done by creating the file containing the Kerberos environments settings.

The krb5.conf below is an example more parameters can be added to make it better fit a specific AD configuration or fulfill security requirements. See: http://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html

Settings file example:

```
[libdefaults]
  default_realm = PLANON.COM
  default_keytab_name = FILE:AbsolutePath\Server\tomcat-*\tomcat.keytab
 default_tkt_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
 default_tgs_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
 permitted_enctypes = aes256-cts-hmac-sha1-96 rc4-hmac des-cbc-crc des-cbc-md5
  forwardable=true
[realms]
  PLANON.COM = {
  kdc = planon.com
  default_domain = planon.com
  }
[domain_realm]
  .planon.com = PLANON.COM
```

For the parameter kdc, the hostname is not required. With only the domain name defined, it will resolve to the correct domain controller itself even if there are multiple domain controllers.

1. Update this example with your configuration.
2. Save this file as krb5.ini in the following: …\Server\tomcat-*\conf.

Amending the Tanuki configuration file

To enable Single Sing On, you must add a parameter to the Tanuki configuration file.

1. Stop the Tanuki service.
2. Go to ...\Server\tanuki\webserver\conf and open the tomcat-wrapper-default.conf file.
3. Locate the # Java Additional Parameters section and add the following parameter (and increment the sequence number):

```
wrapper.java.additional.nr=-Djava.security.krb5.conf=AbsolutePath\\Server\\tomcat-*\\conf\
\krb5.ini
```

Configuring web server

To access Planon Web Client, you need to have the Planon role.

Open the web server’s login configuration file in a text editor. This file is named jaas.config and is located: … \Server\tomcat-*\conf

1. In the jaas.config, add the following jaas configurations at the end of the file:

```
com.sun.security.jgss.krb5.accept {

    com.sun.security.auth.module.Krb5LoginModule required

    doNotPrompt=true principal="HTTP/HostName.planon.com@PLANON.COM"

    useKeyTab=true

   keyTab="AbsolutePath/Server/tomcat-*/tomcat.keytab"

     isInitiator=false

    storeKey=true;

 };
```

2. Update the “com.sun.security.jgss.krb5.accept” configuration with your configuration.
3. Update the server.xml located in …\Server\tomcat-*\conf. Remove the single sign on valve:

```
<Valve className="org.apache.catalina.authenticator.SingleSignOn"
 requireReauthentication="true"/>
```

Planon Web Client configuration

1. Open webclient.xml that is located in: …\Server\tomcat-*\conf\Catalina \localhost

   This file is generated the first time the web server is started. If you do an update, this file will not be overwritten.

2. Remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator). Add the following realm and valve above the trustedServiceKeystore:

```
<Realm className="nl.planon.tomcat.SPNegoRealm"

          stripRealmForGss="false"

          allRolesMode="authOnly"/>

          <Valve className="org.apache.catalina.authenticator.SpnegoAuthenticator"/>
```

If you do not want to use user names including the domain, you must set stripRealmForGss="true".

3. Open the ROOT.xml that is located: …\Server\tomcat-*\conf\Catalina \localhost

   If you do an update, this file will not be overwritten.

4. Remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator). Add the following realm and valve below the AccessKeyValve:

```
<Realm className="org.apache.catalina.realm.CombinedRealm" allRolesMode="authOnly">

 <Realm className="nl.planon.tomcat.SPNegoRealm"

          stripRealmForGss="false"

          allRolesMode="authOnly"/>

 <Realm className="org.apache.catalina.realm.JAASRealm"

          appName="PlanonRealmLogin"

          userClassNames="nl.planon.cerebeus.PnUser"

          roleClassNames="nl.planon.cerebeus.PnRole"
```

```
        allRolesMode="authOnly"/>

</Realm>

<Valve className="org.apache.catalina.authenticator.SpnegoAuthenticator"/>
```

5. Open sc.xml that is located in: …\Server\tomcat-*\conf\Catalina \localhost

   This file is generated the first time the web server is started. If you do an update, this file will not be overwritten.

6. Remove the Realm (PlanonRealmLogin) and FormAuthenticator valve (PnMessageFormAuthenticator) or BasicAuthenticator. Add the following realm and valve above the trustedServiceKeystore:

```
<Realm className="nl.planon.tomcat.SPNegoRealm"

        stripRealmForGss="false"

        allRolesMode="authOnly"/>

        <Valve className="org.apache.catalina.authenticator.SpnegoAuthenticator"/>
```

> If you do not want to use user names including the domain, you must set stripRealmForGss="true".

7. Restart the Tanuki service.

Verify the configuration

The following checks can be executed to verify the configuration:

1. Run the setspn tool to see that the requested SPN is not duplicate. If it is duplicate, Single Sign On will not work:

```
setspn -T * -X
```

2. List the keytab and see that the content is as expected. Use java's klist:

```
<install>\Planon*\Server\jdk-*\bin\klist -t -K -e -k <your keytab
 filename including path>
```

Example outcome with keytab=tomcat.keytab:

```
Key tab: tomcat.keytab, 1 entry found.

[1] Service principal: HTTP/HostName.planon.com@PLANON.COM

      KVNO: 0

      Key type: 23

      Key: 0x4e150649cdf4b2b394ccefbcd08d709a Time stamp: Jan 01, 1970 01:00
```

3. Try to login with the keytab using java's kinit:

```
PathTo\Server\jdk-*\bin\kinit -t PathTo\Tomcat.keytab HTTP/
HostName.planon.com@PLANON.COM
```

If successful the return message for this call will be:
“New ticket is stored in cache file:…”

Enabling logging

By default you do not see all the logging that is related to single sign-on.

