tts performance suite

tts performance suite 2026 - Server installation manual

Contents

1 General Information

1.1 Introduction

1.1.1 Objective

This document describes the requirements of the tts server components, named Curator and WebAccess, and their installation process.

The goal of this document is to provide an understanding of the server configuration and to guarantee its successful installation.

1.1.2 Target audience

System administrators, developers and all interested parties

1.1.3 Prerequisites

It is expected that system administrators, developers and who else may be interested have good knowledge of

- Windows and/or Unix-based operating systems
- Administration and handling of database management systems (Microsoft SQL Server/Oracle)
- Administration and handling of application server Apache Tomcat
- Deployment of web applications (WAR, external web application)
- SQL, HTML, XML

⚠ **Please read this installation guide completely and carefully!**

1.1.4 Contact

- tts Support
  Phone: +49 (0) 2 21 / 17 09 30 - 110
  Fax: +49 (0) 2 21 / 17 09 30 – 170
  `support@tt-s.com`
- Application consultant
  Application consultants are very experienced with the tts performance suite and can guide you through the installation process, providing professional solutions to match your requirements. If no application consultant has yet been assigned, please contact your key account manager.

1.1.5 Structure

The first chapter sheds light on the functionalities and technologies used in the Creator and finished by presenting the System overview and deployments scenarios.

The next section points to the System requirements concerning hardware and software. They should be checked thoroughly and carefully.

The Preparations chapter contains information on the prerequisites to get the tts server running.

The Installation chapter guides you step-by-step through the deployment of the tts server components, including all necessary actions like creating database connections or defining server configuration. Post installation steps deals with initial administration tasks. This chapter closes

with a Troubleshooting section, which provides solutions for common problems. Due to the fact that the tts server is a highly complex application, many configuration options are provided. A variety of other settings are explained in detail in the Appendix. Each property or parameter of the application's services is listed with its name, a description and the possible values.

1.1.6 Conventions

1.1.6.1 Symbols

To highlight important information on the one hand, and "nice to know" details on the other, the following icons are used:

 Attention

 Hint or note

 Tip

1.1.6.2 Abbreviations

Abbreviation	Description
DB	Database
JNDI	Java Naming and Directory Interface
LDAP	Lightweight Directory Access Protocol
SOLR	Solr Search Engine
SSO	Single-Sign-On
WAR	Web Archive

1.1.6.3 Variables

 Variables are marked with a leading $

Variable	Description
$TTPS_HOME	installation directory of tts performance suite
$TOMCAT_HOME	installation directory of Apache Tomcat
$SOLR_INDEX	index directory of Apache Solr for the tts performance suite

1.1.7 Your feedback is highly welcome

tts welcomes your feedback concerning the quality and usefulness of this manual. Your comments and suggestions will be considered as valuable input for future revisions of this manual.

- Found an error? Please let us know where.
- A topic is not described clearly enough? Please let us know which one.
- Need more information? On which topic?
- Something doesn't work for you? Please let us know, so we can provide additional examples.

Please feel free to send us your feedback: `support@tt-s.com`. We appreciate your help!

1.2 Application description

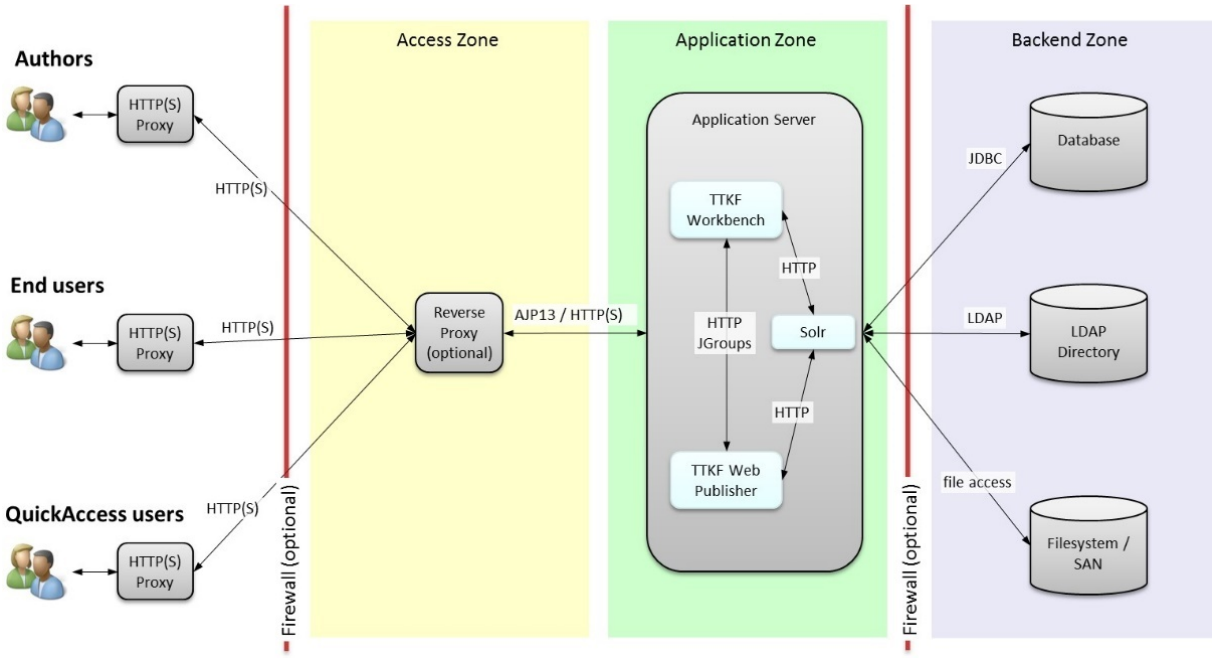
The tts server consists of two components: the Curator and WebAccess. Representing a production and delivery platform, the Curator provides author functionalities, whereas the WebAccess makes the content accessible to the end users.

The server components have been developed in a platform-independent way, using technologies like Java, HTML, JavaScript, CSS, XML, and XSLT (to create platform-independent document types like RTF or PDF). There are only a few components that are platform- dependent, like the 'recorder' software or the 'QuickAccess for Desktop Applications' online- help-system, due to the technology used in them.

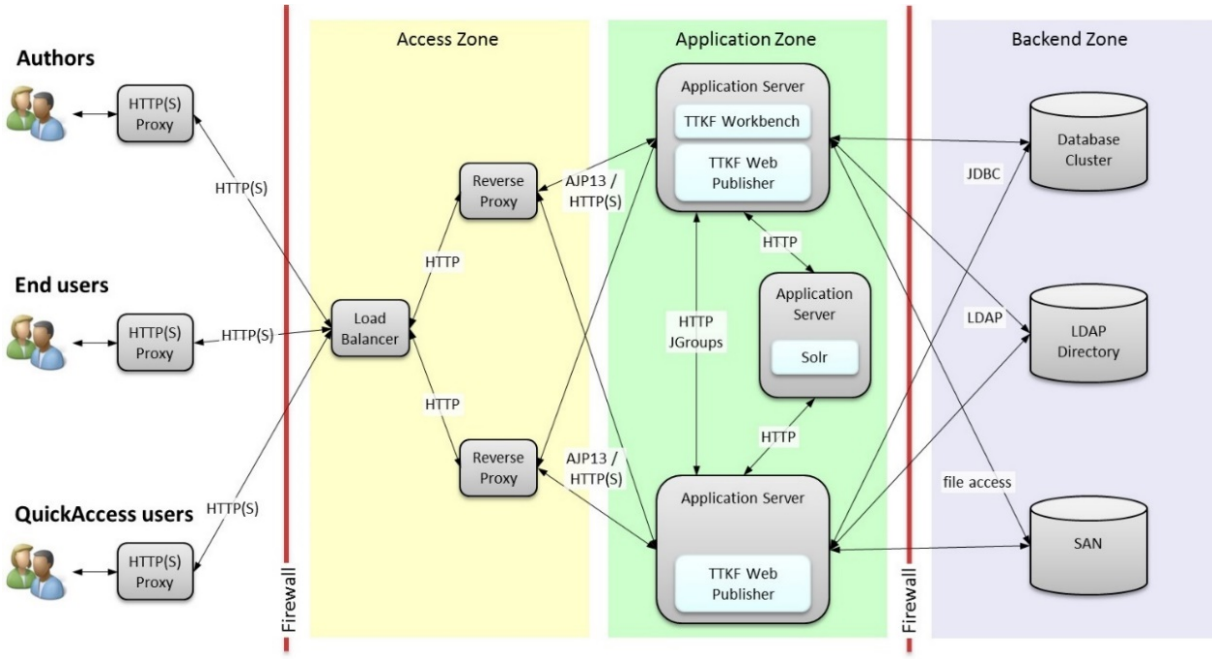
As a result of the server component's architecture, it can take little effort to integrate the tts server in existing environments

1.3 System overview and deployment scenarios

This chapter offers a short outline of the tts Server system environment in two different deployment scenarios.



The first scenario (Figure 1), which is the minimum way in practice, involves one application server hosting both Curator and WebAccess, along with the Solr web application. The backend provides a database, file system (as repository), and an optional LDAP directory for authentication and authorization purposes. End users access the application via HTTP(S).