The following steps describe exactly what should be done to enable it.

1. Stop the Tanuki service.
2. Go to and open ..\Server\tomcat-*\conf\logging.properties
3. Search for and uncomment:

```
org.apache.catalina.authenticator.level = FINE
```

4. Set the following properties to FINE:
   - java.util.logging.ConsoleHandler.level
   - 1catalina.org.apache.juli.FileHandler.level
5. Save and close the file.
6. Restart your service. Logging will subsequently be available in the logs directory: ..\Server\tomcat-*\logs

Configuring browsers

A browser only allows Kerberos authentication if it is a trusted site. Browsers do not always support SSO on the same server as the webserver. They will fall back to normal login with login dialog. If you encounter this try login from a different machine

Configuring Internet Explorer

Open the internet options and check the following configuration:

1. Trusted sites list

a. On the Security tab.
b. Click on the Trusted Sites icon.
c. Click the Sites button.
d. Add the site to the list.

2. Security level

a. On the Security tab.
b. Click the Custom Level button.
c. Search in the list for User Authentication > Logon and enable one of the following options: Automatic logon with current username and password or Automatic logon in intranet zone.
d. For Miscellaneous enable: Web sites in less privileged Web content zones can navigate into this zone.

3. Integrated windows authentication

a. On the Advanced tab.
b. For Security enable: Integrated Windows Authentication is set.

> ⚠ All these settings expect a restart of the Internet Explorer.

Configuring Chrome

Open the internet options and check the following configuration:

1. Trusted sites list

a. On the Security tab.
b. Click on the Local intranet icon.
c. Click the Sites button.
d. Click the Advanced button.
e. Add the site to the list.

Configuring Firefox

The site must be explicitly added to the configuration:

1. In the URL bar go to about:config
2. Filter on network.negotiate-auth.trusted-uris
3. Add the URL to this list.

Troubleshooting SPNEGO and WAFFLE

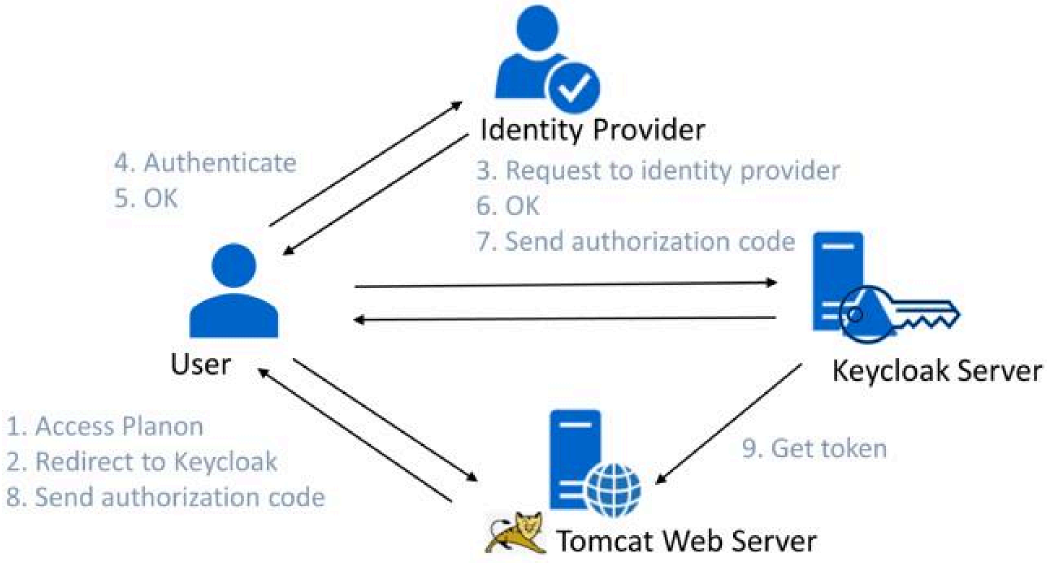
Question	Answer
“Error 400 – Bad Request” on the client. If you enable FINE logging on Catalina you will get the following errors: Error parsing HTTP request header Request header is too large	When a user is a member of a large number of active directory groups the Kerberos authentication token for the user increases in size. The HTTP request that the user sends contains the Kerberos token in the header, and the header size increases as the number of groups goes up. Enlarge the maxHttpHeaderSize in the connector you use in the tomcat-*/ conf > server.xml file.

Tomcat Keycloack adapter

Keycloak is a tool for identity and access management and can be used in front of the Planon application to handle authentication instead of using a standard Planon login.

Authorization is still applied via the Planon application.

The following image depicts the communication with Keycloak.