The second scenario (Figure 2) differs from the first one by being more scalable and dynamic, thanks to using three application servers plus a load balancer that distributes all requests among the reverse proxies. One application server hosts the Curator and one (Preview-) WebAccess. The other server(s) run(s) the main WebAccess. The third one provides with Solr web application the search engine.

Both scenarios, minimum and recommended, are field-tested concepts. The usage depends on the customer's requirements. Nevertheless, we recommend the deployment including a reverse proxy in any case. Moreover, if there is a huge number of end users expected, an installation in a clustered environment is advised, since this scenario offers more flexibility, better scalability, and higher performance.

When using AJP, please read the chapter Configure websocket support for Apache Web-server with AJP in the appendix.

2 System requirements

2.1 Hardware and Software requirements

For the current hardware and software requirements of tts server, please refer to the system requirements.

3 Preparations

3.1 License

Before you start the installation process, a valid license for the tts server is required. Please request the license file via your professional service consultant or tts Support (support@tt-s.com). Please provide the MAC addresses as well as the IP addresses of the servers on which the components are going to be installed.

⚠ Both components, Curator and WebAccess, need their own license file. If you plan to install the applications on separate machines, the MAC and IP address of each server has to be provided.

3.2 Database

To store data persistently, the tts server components require a relational database. Therefore, a new database must be created in your database management system (DBMS).

A database user is needed to access the database. The user must be granted particular permissions for the database, explained later, as well as permissions for Microsoft SQL Server and Oracle.

The user will be configured in the local application context XMLs. Integrated Authentication utilizing an AD user logged into Tomcat is also supported.

With regard to the server configuration, the following pieces of information are required:

- Host name or IP address of the database server
- TCP port of the database service
- Name of the database
- Names and passwords of the database users

⚠ UTF-8 should be used as character set to store data in the database. If the database is used primarily for languages with a Unicode range higher than U+0800 (see www.unicode.org/charts for more information), a UTF-16 character set should be used.

3.2.1 JDBC Driver

Currently, the following JDBC drivers, as provided by the database vendors, are supported:

- Microsoft SQL Server JDBC Driver: latest, compatible JDBC driver
- Oracle JDBC Driver: latest, compatible JDBC driver

We recommend using a driver that matches the database version, particularly for Oracle, as well as the installed Java version.

3.2.2 Microsoft SQL Server

The database user must have the permission to connect to the newly created database. This is usually done during user creation by setting the standard database. If not, access to the database has to be granted separately via the user properties.

The database user should be granted dbo rights, since he needs to create, alter and delete tables, procedures, functions, indices and so on.

Be aware that the port of the database might be set dynamically for a named instance. It may therefore deviate from the default value 1433.

3.2.3 Oracle Database

In Oracle, a tablespace must be created first. The related data file should have a size of at least 50 MB. To automatically assign more disk space, the "AUTOEXTEND" option must be activated. Otherwise, the database cannot store additional data once the space limit is reached.

The database user must be granted the following permissions:

```
SYSTEM PERMISSION: ROLE: CONNECT
SYSTEM PERMISSION: CREATE SESSION
SYSTEM PERMISSION: CREATE TABLE
SYSTEM PERMISSION: CREATE SEQUENCE
SYSTEM PERMISSION: CREATE PROCEDURE
```

3.2.3.1 Oracle 23ai

In Oracle 12c an optimizer which can produce wrong query results in the tts performance suite was introduced. It should be disabled by these two parameters:

```
alter system set optimizer_adaptive_statistics=false
alter system set optimizer_adaptive_plans=false
```

The database user must also be granted unlimited quota for the created tablespace.

Upon request, tts Support can provide you with SQL scripts that can be used to generate the required tablespace and user, and to set the specific rights for the user.

3.3 File System

The following directories are required for the tts server components. One for the repository to save the document contents. The other one is used as a data store for runtime files. The user of the application server needs complete access to those directories to read, write, execute and delete files.

Sample File Structure

```
$TTPS_HOME/store
```

Please refer to the hardware requirements section for information on the hard disk space that is required for both the repository and the data store.

Please make sure the created directories are accessible to all installed instances of the tts server. This also holds true if the instances are running on different machines.

3.4 Solr installation

The tts performance suite infrastructure uses Apache Solr™ for indexing and searching.

Apache Solr™ is a web application available from the Apache Software Foundation. Please install Solr according to their installation instructions: https://solr.apache.org/guide/solr/9_10/deployment-guide/installing-solr.html

⚠ The tts performance suite requires a Solr version 9.10.1. Please use the standard distribution, not the slim version.

⚠ Please use a dedicated Solr instance for the tts performance suite infrastructure.

⚠ Please make sure that Solr is always started before Apache Tomcat.

⚠ For Windows servers we recommend to install the Solr as a service using NSSM. A detailed installation can be provided by Professional Services.

✎ You will need the location of the Solr configuration files, which contain the index. We will refer to that location as $SOLR_HOME. The exact location may differ, depending on your Solr installation, as explained in https://solr.apache.org/guide/solr/9_10/configuration-guide/configuration-files.html

✎ If you prefer to use a custom location for your index, you will need to configure Solr accordingly, and you will need to copy the file solr.xml from the default location into your dedicated $SOLR_HOME.

✎ You will need the path to the Solr installation. We will refer to that location as `$SOLR_INSTALL`

You will need the following properties for the tts Server Components installation:

- $SOLR_HOME, e.g., `C:\solr\server\solr`
- $SOLR_INSTALL, e.g., `C:\solr`
- Solr's URL and Port, e.g., http://127.0.0.1:8983/solr

After installing Solr, you can verify the installation as follows:

- Start Solr, if necessary
- Open the Solr main URL in your browser, e.g., http://127.0.0.1:8983/solr
- Check that the Solr dashboard shows the expected version

3.5 MinIO installation

The tts performance suite infrastructure uses MinIO as interface for the filesystem. MinIO provides a S3-like access to common filesystems, giving advantage of the S3 protocol on-premise, too.

✎ For more Information on MinIO please refer to the official MinIO documentation https://min.io/

3.5.1 Installation on Windows

Download the currently supported MinIO version (please see the System Requirements) from the MinIO download archive: https://dl.min.io/server/minio/release/windows-amd64/archive/ and rename the downloaded file to minio.exe.

The directory you have set for MinIO data in the configuration step will be referred to as < `MINIO_DATA`>.

To start MinIO run the following command.

```
minio.exe server /path/to/folderContaining<MINIO_DATA>/
```

The MinIO server is now started with the default credentials. To change the credentials run the following command.

```
set MINIO_ROOT_USER=newKey
set MINIO_ROOT_PASSWORD=newSecretKey
```

⚠ For windows servers we recommend to install MinIO as a service utilizing NSSM https://nssm.cc/. A detailed installation description can be provided by Professional Services.

⚠ Please make sure that MinIO is always started before Apache Tomcat.

3.5.2 Installation as a Linux Service

The following steps will install MinIO as a SystemD backend service.

The following values are used throughout the setup and are initialized with default values which may be overridden.

```
MINIO_INSTALL_DIR=/usr/local/bin
MINIO_DATA_DIR=/var/minio
MINIO_ENV_FILE=/etc/default/minio
MINIO_GROUP_NAME=minio MINIO_USER_NAME=minio
MINIO_PORT=9000
BUCKET_NAME=bucket
```

Add a user and group to run minio.

```
groupadd $MINIO_GROUP_NAME
useradd -s /sbin/nologin -d /home/minio $MINIO_USER_NAME -g
$MINIO_GROUP_NAME
```

The next steps require these directories to exist.

```
mkdir -p $MINIO_INSTALL_DIR
mkdir -p $MINIO_DATA_DIR
mkdir $MINIO_DATA_DIR/$BUCKET_NAME
chown -R $MINIO_USER_NAME:$MINIO_GROUP_NAME $MINIO_DATA_DIR
```

Download the currently supported MinIO version (please see the System Requirements) from the MinIO download archive: https://dl.min.io/server/minio/release/windows-amd64/archive/ and set the permissions:

```
get https://dl.min.io/server/minio/release/linux-amd64/archive/minio.RELEASE.<
↪ VERSION_NUMBER> -O $MINIO_INSTALL_DIR/minio
```

```
chown -R $MINIO_USER_NAME:$MINIO_GROUP_NAME $MINIO_INSTALL_DIR
chmod u+x $MINIO_INSTALL_DIR/minio
chown -R $MINIO_USER_NAME:$MINIO_GROUP_NAME $MINIO_DATA_DIR
```

The basic configuration of the MinIO server, including the credentials, is contained in the `MINIO_ENV_FILE`. Replace the values for `MINIO_ROOT_USER` and `MINIO_ROOT_PASSWORD` with unique, random, secure values.

```
cat <<EOT > $MINIO_ENV_FILE
MINIO_VOLUMES="$MINIO_DATA_DIR".
MINIO_OPTS="--address :$MINIO_PORT"
MINIO_ROOT_USER=tempkeyid
MINIO_ROOT_PASSWORD=tempkey123456789
EOT
```

```
chown $MINIO_USER_NAME.$MINIO_GROUP_NAME $MINIO_ENV_FILE
```