The Tomcat keycloak adapter and Planon Tomcat Keycloak adapter are installed in the following location: …\Server\tomcat-*\lib

- We only deliver the Keycloak adapter to make it possible to connect Planon to the Keycloak Server to configure other authentication options as SPNEGO, Waffle or Planon login.
- How to configure the Keycloak Server is not part of this.
- For details about how to install and configure Keycloak Server see:

Keycloak json

Generate the Keycloak.json file with the Keycloak server and place it in the following location:

```
…\Server\tomcat-*\conf\keycloak.json
```

Extend the content of the Keycloak.json file with the following parameter and value:

```
"principal-attribute" : "preferred_username”
```

Configuring the web server

To make the Planon ProCenter Web application available for SSO, adapt the following files located in:

```
…\Server\tomcat-*\conf\Catalina\localhost
```

1. Open **Webclient.xml** and remove:

   a. the PnMessageFormAuthenticator valve.

2. Add the following valve above the trustedServiceKeystore parameter:

```
<Valve className="nl.planon.tomcat.keycloak.KeycloakAuthenticatorValve"/> <Parameter
 name="keycloak.config.file" value="AbsolutePath/Server/tomcat-*/keycloak.json" />
```

3. Open **Root.xml** and remove:

   a. the ForgotPasswordLoginValve valve.

   b. the PnMessageFormAuthenticator valve.

4. Add the following valve above the ExcludingRoleValve valve:

```
<Valve className="nl.planon.tomcat.keycloak.KeycloakAuthenticatorValve"/> <Parameter
 name="keycloak.config.file" value="AbsolutePath/Server/tomcat-*/keycloak.json" />
```

5. Open **sc.xml** and remove:

   a. the PnMessageFormAuthenticator valve or the BasicAuthenticator.

6. Add the following valve above the ExcludingRoleValve valve:

```
<Valve className="nl.planon.tomcat.keycloak.KeycloakAuthenticatorValve"/> <Parameter
 name="keycloak.config.file" value="AbsolutePath/Server/tomcat-*/conf/keycloak.json" />
```

Planon login module security

When a user tries to log on and enters an invalid user name or password, an error message will appear. If these details are entered incorrectly three times in a row, Planon Software Suite will automatically quit and the user account will be locked for five minutes.

Locking out the user account is configurable, as explained in the following sections.

Tightening security

The default Planon JAAS-login module (JBossServerLoginModule) comes with a number of options that allow application administrators to tighten the security of Planon Software Suite, and to automatically perform additional audits on the logins of users. By default, these audits are enabled and not only log incorrect login attempts, but also lock out users that perform too many incorrect logins.

Settings overview

The Planon application administrator may configure the audit and locking settings used by the Planon login module.

You can configure the basic settings in System Settings > Password settings:

- Lockout time
- Maximum allowed logon attemps

Next to these basic settings the following options can be set in the login module:

- failedLoginDelay

  Denotes the number of seconds the application waits after an incorrect login attempt before handling a new login attempt (defaults to 1 second).

- resetFailedloginCounter

  Denotes the number of seconds before the incorrect login attempts are reset to 0. If this parameter is not in the login-config.xml it will fall back to its default value which is 30 min/1800 sec. If you are already locked it will not reset, the failedloginUnlockDelay will take care of that.

These settings only apply to the server login module supplied by Planon. For other authentication methods, such as LDAP/ActiveDirectory or Kerberos, these settings are not available. Most of these login modules come with their own options providing similar functionality.

For more information about that, please refer to documentation of the authentication module in use.

Configuring the login module

1. Start the JBoss CLI.
2. Run the following command to set the failedLoginDelay:

```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication=classic/login-
module

=com.planonsoftware.jboss.login.artemis.server.CommitPlanonUserLoginModule:write-
attribute(name

=module-options.failedLoginDelay,value=10)
```

3. Run the following command to set the failedLoginCounter:

```
/subsystem=security/security-domain=PlanonSecurityDomain/authentication=classic/login-
module

=com.planonsoftware.jboss.login.artemis.server.CommitPlanonUserLoginModule:write-
attribute(name

=module-options.resetFailedloginCounter,value=900)
```

4. Restart the application server.

If not explicitly configured, the default settings will be used. It is not possible to completely disable this functionality for the Planon login module.

Frequently asked questions

Question	Answer
Can I disable this functionality?	No, currently you cannot disable it. However, you can make the failed login threshold very large, for example, 999, and set the delay to zero.

Question	Answer
	Note that this could pose a security threat, so be careful with this!
Can I manually unlock an account?	No, currently you cannot manually unlock an account.
Do the other login modules have these features?	Perhaps; please refer to the documentation of that particular login module for more information.
The lockout takes much longer than what is specified!	If users keep trying to log on before the unlock delay is passed, the unlock delay is reset to 0!

Secure data access

Planon Software Suite is a highly configurable system; at various levels you can arrange access to data (**FieldDefiner**, **TSI Manager**, **Authorization**, **Launch Center Management**). However, using TSI filters only may give a false sense of data security. If the data in question includes critical or confidential information that should never be accessible to end users, then a TSI should be used in combination with authorization. Using TSIs without authorization could enable users to access data fields they are not authorized to access using dialog boxes or reports. For an entirely secure solution, we strongly recommend using a TSI in combination with authorization.

For details see the Authorization documentation.

Handling certificates and keystores

The Java Runtime Environment (JRE) stores the certificates that are trusted into a so-called keystore. By default, the keystore contains certificates that are signed by a trusted Certificate Authority (CA). As a result, the Planon Software Suite is allowed to make a connection to a secured location with such a certificate.

It is possible certificates need to be added to the keystore. The Suite installer tool provides the possibility to do this, for details see the Planon ProCenter - Suite Installer manual.