With the above-mentioned environment variables, create this SystemD service file.

```
cat <<EOF > /etc/systemd/system/minio.service
[Unit]
Description=MinIO
Documentation=https://docs.min.io
Wants=network-online.target
After=network-online.target, cloud-final.service
AssertFileIsExecutable=$MINIO_INSTALL_DIR/minio

[Service]
WorkingDirectory=$MINIO_DATA_DIR
User=$MINIO_USER_NAME
Group=$MINIO_GROUP_NAME

EnvironmentFile=$MINIO_ENV_FILE
ExecStartPre=/bin/bash -c "if [ -z \"\${MINIO_VOLUMES}\" ]; then
echo
\"Variable MINIO_VOLUMES not set in $MINIO_ENV_FILE\"; exit 1; fi"

ExecStart=$MINIO_INSTALL_DIR/minio server \$MINIO_OPTS
\$MINIO_VOLUMES

# Let systemd restart this service always
Restart=always

# Specifies the maximum file descriptor number that can be opened by
this process
LimitNOFILE=65536

# Disable timeout logic and wait until process is stopped
TimeoutStopSec=infinity
SendSIGKILL=no

[Install]
```

```
WantedBy=multi-user.target
EOF
```

You can now enable and run the MinIO service using systemctl.

```
systemctl enable minio
systemctl start minio
```

You can use the standard commands from systemctl to manage MinIO.

```
systemctl start minio
systemctl stop minio
systemctl restart minio
systemctl status minio
```

The output can be retrieved from journal.

```
journalctl -u minio
```

4 Installation step-by-step

4.1 Checklist

Before you start deploying the tts performance suite server, please ensure that the following requirements are met:

- The installation files are available.
- The licenses for the components are available.
- Database and Database user(s) have been created.
- All required paths on file system are available.
- Details of distributed caching (Multicast-IP-address and port and so on).
- The application server is installed properly.
- The Solr search engine is installed, and $SOLR_INDEX is configured.
- The MinIO Server is installed and configured

⚠ **Before you start tts performance suite server for the first time, you are strongly advised to read the Post installation steps chapter.**

4.2 Deployment of the Solr core

⚠ A Solr installation is required for this step. Please see *Solr installation*.

⚠ If updating an existing system, please delete the index before continuing.

In this step, you deploy the preconfigured Solr core and the necessary plugins to your Solr installation.

1. Stop the Solr service, if it is running.
2. Copy the `tts-folder` from the artifact `solr.zip` into the $SOLR_HOME directory (see *Solr installation*).
3. Confirm the following directory structure:

```
$SOLR_HOME/tts-server/conf
$SOLR_HOME/tts-server/conf/schema.xml
$SOLR_HOME/tts-server/conf/solrconfig.xml
$SOLR_HOME/tts-server/conf/stemdict.txt
```

4. Optional: You can change the name of your Solr core by renaming the tts-server. In that case, please use the new name in the following instructions.
5. Copy the single jar file ttps-ttkf-server-solr-config-*.jar from tts-plugins folder in the solr.zip into the general Solr lib directory $SOLR_INSTALL/lib (see *Solr installation*).
6. Copy the following jar files from the directory $SOLR_INSTALL/modules/analysis-extras/lib to the general Solr lib directory $SOLR_INSTALL/lib:

```
solr-analysis-extras*.jar
lucene-analysis-icu*.jar
icu4j*.jar
```

7. Configure the `max boolean clauses` property to 4096 or higher:

- Either pass as a JVM argument: `-Dsolr.max.booleanClauses=4096`
- Or define an environment variable: `SOLR_MAX_BOOLEAN_CLAUSES=4096`
- Or define it in $SOLR_HOME/solr.xml and in $SOLR_HOME/tts-server/conf/solrconfig.xml, without reference to an environment variable *in both files*

8. Start Solr.
9. Visit `http://[solrUrl]:[solrPort]/solr/#/[core]` to confirm Solr is running and the required core is present, e.g., *http://localhost:8983/solr/#/tts-server*

⚠ If you do not have the Solr configuration files, please consult your application consultant.

4.3 Installation environment

You need to configure the tts server components before you can deploy and start the application successfully. The basic configuration comprises **server properties**, **database connection**, and **file system paths**. Once these settings have been provided, the server can be run in default mode.

Additional settings may be necessary for services and features like user authentication and authorization, caching, or workflow and versioning. They are explained in detail in the appendix.

The actual configuration is carried out via a file named *application-config.properties*. Please note that the application has a fallback mechanism to load configuration parameters by scanning several external locations before using the system's default value:

1. Properties file with default name *application-config.properties*
2. Context parameters
3. JVM arguments

4.3.1 Set up installation environment

As best practice, we recommend setting up a simple installation environment to keep things clear. Therefore, just create the following folder structure:

```
$TTPS_HOME/apps
$TTPS_HOME/logs
$TTPS_HOME/repository
$TTPS_HOME/store
$TTPS_HOME/temp
```

The *apps* folder is there to store the single .WAR files of the Curator and the WebAccess.

The *logs* directory may contain generated log files, if you configure it that way. Especially if the application server does not unpack the .WAR files during deployment, we will need to configure the application to write logs in that folder.

The application's file repository will be setup within repository; runtime files are located in *store*. For temporary installation purposes, files may be stored in *temp*.

Of course, you may place these folders wherever you like, but please consider at least keeping the document content within repository, separate from runtime files in *store*.

At last, we place the server license file and the application-config.properties directly in $TTPS_HOME.

4.3.2 Configuring the tts performance suite using application-config.properties

4.3.2.1 What is application-config.properties

The *application-config.properties* file is actually a simple text file with a .properties extension, serving - among other things - especially as a configuration mechanism for Java applications. Within this file, you can define application parameters using key-value-pairs separated by a '='.

Defining parameters is subject to simple rules:

- The file must have a *.properties* extension
- Parameter and value are separated with an '=' on a single line
- Parameter and value are valid
- Parameters for Curator match the following syntax:
  `ttkf.server.<servicename>.<propertyname>` or
  `ttkf.integrator.<servicename>.<propertyname>`
- Parameters for WebAccess match the following syntax:
  `ttkf.server.<servicename>.<propertyname>` or
  `ttkf.accelerator.<servicename>.<propertyname>`
- File paths must be separated with a slash "/" instead of backslash "\"

The following example shows a snippet of an *application-config.properties* file with a configuration for a SQL Server datasource:

```
1  # curator database configuration
2  ttkf.server.data.hibernate.default_schema=tts
3  ttkf.server.data.hibernate.dialect=de.tts.bd.business.data.UnicodeSQLServerDialect
4
5  # webaccess database configuration
6  ttkf.accelerator.data.portal.hibernate.default_schema=tts
7  ttkf.accelerator.data.portal.hibernate.dialect=de.tts.bd.business.data.
↪ UnicodeSQLServerDialect
8
9  # store directory
10 ttkf.server.store.base.directory=/ttkf/datastore
```

4.3.2.2 How to make application-config.properties accessible?

There exist two options to provide the application with the configuration file externally. Either you extend the class path so the application will find that properties file; or you directly define its path within the context file as a parameter.

For better maintenance, we suggest to configure the path to the location of *application-config.properties* directly in the corresponding context file.

To do so, add the *ttkf.server.properties* context parameter in the context file during deployment (see the deployment chapter):

```
1  ...
2  <Parameter name="ttkf.server.properties" override="false" value="C:/tts␣
↪ performance␣suite/application-config.properties" />
3  ...
```

4.3.2.3 Example of a minimum configuration

The following example illustrates the installation of the tts server components using an Oracle database.

Database connection

Initially, you configure the database connection. Since we want to use a data source, we just need to set hibernate dialect and default schema. The internal JNDI name to which the data source is bound is *java:comp/env/jdbc/TTKFDS*.

```
ttkf.server.data.hibernate.dialect=de.tts.bd.business.data.UnicodeOracle12cDialect
ttkf.server.data.hibernate.default_schema=TTS
```

We define the connection properties for WebAccess as well:

```
ttkf.accelerator.data.portal.hibernate.dialect=de.tts.bd.business.data.
UnicodeOracle12cDialect
ttkf.accelerator.data.hibernate.portal.default_schema=TTS
```


See the Database service chapter in the appendix for more details.

Application properties

We begin by defining the context names of Curator and WebAccess. Since the context names can be chosen freely, it is necessary for intercommunication of the applications to know the context name of each other.

```
ttkf.server.properties.acceleratorContextPath = /webaccess
ttkf.server.properties.integratorContextPath = /curator
```

Next, we set the location of license file, the directory of the application server's logs and the super user IPs.

```
ttkf.server.properties.licenseFile=file:C:/ttps/server.ttlk
ttkf.server.properties.applicationServerLogDirectory=C:/ttps/logs
ttkf.server.properties.superuserIPs=127.0.0.1,192.168.85.48
```

Because only the Curator can re/index entities with Solr, the Curator and the WebAccess have to know each other's URL.

```
ttkf.server.properties.internal_acceleratorURL = http://192.168.86.44:9980/
webaccess
ttkf.server.properties.internal_integratorURL = http://192.168.86.44:9980/curator
```


See the Properties service chapter in the appendix for more details.