Java also comes with a tool to handle keystores and certificates. For example:

- Importing certificates
- Check certificates in the keystore
- Create your own keystore and certificate
- etc.

For more information on how to use this tool, see .

The keytool can be found in the following location:

- Server JDK: …\Server\jdk-*\bin

Trusted Services

Some services are required to make use of the Planon Software Suite business interface, executing business logic on behalf of a particular user. In order for the code to function as intended (i.e. with the proper authorization) such services must be able to log on acting as that particular user.

Since these services will be unaware of the user’s password, obviously for the sake of security, another means of authentication is required.

The **Trusted Services** solution allows the registration of services that are considered ‘trusted’. This registration involves adding a certificate for each service configured. Such a certificate is related to a particular service and corresponds with a key pair that is only available to that service.

Each service can now authenticate itself using an item of signed information. Planon Software Suite then enables each service authenticated this way to act as a known Planon user.

The authentication of a service takes place with each request from that service.

The installer automatically generates certificates and keystores for trusted services used by the Planon software such as mobile apps or the Planon application. The complete configuration is also automatically done at the time of installation. The trusted service is placed into the database and the certificate is automatically loaded on starting the application server. The certificate is reloaded each time the application server is restarted. To make sure always the latest installed certificate is used. This applies only for the automatically generated certificates and only if located in: ...\Server\wildfly-*\standalone. This is the location the installer will place these certificates automatically.

> ❗ If you run multiple application servers, make sure you use the same certificate for a specific service on all the application servers on which the service runs. Keep in mind the installer automatically generates certificates and is not aware about multiple application servers.

In the case of Trusted Services, we will create a self-signed certificate. If you want to use an already existing keystore entry, or you want to include the certificate in a chain, you will have to perform further actions, which due to the diversity of possible configurations, are not described in this manual.

> ℹ For information about keystores and certificates, see Handling certificates and keystores.

Configuring a trusted service

In **System Settings**, the **Trusted Services** TSI is used to manage trusted services. You can add a new service on the first selection level. The **Service ID** field should contain the alias name you used to create the key pair.

1. On the Key field, select the certificate that belongs to your trusted service.
2. Enter a name or description of the trusted service (optional).
3. Save the trusted service entry.

Before the service can use the application, its status must be set to Activated.

Logging

Application server logs

Planon makes use of a number of log files to log different types of information. On the application server, the following log files exist.

In the following location ...\Server\tanuki\appserver\logs\:

- wrapper-default.log: contains all the general logging and the service events of the Planon application. This log file can be used to generically diagnose the Planon application server. It contains a broad variety of information.

In the following location ...\Server\wildfly-*\standalone\log\:

- server.log: contains general runtime information for the Planon application server. Runtime events on the Planon application server are logged here. This log file rotates every day.
- boot.log: contains the start-up information for the Planon application server. This log file contains information about the start-up process itself, but also general information about the server, hardware and configuration. This can be useful when start-up issues arise, or when the exact configuration of the server is not known.
- perfmon.log: if performance logging is enabled, the application will log performance metrics at regular intervals into this file. Performance logging can be enabled in **System settings > Performance monitoring**. Set the **Is activated** field to **Yes**.
- securityautdit.log: if audit logging is enabled, the application will log audit events into this file.

Where to configure the application server logging

Planon uses the standard logging which comes with JBoss.

The logging is defined in the standalone-full.xml and can be configured with the JBoss CLI.

Below is an example to get more detailed logging for investigation purposes. If enabled it will generate a lot of logging and you should only do this for testing purposes:

1. Open the JBoss CLI.
2. Set the logger category from which the information is needed to DEBUG. For example logger category pnlog.DEFAULT:

```
/subsystem=logging/logger=pnlog.DEFAULT/:write-attribute(name=level,value=DEBUG)
```

3. Always set the console handler to the same log level that you want to log. Otherwise, the changes to the log level will not be reflected in the log file. In our example that would be DEBUG:

```
/subsystem=logging/console-handler=CONSOLE/:write-attribute(name=level,value=DEBUG)
```

4. The log will be available in your server.log located ...\Server\wildfly-*\standalone\log

For other options, see Audit log for the application server.

> If you do not know which logger should be set, use the JBoss CLI GUI. This tool allows you to easily view the different logger categories and change the log level from info to debug and back. For example if you are interested in Scheduler issues, set pnlog.SCHEDULERENGINE to DEBUG.

Web server logs

Planon makes use of a number of log files to log different types of information. On the web server, the following log files exist.

In the following location ...\Server\tanuki\webserver\logs:

- wrapper-default.log: contains all the general logging and the service events of the Planon application web server. This log file can be used to generically diagnose the Planon web server. It contains a broad variety of information.

In the following location ..\Server\tomcat-*\logs:

- catalina.*.log: contains messages from the Tomcat server itself.
- localhost.*.log: contains messages from the Planon application running on the web server.
- pss2.log, kiosk.log, etc.: contains messages from the Planon module running on the web server.
- localhost_access_log.*.log: contains a log line for every web request that is completed at the web server.

Where to configure the web server logging

Planon uses the standard logging which comes with Tomcat.

The logging in defined in the ..\Server\tomcat-*\conf\logging.properties.

Below is an example to get more detailed logging for investigation purposes. If enabled it will generate a lot of logging and you should only do this for testing purposes:

> Setting the level to FINE can expose sensitive information.

1. Set the following property to FINE:

   java.util.logging.ConsoleHandler.level

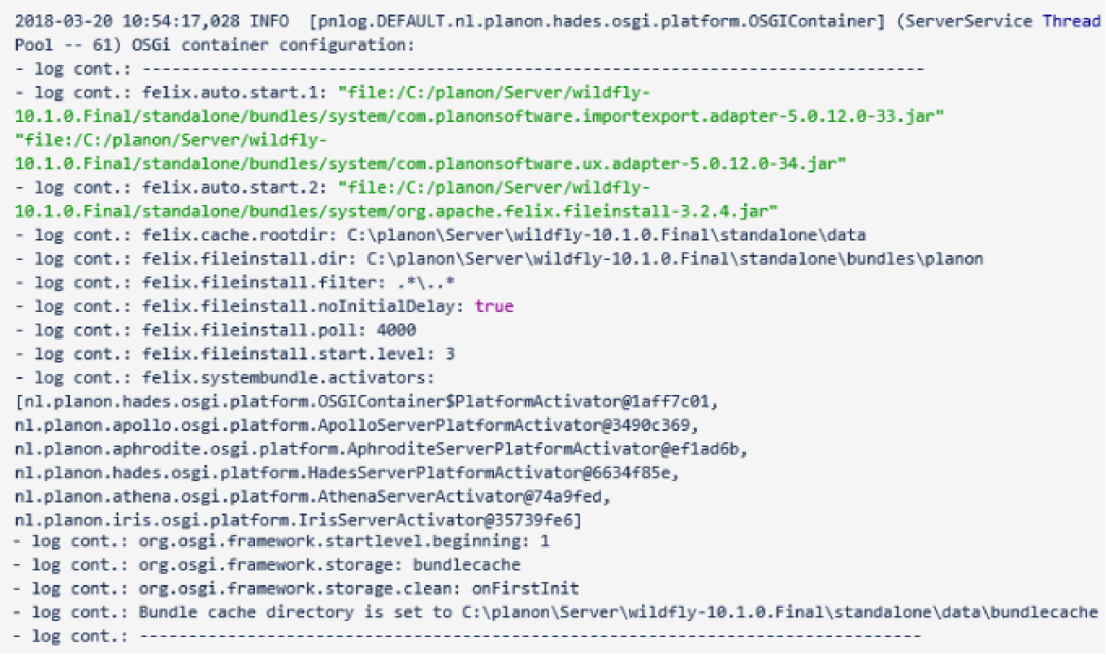
2. Save and close the file.
3. Restart your service. Logging will subsequently be available in the logs directory: ..\Server\tomcat-*\logs

Log lines with carriage returns and line feeds

If one log line contains carriage returns and line feeds, the corresponding lines are prefixed by **- log cont.:** to indicate that you are still looking at the same log line and *not* at a new action from the application.

> This applies to all logging originating from Planon. Logging provided by third-party components cannot always be prefixed by **- log cont.:**, because Planon has no control of this. For example, Tanuki service wrapper or WildFly are started before Planon becomes active.

For example, see the following information at start-up of the application server:



```
2018-03-20 10:54:17,028 INFO  [pnlog.DEFAULT.nl.planon.hades.osgi.platform.OSGIContainer] (ServerService Thread
Pool -- 61) OSGi container configuration:
- log cont.: --------------------------------------------------------------------------------
- log cont.: felix.auto.start.1: "file:/C:/planon/Server/wildfly-
10.1.0.Final/standalone/bundles/system/com.planonsoftware.importexport.adapter-5.0.12.0-33.jar"
"file:/C:/planon/Server/wildfly-
10.1.0.Final/standalone/bundles/system/com.planonsoftware.ux.adapter-5.0.12.0-34.jar"
- log cont.: felix.auto.start.2: "file:/C:/planon/Server/wildfly-
10.1.0.Final/standalone/bundles/system/org.apache.felix.fileinstall-3.2.4.jar"
- log cont.: felix.cache.rootdir: C:\planon\Server\wildfly-10.1.0.Final\standalone\data
- log cont.: felix.fileinstall.dir: C:\planon\Server\wildfly-10.1.0.Final\standalone\bundles\planon
- log cont.: felix.fileinstall.filter: .*\..*
- log cont.: felix.fileinstall.noInitialDelay: true
- log cont.: felix.fileinstall.poll: 4000
- log cont.: felix.fileinstall.start.level: 3
- log cont.: felix.systembundle.activators:
[nl.planon.hades.osgi.platform.OSGIContainer$PlatformActivator@1aff7c01,
nl.planon.apollo.osgi.platform.ApolloServerPlatformActivator@3490c369,
nl.planon.aphrodite.osgi.platform.AphroditeServerPlatformActivator@ef1ad6b,
nl.planon.hades.osgi.platform.HadesServerPlatformActivator@6634f85e,
nl.planon.athena.osgi.platform.AthenaServerActivator@74a9fed,
nl.planon.iris.osgi.platform.IrisServerActivator@35739fe6]
- log cont.: org.osgi.framework.startlevel.beginning: 1
- log cont.: org.osgi.framework.storage: bundlecache
- log cont.: org.osgi.framework.storage.clean: onFirstInit
- log cont.: Bundle cache directory is set to C:\planon\Server\wildfly-10.1.0.Final\standalone\data\bundlecache
- log cont.: --------------------------------------------------------------------------------
```

Audit log for the application server

Audit logging is done to monitor the following events:

- Login/logoff
- Failed logins
- Password changes
- Password resets

The logs are generated in: ...\Server\wildfly-*\standalone\log.