Solr

This mandatory parameter defines how the Curator and WebAccess can reach the Solr server.

```
ttkf.server.search.server.url=http://127.0.0.1:8983/solr/tts-server
```


See the Search Service chapter in the appendix for more details.

Data store

Now we need to configure the base directory of the runtime files. So, we just define the location of our created folder $TTPS_HOME/store.

```
ttkf.server.store.base.directory = C:/ttps/store
```


See the Store service chapter in the appendix for more details.

Distributed cache

To get the distributed cache working properly we need to configure several parameters. Be sure to have information about the mode of the distributed cache (unicast/multicast) and the corresponding IP-addresses and ports ready.

We will configure multicast, which is the default mode.

```
ttkf.server.cache.peers.multicastGroupPort=45637
ttkf.server.cache.peers.multicastGroupAddress=228.8.18.9
```


See the Cache service chapter in the appendix for more details.

Repository

For the use of MinIO certain parameters are needed.

To authenticate with MinIO:

```
ttkf.server.repository.accessKeyId = {MinIO credentials}
ttkf.server.repository.secretKey = {MinIO credentials}
```

Configure repository location:

```
ttkf.server.repository.url=s3://bucket/repository
```


See the Repository service chapter in the appendix for more details.

4.3.2.4 How to escape special characters within the .properties file

The encoding of a .properties file is per definition ISO-8859-1, also known as Latin-1. All non-Latin-1 characters must be entered using Unicode escape characters, e.g. \uHHHH, where HHHH is a hexadecimal index of the character in the Unicode character set. A non-Latin-1 text file can be converted to a correct *.properties* file by using the *native2ascii* tool that ships with the JDK.

4.4 Data source

4.4.1 Creating a data source with Apache Tomcat

In Apache Tomcat, you can create a data source in a very simple way by adding it as a resource. As best practice, it is recommended to define this resource in both files *webaccess.xml* and *curator.xml*, located at *$TOMCAT_HOME/conf/Catalina/localhost*.

Since version 8.5, Tomcat provides an improved database connection pool (DBCP2). As an alternative, you can also use the Tomcat JDBC Connection Pool, for this refer to https://tomcat.apache.org/tomcat-8.5-doc/jdbc-pool.html#Introduction.

In the following step-by-step example, a data source pointing to an Oracle database as well as a SQL Server database will be generated.

1. Stop the Tomcat if it is running.
2. Open the *webaccess.xml* (similarly for *curator.xml*) in *$TOMCAT_HOME/conf/Catalina/localhost*, where resources can be defined.
3. Add your data source as a resource like below example.

```
<!-- Syntax -->
<Resource
name="[JNDI-name]"
auth="[authentication␣type]"
type="[type␣of␣datasource]"
username="[name␣of␣db␣user]"
password="[password␣of␣db␣user]"
driverClassName="[class␣name␣of␣jdbc␣driver]"
url="[connection␣url]"
initialsize="[initial␣number␣of␣connections␣on␣startup␣of␣pool]"
minIdle="[minimum␣number␣of␣idle␣connections␣to␣be␣kept␣all␣time]"
maxTotal="[maximum␣number␣of␣connections␣at␣the␣same␣time]"
maxIdle="[maximum␣number␣of␣idle␣connections␣to␣be␣kept␣at␣all␣time]"
timeBetweenEvictionRunsMillis="30000"
minEvictableIdleTimeMillis="60000"
testOnConnect="true"
testWhileIdle="true"
validationQuery="select␣1␣from␣dual"
validationInterval="30000"
suspectTimeout="60"
logAbandoned="true"
removeAbandonedTimeout="120"
removeAbandoned="true"
abandonWhenPercentageFull="60"
jdbcInterceptors="ResetAbandonedTimer;SlowQueryReport"
/>
```

```
<!-- Example (Oracle) -->
<Resource
name="jdbc/GTTKFDS"
auth="Container" type="javax.sql.DataSource"
username="TTS"
password="TTS" driverClassName="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:XE"
initialsize="10"
minIdle="10"
maxTotal="100"
maxIdle="50"t
imeBetweenEvictionRunsMillis="30000"
testOnConnect="true"
testWhileIdle="true"
validationQuery="select␣1␣from␣dual"
```

```
validationInterval="30000"
suspectTimeout="60"
ogAbandoned="true"
removeAbandonedTimeout="120"
removeAbandoned="true"
abandonWhenPercentageFull="60"
jdbcInterceptors="ResetAbandonendTimer;SlowQueryReport"
/>
```

4. Continue deploying the applications.

4.4.2 Escaping special character

If it is necessary to use special characters within the tomcat configuration files, e.g., for passwords, the characters have to be xml escaped. A free online conversion tool can be used to convert the characters.

Incorrect example:

```
<Resource name=="jdbc/GTTKFDS"
username="TTS"
password="!"§$%&/()=?ß²"..>
```

Correctly escaped example:

```
<Resource name="jdbc/GTTKFDS" username="TTS"
password="!&amp;quot;&amp;#167;$%&amp;amp;/()=?&amp;#223;&amp;#178;"..>
```

4.4.3 MS SQL Server: Remarks on encryption

For MS SQL Server, the JDBC driver versions 10.2 and above use encrypted connections by default. Please refer to the MS SQL Server documentation for configuring the SQL Server Database Engine for encrypting connections for details on the database server configuration.

If you do not configure your database server to use certificates, and if there are no certificates installed on the database server's machine, MS SQL Server will use self-signed certificates as a fallback. In that case, you need to append the setting "trustServerCertificate=true" to the connection URL, so that the client (your JNDI data source) will trust the database server's certificates.

```
<!-- Example for an MS SQL Server connection URL with self-signed certificates -->
<Resource name="jdbc/GTTKFDS" username="…" password="…"
url="jdbc:sqlserver://hostname:1433…;;trustServerCertificate=true" …>
```

If you want to use unencrypted connections instead, you can set "encrypt=false" in the connection URL.

```
<!-- Example for an MS SQL Server connection URL with encryption explicitly
↪ disabled -->
<Resource name="jdbc/GTTKFDS" username="…" password="…"
url="jdbc:sqlserver://hostname:1433…;;encrypt=false" …>
```

⚠ For a production environment, we strongly recommend using encrypted connections with certificates from a trusted certification authority.

4.5 Application server settings

4.5.1 JVM settings

4.5.1.1 Memory

The JVM of each server instance must meet at least the following memory settings and might be adapted to the environment's requirements:

- Maximum Java heap size: 1024 MB (-Xmx1024m)
- Initial Java heap size: 256 MB (-Xms256m)
- Maximum thread stack size: (-Xss512k)
- Metaspace: -XX:MaxMetaspaceSize=512m
  Since Java 8 the Metaspace does not need to be set. So, either ignore the flag or set it to an appropriate value. In case of memory limitations, the value needs to be increased.

In case the web applications are deployed on the same Apache Tomcat, the following memory settings are at least to be set:

- Maximum Java heap size: 2048 MB (-Xmx2g)
- Initial Java heap size: 512 MB (-Xms512m)
- Maximum thread stack size: (-Xss512k)
- Metaspace: -XX:MaxMetaspaceSize=768m
  As mentioned above, this value does not need to be configured. But if it is set explicitly, it should be at least 768 MB. In some cases, 1 GB might be necessary.

4.5.1.2 Encoding

Depending on the Java distribution UTF-8 might not be the default encoding. Thus, the following JVM parameter must be set:

```
1 -Dfile.encoding=utf-8
```

4.5.1.3 Thread size ForkJoinPool

Depending on the deployment of the web applications Curator, WebAccess and Creator, the size of the parallel allowed threads for the ForkJoinPool must be checked and set properly.

How to configure ForkJoinPool (Decision Tree)

Does the host machine provide more or equal than 10 (logical) processors?

- Yes → Done!
- No → Are the web applications Curator, WebAccess and Creator deployed on the same Apache Tomcat, sharing the same JVM?
  - No → Done!
  - Yes → Provide JVM argument for 10 parallel threads:

```
1 -Djava.util.concurrent.ForkJoinPool.common.parallelism=10
```

4.5.2 URL Encoding

To ensure that parameters in URLs are interpreted correctly (as UTF-8), the useBodyEncodingForURI attribute must be defined. This is done in the *server.xml* file of the application server, by adding *useBodyEncodingForURI* attribute with *true* as its value in the http-connector. The default value is *false*.