Logging for ‘logon/logoff’

Each time a user logs on or off, a log is created which is stored in the logon_logoff.log file in the application server’s log folder.

The following sections describe the configurations that are required for saving the logs in CSV and normal text format.

Configuration:

- Open the JBoss CLI
- Run the following commands:

For text format

```
/subsystem=logging/periodic-rotating-file-handler=LOGINLOGOFFS:add(append

        =true,file={"path"=>"loginlogoff.log", "relative-to"=>"jboss.server.log.dir"},suffix

        =".yyyy-MM-dd",autoflush=true)

/subsystem=logging/periodic-rotating-file-handler=LOGINLOGOFFS:write-attribute(name

        =named-formatter,value=PATTERN)

/subsystem=logging/logger

 =pnlog.AUTHENTICATION.nl.planon.hades.beans.definitionmanager.DefinitionManagerBean:add(level=DEBUG)

/subsystem=logging/logger
```

```
 =pnlog.AUTHENTICATION.nl.planon.hades.beans.definitionmanager.DefinitionManagerBean:add-
handler(name=LOGINLOGOFFS)
```

For CSV format

```
/subsystem=logging/size-rotating-file-handler=LOGINLOGOFFS:add(append

        =true,file={"path"=>"loginlogoff.csv", "relative-to"=>"jboss.server.log.dir"},max-backup-
index

        =1,rotate-size=10000k,autoflush=true)
```

```
/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.beans.definitionmanager.DefinitionManagerBean:

        add(use-parent-handlers=false,level=DEBUG)
```

```
/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.beans.definitionmanager.DefinitionManagerBean:

        add-handler(name=LOGINLOGOFFS)
```

Logging for failed logins

Each time a user logon fails, a log is created which is stored in the failedlogins.log file in the log folder of the application server.

A log is registered due to the following reasons:

- Rejected log-on: “Rejected log-on for account <UserName> due to incorrect password”
- Rejected log-on: “Rejected log-on for account <UserName> due to expired password”
- Rejected log-on: “Rejected log-on for account <UserName> due to inconsistent account data in the database”

Configuration:

- Open the JBoss CLI.
- Run the following commands:

For text format

```
/subsystem=logging/periodic-rotating-file-handler=FAILEDLOGINS:add(append

        =true,file={"path"=>"failedlogins.log", "relative-to"=>"jboss.server.log.dir"},suffix
```

```
              =".yyyy-MM-dd",autoflush=true)

/subsystem=logging/periodic-rotating-file-handler=FAILEDLOGINS:write-attribute(name
              =named-formatter,value=PATTERN)

/subsystem=logging/logger

 =pnlog.AUTHENTICATION.com.planonsoftware.jboss.login.artemis.server.Authenticator:add(level=DEBUG)

/subsystem=logging/logger

 =pnlog.AUTHENTICATION.com.planonsoftware.jboss.login.artemis.server.Authenticator:
              add-handler(name=FAILEDLOGINS)
```

For CSV format

```
/subsystem=logging/size-rotating-file-handler=FAILEDLOGINS:add(append
              =true,file={"path"=>"failedlogins.csv", "relative-to"=>"jboss.server.log.dir"},
              max-backup-index=1,rotate-size=10000k,autoflush=true)

/subsystem=logging/
logger=pnlog.AUTHENTICATION.com.planonsoftware.jboss.login.artemis.server.Authenticator:
              add(use-parent-handlers=false,level=DEBUG)

/subsystem=logging/
logger=pnlog.AUTHENTICATION.com.planonsoftware.jboss.login.artemis.server.Authenticator:
              add-handler(name=FAILEDLOGINS)
```

Logging for password changes

Each time the password of a user is changed via the "change password" action, a log is created which is stored in the "passwordchanges.log" file in the application server’s log folder.

Configuration:

- Open the JBoss CLI.
- Run the following commands:

For text format

```
/subsystem=logging/periodic-rotating-file-handler=PASSWORDCHANGE:add(append
           =true,file={"path"=>"passwordchange.log", "relative-to"=>"jboss.server.log.dir"},suffix
           =".yyyy-MM-dd",autoflush=true)
```

```
/subsystem=logging/periodic-rotating-file-handler=PASSWORDCHANGE:write-attribute(name
           =named-formatter,value=PATTERN)
```

```
/subsystem=logging/logger

 =pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMChangePasswordDef:add(level=DEBUG
```

```
/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMChangePasswordDef:
           add-handler(name=PASSWORDCHANGE)
```

For CSV format

```
/subsystem=logging/size-rotating-file-handler=PASSWORDCHANGE:add(append
         =true,file={"path"=>"passwordchanges.csv", "relative-to"=>"jboss.server.log.dir"},
         max-backup-index=1,rotate-size=10000k,autoflush=true)
```

```
/subsystem=logging/logger=

 pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMChangePasswordDef:
         add(use-parent-handlers=false,level=DEBUG)
```

```
/subsystem=logging/logger
```

```
 =pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMChangePasswordDef:

          add-handler(name=PASSWORDCHANGE)
```

Logging for password resets

Each time the password of a user is reset, a log is created and is stored in the "passwordresets.log" file in the application server’s log folder.