Example:

```
<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort=
↪ "8443" useBodyEncodingForURI="true" />
```

4.5.3 Cookie Processor

Some cookies need to be shared across the separate web applications of the tts Performance Suite. Therefore, please define the following cookie policy with the CookieProcessor inside the *context.xml* (or in *curator.xml* and *webaccess.xml*):

```
<CookieProcessor sameSiteCookies="none" />
```

4.5.4 Error pages

The deployed WAR packages do not provide error pages for http error status codes, but will fall back to the error pages provided by the tomcat installation. This ensures a consistent layout of all error pages and allows for customization. The default error pages are shipped as a collection of static html files and need to be configured in the *server.xml* file of the application server:

```
<Host...>
...
<Valve className="org.apache.catalina.valves.ErrorReportValve"
 errorCode.404="path/to/error404.html"
 errorCode.403="path/to/error403.html"
 errorCode.500="path/to/error500.html"
 errorCode.0="path/to/error.html"
 showReport="false"
 showServerInfo="false"/>
...
</Host>
```

The `showReport` and `showServerInfo` attributes prevent an information leak when no specific error page could be displayed (for instance due to a wrong path). Thus, for security reasons it is strongly recommended to set them both to `false`.

4.6 MinIO configuration

4.6.1 Install MinIO

A MinIO installation is required for this step. Please see *MinIO installation*.

For this guide, you need the following:

- The local address of the MinIO server, e.g., *http://localhost:9000*, `<MINIO_INTERNAL>`

- If a load balancer or proxy is used, the external address as configured in the load balancer or proxy configuration, `<MINIO_EXTERNAL>`
- The MinIO data directory, `<MINIO_DATA>`
- The MinIO credentials, `<MINIO_ROOT_USER>` and `<MINIO_ROOT_PASSWORD>` Set during *MinIO installation*.

4.6.2 Create a bucket

Create a folder `<bucket>`, e.g., *data* or *bucket*, within `<MINIO_DATA>`

⚠ This is where the document repository and any stores you wish to keep behind MinIO will be.

Do not use uppercase or invalid characters like "_" for the bucket name.

4.6.3 Set the URL to the repository

The root path (`<path>`) will be the location for documents, e.g., `<MINIO_DATA>/bucket/repository/d5/86443033-22da-4489-9678-85d1348940e0/file.txt`

Note that `<path>` may not be empty.

Set the repository URL parameter in *application-config.properties*:

```
1 ttkf.server.repository.url=s3://<bucket>/<path>
```

Use the S3 URL format, for example:

```
1 ttkf.server.repository.url=s3://bucket/repository
```

4.6.4 Set the credentials for MinIO

The tts performance suite supports several methods for supplying credentials, the simplest of which is to provide the values in the *application-config.properties*. All methods are described here and the best method can be selected according to your requirements.

For each of the following methods you will need an access key (`<MINIO_ROOT_USER>`, sometimes called the Access Key ID) and a secret key (`<MINIO_ROOT_PASSWORD>`).

4.6.4.1 application-config.properties

Set the following parameters in the application-config.properties:

```
1 ttkf.server.repository.accessKeyId=<MINIO_ROOT_USER>
2 ttkf.server.repository.secretKey=<MINIO_ROOT_PASSWORD>
```

4.6.4.2 Environment variables

The credentials can be passed to the Server by setting the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_KEY`:

```
export AWS_ACCESS_KEY_ID=<MINIO_ROOT_USER>
export AWS_SECRET_KEY=<MINIO_ROOT_PASSWORD>
```

4.6.4.3 System properties

These properties may also be set in Tomcat's catalina.properties file:

```
aws.accessKeyId=<MINIO_ROOT_USER>
aws.secretKey=<MINIO_ROOT_PASSWORD>
```

4.6.4.4 AWS profile

The connection to MinIO can also be configured via an AWS profile. This can be done automatically (via the AWS Command Line Interface and the `aws configure` command) or manually, as described here.

Create a file `.aws/credentials` in your home directory with the following content:

```
[default]
aws_access_key_id=<MINIO_ROOT_USER>
aws_secret_access_key=<MINIO_ROOT_PASSWORD>
```

On *Linux*: ~/.aws/credentials
On *Windows*: %USERPROFILE%/.aws/credentials

4.6.5 Set the MinIO endpoint

Set the following parameters in application-config.properties:

```
ttkf.server.repository.endpoint.internal=<MINIO_INTERNAL>
ttkf.server.repository.endpoint.external=<MINIO_EXTERNAL>
```

Trailing slashes are not allowed.

External endpoint is necessary only when using a load balancer or proxy or when the internal endpoint is not accessible from the author machines running the Producer.

Make sure MinIO is directly accessible from the author machines, otherwise the upload of documents will fail.

HTTP requests (especially PUT) to MinIO from author machines must be possible.

4.6.6 Required MinIO version

For security reasons, we strongly recommend to install a current MinIO version.

4.7 Creator configuration

Please see the Creator Installation Manual for the steps to install the Creator.

4.8 Authorization Server (AuthServer) configuration

The tts server uses an authorization server to authorize internal communication via access and refresh tokens. This web application is based on Spring's Authorization server.

https://docs.spring.io/spring-authorization-server/reference/1.5/index.html

The AuthServer must be deployed as a separate web application (WAR) with the following parameters:

```
# the endpoint to validate the passed otp
ttkf.authserver.oauth2.workbench.otp.endpoint=http://localhost/workbench/authentication/consume/otp
# the url the AuthServer is served by (must be reachable from Curator/Creator)
spring.security.oauth2.authorizationserver.issuer=http://localhost/auth
# the client secret, choose a hard to guess value. The {noop} in front of the secret is required.
spring.security.oauth2.authorizationserver.client.curator.registration.client-secret={noop}secret
```

On Curator side, these parameters must be configured:

```
# the internal url of the AuthServer - reachable from Curator
ttkf.server.authserver.internal.url=http://localhost/auth
# the client secret, choose a hard to guess value (without the {noop}).
ttkf.server.authserver.client.secret=secret
```

4.9 Deployment

For Apache Tomcat, the deployment of the Curator is described step-by-step by way of example. All steps must be carried out for the WebAccess and AuthServer instances as well.

The example shows the deployment via a .WAR file. Usually, tts provides pre-configured .WAR files for both components, Curator and WebAccess.

1. Stop Tomcat if it is running.
2. Copy *curator.war* (from $TTPS_HOME/apps) to $TOMCAT_HOME/webapps.
3. Start Tomcat. Now the server will unpack the .WAR file into *$TOMCAT_HOME/webapps/curator*
4. Stop Tomcat.
5. Create the `curator.xml` file in the `$TOMCAT_HOME/conf/Catalina/localhost` directory.
6. Configure the data source and the location of the *application-config.properties* within this context file.

   Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/curator" reloadable="false">
<ResourceLink type="javax.sql.DataSource" name="jdbc/TTKFDS" global="jdbc/GTTKFDS" />
```

```
<Parameter name="ttkf.server.properties" override="false" value="file:C:/tts/
↪ application-config.properties" />
</Context>
```

7. Restart Tomcat. The Curator should be accessible via the context */curator* now.
8. Proceed with the WebAccess.

4.10 Post installation steps

4.10.1 Initialization of the database schema

If the steps described in the previous chapters have been carried out, the application should now be properly configured and deployed, thus being ready to start by starting Apache Tomcat.

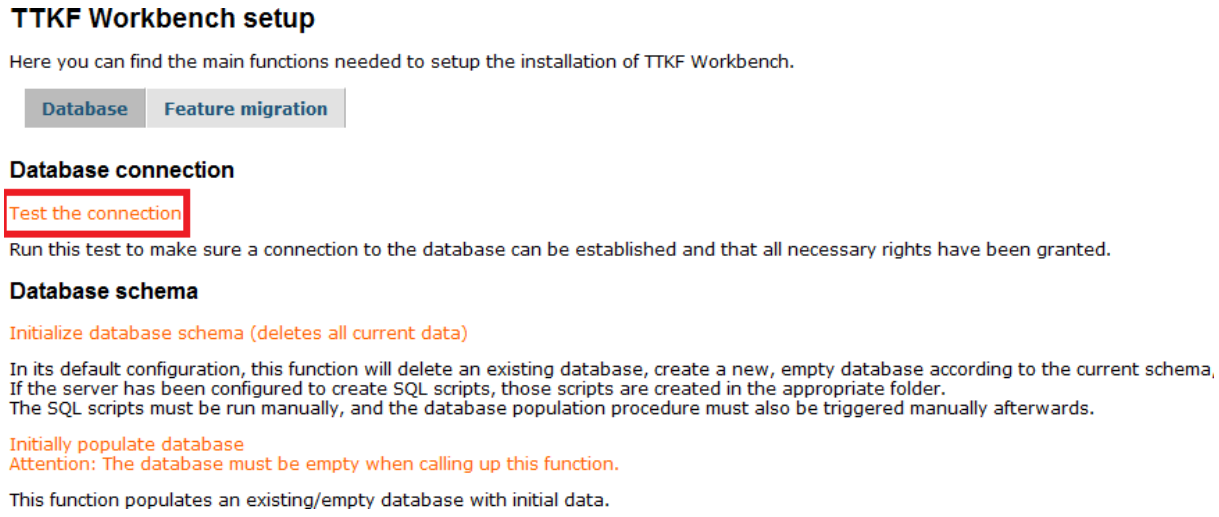
⚠ To initialize the database schema, a browser has to be started on the application server machine. If this is not possible and you thus have to use the browser on another computer, the IP address of that machine must be declared as superuser IP (property `superuserIPs` of Properties service).

✎ The webpage for initializing the database is located at the following URL:
http://[server]:[port]/[contextCurator]/site/install/install.do

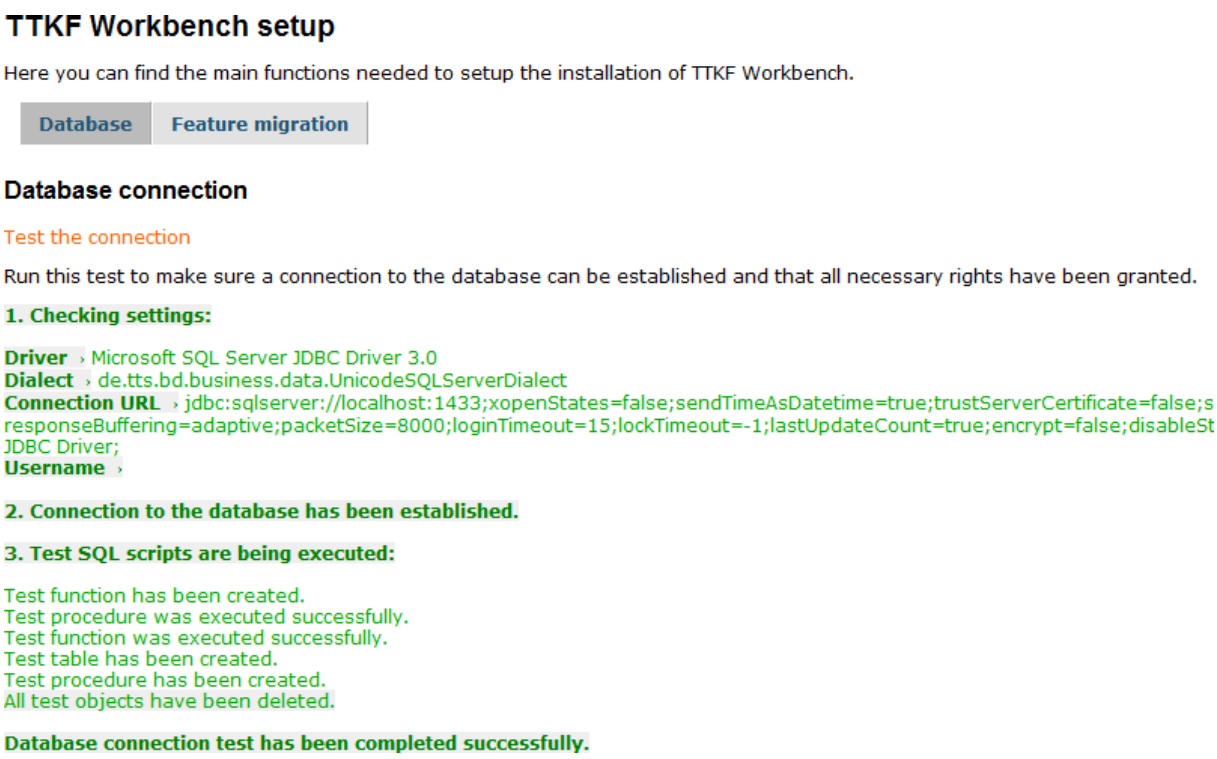
Start by entering the URL given above in the browser. A straightforward administration page appears, offering to test the database connection, (re-)create or update the database schema, or to import/export the entire database content.

4.10.1.1 Test database connection

To ensure that the database connection is working properly and that the database user has been granted all necessary permissions, the Curator setup provides a database connection test:

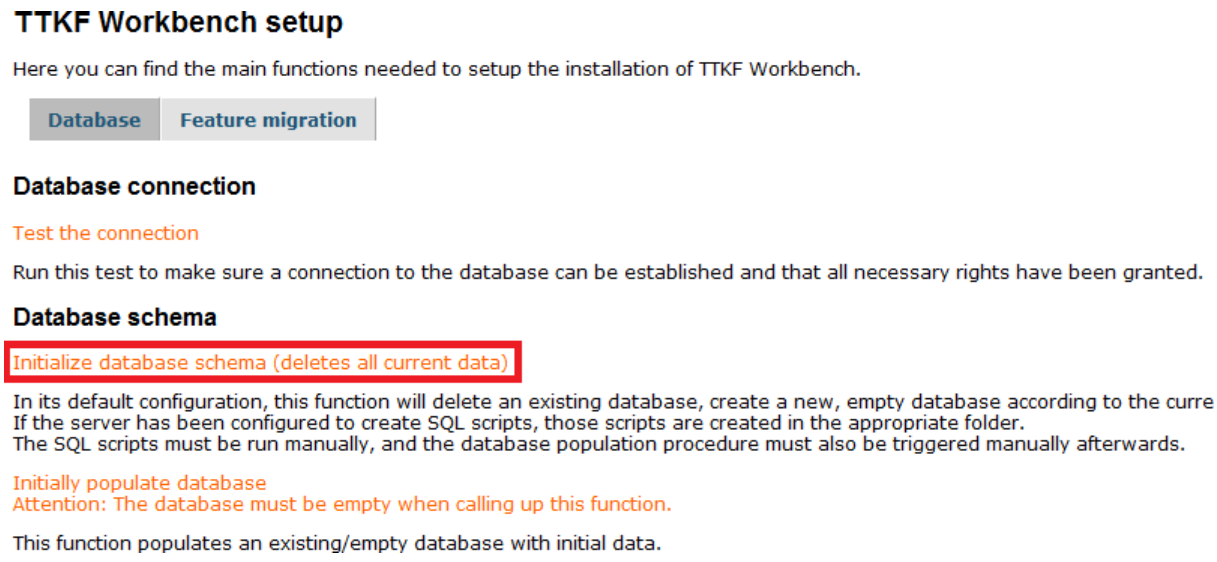


Shortly after clicking the Test the connection link, the result of the test will be displayed, representing successful tests in green, failed operations in red print.



4.10.1.2 Initialize database

If you click the *Initialize database* link, the Curator creates the database schema as well as initial data. After a few moments, the login screen will then appear. A full restart of the tts server will be necessary to ensure that all services are running properly.



When (re-)creating the database schema or importing database content, the previous data will be deleted completely. Please use this function very carefully!

When (re-)creating the database schema or importing database content, the previous data will be deleted completely. Please use this function very carefully

⚠ After initializing the database schema, the application should be restarted. Now the WebAccess may be started as well. For an easier analysis of possible problems, the log file of

the application server should be saved before restarting the server or applications.

4.10.2 First login as administrator

During database schema creation, a few initial data have already been created, like the administrator user:

Username: admin
Password: admin

The first login is done by calling *http://[server]:[port]/[contextCurator]/* in the browser and entering the administrator's username and password.

⚠ For security reasons, change the administrator password upon first login.

4.10.3 Next steps

The next steps for the administrator will usually encompass:

- Defining repositories and document types
- Assigning repositories to document types
- Adding users and maintenance roles
- Assigning authorizations to external users
- Defining a process model
- Managing custom properties
- ...

Please see the Administrator manual for more details on the steps outlined above.

4.11 Troubleshooting

The configuration and deployment of a complex application such as tts server sometimes results in errors. In this chapter, the most common issues and their solutions will be discussed.

Before you contact tts support, you are encouraged to have a look at this chapter, as an answer to the problem you are facing may be given here. In case the problem cannot be solved, tts support will be glad to help you.

Please note, however, that support activities are not covered by the maintenance fee if the problem results from a mistake the customer made during installation.

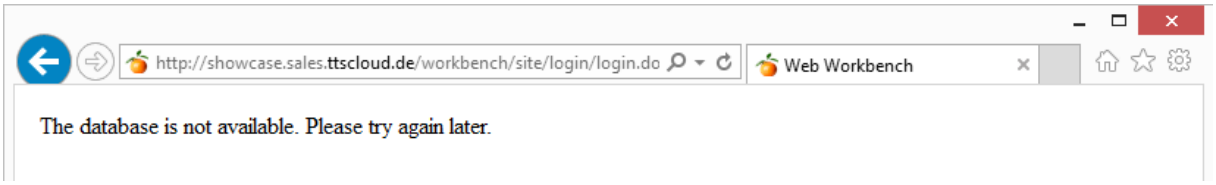
⚠ In order for tts support to be able to help you in the best possible way, please attach log files and configuration files of the tts server to your support request.

If the Curator is up and running, the log and configuration files can be downloaded from the Administration section (> Settings > Support > Server management). Otherwise, the log files of the application server should be attached instead.

4.11.1 The server does not start up

- Is the application server running correctly?
- Is the context path, if configured, pointing to the right directory (Apache Tomcat)?
- Is the license of tts server available and valid?
- Is the *application-config.properties* file available and configured properly?
- Is the data store defined in *application-config.properties* available?

4.11.2 Database is unavailable



- Is the database up and running?
- Can the database be accessed from the machine on which the application server is running?
- Has the port been set correctly?
- Is the specified data source available?
- Is the appropriate JDBC driver properly defined/stored on the application server?
- Is the data store defined in *application-config.properties* available?

4.11.3 Login fails

- Has the correct username and password been entered?
- Is the user already logged in?
- Has the maximum permissible number of logged-in users been reached?
- Has the license expired?
- Is this tts server addressing the right database (if pointing to a database initialized from another tts server, user login might fail)?

5 Migrating from previous versions

5.1 General remarks

This chapter provides a short summary of the most important changes that were introduced in the major releases, focusing on installation and update routines as well as migration scenarios.

5.2 Updating from 2025r2 to 2026

5.2.1 Solr 9.10.1 required

The minimum supported Solr version is 9.10.1. Please install Solr 9.10.1 and configure it according to the section on Solr core deployment. Then rebuild the index.

5.3 Updating from 2025 to 2025r2

5.3.1 Changes concerning Application Configuration parameters

5.3.1.1 Protect non-alphanumeric key parts with square brackets

Keys in the application configuration files consist of multiple key parts, separated by a dot. Historically, the application supports keys that contain non-alphanumeric characters. With release 2025r2, key parts containing such non-alphanumeric characters (except `-`) need to be surrounded by square brackets `[ ]` to ensure proper parsing.

For example,