The log contains date/time of the password reset and a message: “Password for account <UserName> was reset by account <UserName of account that changed the password>”

Configuration:

- Open the JBoss CLI.
- Run the following commands:

For text format

```
/subsystem=logging/periodic-rotating-file-handler=PASSWORDRESET:add(append

                   =true,file={"path"=>"passwordreset.log", "relative-to"=>"jboss.server.log.dir"},suffix

                   =".yyyy-MM-dd",autoflush=true)
```

```
/subsystem=logging/periodic-rotating-file-handler=PASSWORDRESET:write-attribute(name

                   =named-formatter,value=PATTERN)
```

```
/subsystem=logging/logger

 =pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMResetPasswordDef:add(level=DEBUG)
```

```
/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMResetPasswordDef:

                   add-handler(name=PASSWORDRESET)
```

For CSV format:

```
/subsystem=logging/size-rotating-file-handler=PASSWORDRESET:add(append

              =true,file={"path"=>"passwordreset.csv", "relative-to"=>"jboss.server.log.dir"},
            max-backup-index=1,rotate-size=10000k,autoflush=true)


/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMResetPasswordDef:

            add(use-parent-handlers=false,level=DEBUG)


/subsystem=logging/
logger=pnlog.AUTHENTICATION.nl.planon.hades.businessmodel.account.bom.BOMResetPasswordDef:

            add-handler(name=PASSWORDRESET)
```

Security logging

- If you want to log the login and logout timings of the users (in the selected user group), go to the **User groups** TSI > **User groups** and enable the **Additional security logging** field by selecting **Yes**. This data will be stored in the security logging file. To enable this feature, you must first make the security logging settings.
For more information, refer to **Authorization** > **Security logging**.
- The **Additional security logging** setting is not enabled by default because it comes with a performance penalty. If you start logging for each user, this will create a considerable overhead. Only enable this setting if you need the information to retrieve more detailed information for analyzing issues.

Enabling security logging

In addition to audit logging, it is possible to monitor security.

To switch on security logging, apply the following configuration:

1. Start the JBoss CLI.
2. Run the following command to update the logger category AuditLogReporter:

```
/subsystem=logging/logger=pnlog.DEFAULT.nl.planon.auditlogging.AuditLogReporter:

write-attribute(name=level,value="INFO")
```

3. To switch it off, set it back to "WARN".

Security logging: change file name or location

By default security logging logs to ...\Server\wildfly-*\standalone\log\securityaudit.log.

To change the file name or location, proceed as follows:

1. Start the JBoss CLI .
2. Update the handler for the AuditLogReporter, SECURITY_AUDIT. Set the file property to the desired value, for example:

```
/subsystem=logging/periodic-rotating-file-handler=SECURITY_AUDIT/:write-attribute(name

=file,value={"relative-to" => "jboss.server.log.dir","path" => "Test/Test.log"})
```

Logging for anonymization

When an account is anonymized, an entry describing this action will be included in the audit log. The log will display a date-time stamp, the name of the account that performed the action and the name of the account that was anonymized.

```
12-01-2018:09:00 ~|~LABOUT-Lars Bout~|~Account anonymized~|~PAWIFI-Patrick Wifian~|
~NULL~|~
```

> This feature allows system administrators to check the status of accounts and find out what action was undertaken and by whom.

What is logged?

Events that are logged

- Licensing changes
  - Linking or unlinking a solution license to a user group.
- Authentication changes
  - Adding/deleting a user
  - Adding/deleting a user group
  - Adding/deleting users from a user group
  - Resetting or changing a user’s password

    - Adding/deleting a product from a user group
    - Updating a user group
- (Failed) user actions
    - Failure to log on by a user
    - When a user tries to log on and his/her account is locked
    - First time user login
    - Failed user login (inactive start / end date)
    - When an account is locked (multiple wrong logins, end date, password expired)
- Password settings
    - Changing the password strength
    - Changing password settings
- Logging in/out Planon administrators
    - Logging of login/log off of users linked to the Planon administrator group

> - If you want to log the login and logout timings of the users (in the selected user group), go to the **User groups** TSI > **User groups** and enable the **Additional security logging** field by selecting **Yes**. This data will be stored in the security logging file. To enable this feature, you must first make the security logging settings.
> For more information, refer to **Authorization** > **Security logging**.
> - The **Additional security logging** setting is not enabled by default because it comes with a performance penalty. If you start logging for each user, this will create a considerable overhead. Only enable this setting if you need the information to retrieve more detailed information for analyzing issues.