```
1 ttkf.server.collectionContentHandler.customSupportedTypes.application/vnd.
↪ openxmlformats-officedocument.wordprocessingml.document=20
```

must be changed to

```
1 ttkf.server.collectionContentHandler.customSupportedTypes.[application/vnd.
↪ openxmlformats-officedocument.wordprocessingml.document]=20
```

5.3.1.2 Outdated keys

- Instead of `ttkf.server.properties.features.sios.enable`, please use `ttkf.server.feature.qa.sios`. Possible values are `true|false`, with default false.
- Please remove the key `ttkf.server.epss.mode`, as the class mode is no longer supported.

5.3.2 Migrate MinIO if necessary

⚠ All components of the tts Performance Suite now require a current MinIO version. If updating from MinIO version *2022-10-24T18-35-07Z*, a migration of the data is necessary.

For migration, please refer to the MinIO Migration Guide. You can use the mc mirror command to transfer data from an old MinIO server to a new one, or from a file system directory to a MinIO server.

5.3.3 Install and configure AuthServer

With release 2025r2 a new mandatory web application is introduced. The AuthServer provides authorization for internal communication. See chapter 4.8 Authorization Server (AuthServer) configuration for detailed information.

⚠ Without a proper deployed and configured AuthServer the ttsps Server is not working.

5.3.4 Install Solr 9.9.x

The minimum supported Solr version is 9.9.0. As the Solr configuration has changed, please install Solr 9.9 and configure it according to the section on Solr core deployment. Then rebuild the index.

5.4 Updating from 2024r2 to 2025

5.4.1 Java 21

The Curator and WebAccess of the tts Performance Suite 2025 require Java 21.

5.4.2 Increased ThreadStackSize for the JVM

The recommended stack size per JVM should be increased to 512K. The corresponding JVM argument is: `-Xss512k`.

5.4.3 Thred size ForkJoinPool

See chapter JVM arguments for more information about checking and setting the thread size properly according your setup.

5.4.4 Specify a prefix for the ttkf.server.properties property

If the path to the `application-config.properties` file is defined by the parameter `ttkf.server.properties` in the Tomcat context (or as an environment variable, or as a JVM argument), it is now recommended to prefix its value: either with `file:` if it points to a file on the file system, or with `classpath:` if it points to a file on the class path. Not providing a prefix still works, for now. The web applications will first try to load the `application-config.properties` from the file system and then from the class path, but a warning will be logged.

5.4.5 Migrate MinIO to the current version

⚠ The Curator and WebAccess applications of ttps 2025 are still compatible with MinIO version *2022-10-24T18-35-07Z*. However, the current Creator version already requires a current MinIO version, and the Curator and WebAccess of the 2025r2 release will also require it. Thus, we strongly recommend to install a current MinIO version.

For migration, please refer to the MinIO Migration Guide. You can use the mc mirror command to transfer data from an old MinIO server to a new one, or from a file system directory to a MinIO server.

5.4.6 Proxy or load balancer: Additional websocket endpoint

If the tts Performance Suite is running behind a proxy or a load balancer, please make sure to add the new websocket endpoint `/ui/subscription` for the Curator.

```
<!-- Example snippet from an Apache reverse proxy, please adapt to your scenario:
-->
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteRule /curator/ui/subscription/(.*) ws://<host>:<port>/curator/ui/
subscription/$1 [P,L]
```

5.4.7 Login modules can only be configured for a WebAccess with required login

If the WebAccess is configured with any login modules, login must be configured as required.

This concerns the application configuration parameters:

`ttkf.server.user.loginModules` or `ttkf.accelerator.user.loginModules`

`ttkf.server.properties.portalLoginRequired`

If login is configured as optional with `ttkf.server.properties.portalLoginRequired=false`, the WebAccess will abandon startup and log an error with an `IllegalConfigurationException`. To fix this, either remove the login modules or enable authentication in the WebAccess.

5.4.8 Feature flag for cycle display type

To enable the cycle display type for processes, configure the following feature flag in the `application-config.properties`:

```
ttkf.server.feature.process.displaytype.cycle = true
```

The cycle display type is only available for processes, and only in configurations based on initial_2024.

5.4.9 Feature flags for the Curator Edit View

The Edit View feature in the Curator is now enabled by default. It can still be disabled with:

```
ttkf.server.feature.development.edit.view = disabled
```

An additional feature flag controls whether to use the new preview feature or stick with the previous inline representation of the document content:

```
ttkf.server.feature.development.edit.view.preview = enabled|disabled
(default: enabled)
```

5.5 Updating from 2024 to 2024r2

5.5.1 Jakarta EE 10 and Tomcat 10.1

The tts performance suite 2024R2 moved up to Jakarta EE 10 and requires now Servlet Container Tomcat 10.1. Due to the implementation of Servlet API 6.0 and JSP API 3.0, running on Tomcat 9 is not possible anymore!

⚠ The Creator still uses Jakarta EE 8 and must be deployed either on a separate Tomcat instance or in the legacyWebApps folder of Tomcat 10.1. See the Creator manual for detailed information

5.5.2 Solr 9.6.0 required

The minimum supported Solr version is 9.6.0. An Update from 9.5 to 9.6 will not require configuration changes.

5.5.3 SAP LSO Webservice removed

The SAP LSO webservice has reached end-of-life and has been removed in 2024R2.

5.5.4 LegacyCookieProcessor

The LegacyCookieProcessor has been deprecated in Tomcat 9 and removed in Tomcat 10. Therefore, this configuration entry must be adjusted by removing `className` attribute in `CookieProcessor-Tag` in `server.xml` or in corresponding context files (e.g. `curator.xml`). Additionally, the curator now supports cookies with `SameSite=Lax`.

```
<!-- Tomcat 9 -->
<CookieProcessor
className="org.apache.tomcat.util.http.LegacyCookieProcessor"
sameSiteCookies="None" />
<!-- Tomcat 10.1 -->
<CookieProcessor sameSiteCookies="Lax" />
```

5.5.5 Hibernate Dialects

Internal Hibernate 6 update required some type mapping changes. Therefore, some new Hibernate Dialects were introduced for tts performance suite Server 2024R2.

```
# Oracle
# Unicode dialect should be preferred
ttkf.server.data.hibernate.dialect = com.tts.data.dialect.
↪ LegacyUnicodeOracleDialect
ttkf.accelerator.data.portal.hibernate.dialect = com.tts.data.dialect.
↪ LegacyUnicodeOracleDialect

# Non-unicode dialect
ttkf.server.data.hibernate.dialect = com.tts.data.dialect.LegacyOracleDialect
ttkf.accelerator.data.portal.hibernate.dialect = com.tts.data.dialect.
↪ LegacyOracleDialect