- Authorization changes
    - Switching on/off authorization
    - Switching on/off business object authorization
    - Changes to user group permissions:
        - Adding/deleting/updating action filters
        - Updating authorization filters (when linked to user group)
        - Adding/deleting/updating authorization links
    - Updating function profiles (when linked to a user group)
        - Adding/deleting field rights
        - Updating field rights
        - Adding/deleting actions
        - Adding/deleting status transitions
        - Adding/deleting extended actions
        - Changing the permission type of BORight

        - Changing the function profile default permission type

Environment Management gadget

In the Cloud, the following events are logged:

- **Disk**
    - Changing disk space allocation
- **Customize**
    - Changing the welcome image
    - Changing the welcome image to default
    - Changing the favicon image
    - Changing the favicon image to default
    - Changing the error page URL
- **Backups**
    - Creating manual backup
    - Restoring backup (including backup ID, destination and with or without resetting user password)
    - Deleting a backup
    - Converting a backup from *incremental* to *full*
    - Changing a backup name
    - Changing a backup expiry date
    - Changing a backup comment
    - Changing retention period
    - Changing **Save disk space** (toggle for: incremental full backups)
- **Danger zone**
    - Restarting
    - Scheduling a restart
    - Canceling a scheduled restart
    - Upgrading
    - Scheduling an upgrade
    - Canceling a scheduled upgrade
    - Reloading TMS
    - Changing the WebDAV password
    - Reseting NYX credentials

    - Importing a clone
    - Creating a clone voucher
    - Importing Accelerator
- **IP whitelisting**
    - Enabling IP whitelist
    - Disabling IP whitelist
    - Changing IP whitelist settings
- **SSO**
    - Enabling the SSO realm
    - Enabling SSO
    - Disabling SSO
    - Resetting the SSO Admin password
- **Domain settings**
    - Changing the domain alias setting
    - Changing mutual SSL settings
    - Changing portal URL settings

Tanuki service wrapper configuration

Planon configuration

The default configuration of the Tanuki service wrapper has been extended by Planon. Please find the *-wrapper-default.conf file at the following locations:

\Server\tanuki\webserver\conf

\Server\tanuki\appserver\conf

Generic configuration changes

- On top of the configuration file some parameters are defined by default. They are referenced in the file with the following concept %parameter name%. You can use this concept for own parameters too. The parameters are formatted as follows. For example application server Java home: set.JAVA_HOME=[JAVA_HOME]. For all parameters see the configuration files.
- On startup of the service details about environment and JVM are logged:

```
wrapper.environment.dump=TRUE

wrapper.java.version.output=TRUE
```

- Ignore gaps in numbering in wrapper.java.additional parameters:

```
wrapper.ignore_sequence_gaps=TRUE
```

- The option wrapper.ping.timeout already existed and it is set to 1800 seconds (half an hour). Previously, the service would try to restart the JVM after that time. Now, this behavior is extended to also log a thread-dump:

```
wrapper.ping.timeout.action=DUMP,RESTART
```

- Log a warning if the JVM does not respond within this period (number of seconds).

```
wrapper.ping.alert.threshold=10
```

- The following parameters can be used to limit the size of the log file and to set a maximum size so that the file size does not grow unmanageably:

```
wrapper.logfile.maxsize=10m

wrapper.logfile.maxfiles=10
```

The log file size may be abbreviated as 'k' (kb) or 'm' (mb) as the suffix for the integer. For example, 10m = 10 megabytes.

The default value is set to 10m and 10 files.
Setting the maximum size to the log file is not mandatory.

To allow the log file to expand unrestrictedly, remove the previous two lines from the jboss-wrapper-default.conf file, but be aware that this can cause disk space issues in future.

Tanuki for the application server configuration changes

- The following parameters define the memory settings:

```
# Initial Java Heap Size (in MB)

wrapper.java.initmemory= 2048

# Maximum Java Heap Size (in MB)

wrapper.java.maxmemory= 2048
```

Initially, the minimal and maximum settings are equal to provide the maximum memory size to the server immediately.

For reasons of memory usage, you could set a lower minimum size, but this is not recommended (to prevent out-of-memory).

You could set a higher maximum size.

Additional configuration possibilities

For a very useful documentation of all the properties coming with the Tanuki service wrapper, refer to: http://wrapper.tanukisoftware.com/doc/english/properties.html

Software health check

The URL endpoint **/health** allows you to verify if your servers are up and running. A round-trip is executed from webserver to application server and database server, and back. The endpoint **/health** page returns **OK** if everything is up and running, and if that is not the case, **NOK** is returned.

Procedure

1. Make sure that there has been a valid login on the webserver. The /health URL only functions properly after one valid login on the webserver. Each time the webserver is restarted, one login is needed to have the URL work properly again.

> ❗ If there has been no valid login on the webserver, the **/health** URL returns **OK** but *no* round-trip is done and no check is performed!

2. Go to the endpoint /health.

   The check is now performed.

Data collection

Within the Planon application, data that is collected pertains to:

- License usage (as covered in contracts);
- System events (these are technical events such as server start-up or unhandled errors, such as Java exceptions);
- Anonymous interactions of users with Planon that relate only to software usage and are used as input for product improvements by Planon.

No personal (PII/PSI) data and no customer data is ever collected.

Data collection for on-premise customers

Loading a license in an on-premise Planon environment (L91 or later), automatically invokes settings that -under certain circumstances (if outbound traffic to the public Internet is not is blocked)- enable data collection.

A Planon Administrator can turn this off again by adding the **Data collection** TSI and disabling it.

Data collection cannot be disabled for Planon Cloud environments. Nor can it be disabled *prior to* loading a license containing the data collection settings.

Procedure

1. Add the **Data collection** TSI to a navigation group in your navigation panel.
2. Log out / Log in and go to **Data collection**.
3. Set the **Tracking enabled?** field to **No**.

   You have now disabled data collection entirely for your environment.

Index