# MS SQL Server
# Unicode dialect should be preferred
ttkf.server.data.hibernate.dialect = com.tts.data.dialect.
↪ LegacyUnicodeSQLServerDialect
ttkf.accelerator.data.portal.hibernate.dialect = com.tts.data.dialect.
↪ LegacyUnicodeSQLServerDialect
# Non-unicode dialect
ttkf.server.data.hibernate.dialect = com.tts.data.dialect.LegacySQLServerDialect
ttkf.accelerator.data.portal.hibernate.dialect = com.tts.data.dialect.
↪ LegacySQLServerDialect
```

5.5.6 New QuickAccess Design

A new design iteration was added for the QuickAccess and is available in the "config/initial_2024" folder. To activate it rename the config file `templates.initial_2024_redesign.xml.example` to templates.initial_2024_redesign.xml and restart the server.

5.5.7 Creator opens in modal dialog

By default, the Creator opens in a modal dialog inside the Curator now. To switch back to the old behavior (new browser tab), activate the feature flag `ttkf.server.feature.creator.inNewTab`.

5.6 Updating from 2023r2 to 2024

5.6.1 Solr 9.5.0 required

⚠ Please delete the old index files before updating. The Server will rebuild the index after starting.

The tts performance suite Server 2024 requires Solr 9.5.0.

In addition, the structure of the `solr.zip` artifact has changed and now contains an optional solr.xml file. Please copy the tts-server folder from the `solr.zip` as normal. The `solr.xml` file may be required when configuring the maximum Boolean clauses (described below).

5.6.1.1 Configure the max Boolean clauses

The maximum Boolean clauses allowed in a search is per default 1024. This must be set to 4096 or higher:

- Either pass as a JVM argument: `-Dsolr.max.boooleanClauses=4096`
- Or define an environment variable: `SOLR_MAX_BOOLEAN_CLAUSES=4096`
- Or define it in the solr.xml:
  - If you have an existing solr.xml, configure it there
  - If not, feel free to use the file `solr.xml` in the artifact `solr.zip`.

5.6.2 Removal of legacy SAML

With the performance suite 2024, the legacy SAML implementation has been removed. Therefore, the login module `SAMLAuthenticationLoginModule` is no longer supported. Please use the new SAML login module instead: `com.tts.authentication.saml.login.SamlLoginModule`

💡 Please see the SAML section for configuration details or refer to your professional services consultant for assistance.

Removal of Guidebase and Quick Guides

As Quick Guides have reached their end of life, they have been completely removed in the 2024. Creating, editing, searching and displaying Quick Guides is no longer possible. There are three things to consider:

1. Please remove the guide repository manually when updating to the 2024.
2. Please remove the process queue before starting the 2024 for the first time.
3. Please remove the following obsolete configuration parameters:

- `ttkf.server.guidebase.rootdir`
- `ttkf.server.guide.repository.url`
- `ttkf.server.guidebase.paging.size`

5.7 Updating from 2023 to 2023r2

5.7.1 New Configuration Keys Available for Caching Newest CRE

The Server (both the Curator and the WebAccess) caches the latest CRE identifier for each configuration. New configuration parameters are available to configure the size of this cache and the expiry time:

```
ttkf.server.cache.newestCreCache.size = 32
ttkf.server.cache.newestCreCache.expires = 600
```

The parameters are optional and the default values are as given above.

5.7.2 Windows SSO – alternative SecurityFilterProvider available

An alternative SecurityFilterProvider has been provided when using Windows SSO to log in to the QuickAccess through a reverse proxy or load balancer:

```
ttkf.server.user.login.sso.windows.securityFilterProviders=ttn.bd.login.sso.
↪ windows.authentication.SessionBasedConnectionIdNegotiateSecurityFilterProvider
```

This alternative should only be required if the QuickAccess prompts the user for their credentials; it may be that the reverse proxy is changing the ports used to connect to the backend server.

5.8 Updating from 2022r2 to 2023

5.8.1 Business Guidance Creator

Starting with the 2023 release, the Business Guidance Creator must be deployed and configured. The following properties are therefore required:

```
ttkf.server.businessguidancecreator.endpoint=https://creator-url
ttkf.server.businessguidancecreator.presignedurl.publickey.endpoint=https://
↪ creator-url/v1/presigned-url/certificate
```

For more details on setting up the Business Guidance Creator please see the Creator Installation Manual.

5.8.2 Search Index API

A new endpoint to trigger the recreation of Solr search index has been introduced. See the Search Service section in the appendix for more details.

5.8.3 Rate Limiting for Push Notification download activated by default

The rate limiting for downloading push notifications is activated by default. If you wish to deactivate this behavior, the configuration parameter must be set to false explicitly.

```
ttkf.server.pushnotification.rate.limit.enabled = false
```

5.9 Updating from 2022 to 2022 R2

5.9.1 Allowed URI schemes for URL documents

Starting with the 2022R2 release, URL documents must be valid URIs. Also, the allowed URI schemes can be configured in the `application-config.properties`. The properties accept a comma separated list of values.

Example:

```
ttkf.server.url.documents.disallowedSchemes=http, javascript
ttkf.server.url.documents.allowedSchemes=quickaccess, https
```

A URL document can be saved if and only if the scheme is not in the set of `disallowedSchemes` and is in the set of `allowedSchemes`.

It is also possible to (dis)allow all schemes by setting the corresponding property to the wildcard (`*`) or to (dis)allow no schemes by leaving the property value empty.

Example:

```
ttkf.server.url.documents.disallowedSchemes=
ttkf.server.url.documents.allowedSchemes=*
```

By default, the `disallowedSchemes` are set to `javascript` and the `allowedSchemes` are set to the wildcard `*`. This results in all schemes enabled except the `javascript`-scheme

Defaults:

```
ttkf.server.url.documents.disallowedSchemes=javascript
ttkf.server.url.documents.allowedSchemes=*
```

5.9.2 Redis Cache

To simplify the internal complexity, the former Redis cache provider implementations have been merged into one `RedisCacheProvider`. The parameter `ttkf.server.cache.redis.cluster` now defines, if Redis is running in cluster or standalone mode.

```
ttkf.server.cache.cacheProvider=com.tts.serverfoundation.cache.redis.
↪ RedisCacheProvider
ttkf.server.cache.redis.cluster=true
```

5.9.2.1 Redis Client Connection Pool

With 2022R2 release, both Redis cache implementations, stand-alone and cluster, offer a possibility to configure their client connection pool:

```
#defaults
ttkf.server.cache.redis.pool.maxTotal=16
ttkf.server.cache.redis.pool.maxIdle=16
ttkf.server.cache.redis.pool.minIdle=2

# configure as your scenario pleases
ttkf.server.cache.redis.pool.maxTotal=20
ttkf.server.cache.redis.pool.maxIdle=20
ttkf.server.cache.redis.pool.minIdle=5
```

5.10 Updating to 2022

See the installation manual for 2025R2 on how to update to 2022.

6 Appendix

6.1 Properties service

Curator & WebAccess

Application-wide settings are defined in the properties service.

acceleratorContextPath (Curator only)

Defines the site relative URL to the WebAccesss. In case several WebAccesses are installed, the one used for preview should be given here.

Possible value: `/webaccess/`

integratorContextPath (WebAccess only)

Defines the site relative URL to the Curator.

Possible value: `/curator/`

superuserIPs

A comma-separated list of IP addresses for which super user functions, like database initialization, are permitted. Do make sure that this property is not left blank (read: you should at least enter `127.0.0.1` here.)

Possible value: `127.0.0.1`

allowLoginCookie

Allows remembering login data by means of a browser cookie.

Possible values:

- `yes` (default)
- no

automanageDocumentLanguage

Automatically determines in which language multilingual meta data are saved (by using content language).

Possible values:

- `yes` (default)
- `no`

xmlReaderClass

Defines an XML reader if the application server does not supply one.
(This property is removed since 2021 R2)

Possible value: Fully qualified Java class name.

Example: `org.apache.xerces.parsers.SAXParser`

applicationServerLogDirectory

Defines a directory path for application server log files. If set, the administrative function download log files include the log files from the server.

Possible value: Any valid path on a file system.

Example: `c:/myAppServerDir/logs`

checkedOutLicenseValidity

Sets the number of days a checked-out license remains valid.

Possible value: Any positive integer value, e.g., 14.

httpCacheValidation

Defines the method of HTTP cache validation.

Possible values:

- `ETag`
- `LastModified` (default)

httpCacheExpiration

Defines the method of HTTP cache expiration.

Possible values:

- `Expires`
- `CacheControl` (default)

httpCacheExpirationAge

Defines the age of HTTP cache expiration sent by the server in seconds.

Example: `31556926` (1 year) (default)

showStatifyCertificateButton

If set to `true`, the option “Show Certificate Button” within the stratification is enabled.

Possible values:

- `true` (default)
- `false`

qaIndexLength

Sets the maximum length of an index entry in the QuickAccess signature.

Possible value:

- Any positive integer value, e.g., 64.
- `0` = no restriction

qaWildCardRestriction

Restricts level-wildcard assignments (generic signature).
`1` = level is required and cannot be used as a wildcard.
`0` = level can be assigned as a wildcard by the user.

Example: `GEN:1001;URL:1001;SAP:10001`

serverUrl

Represents the URL and context path of the application used to backtrack imported objects to their origin.

Possible value: Any valid URL.

Example: `http://sampleServer:8080/curator`

external_integratorURL

Represents the URL to access the Curator from external locations, for example in a reverse proxy scenario. External URL and the address for login usually are the same.

Therefore, the URL should be equal concerning case sensitivity, otherwise firewalls or proxies might block these URLs. If not specified the external URL of the Curator is mapped to the internal URL.

Possible value: Any valid URL.

Example: `http://externalserver/curator`

Attention: This parameter is required for creating user generated content using the Creator.

external_acceleratorURL

Represents the URL to access the WebAccess from external locations, for example in a reverse proxy scenario. External URL and the address for login usually are the same.

Therefore, the URL should be equal concerning case sensitivity, otherwise firewalls or proxies might block these URLs. If not specified the external URL of the WebAccess is mapped to the internal URL.

Possible value: Any valid URL.

Example: `http://externalserver/webaccess`

internal_integratorURL

Represents the URL to access the Curator in an internal way, mainly used for internal server communication between the Curator and WebAccess. This property is mandatory!

Possible value: Any valid URL.

Example: `http://internalserver:8080/curator`

internal_acceleratorURL

Represents the URL to access the WebAccess internally, mainly for internal server communication between the Curator and WebAccess. This property is mandatory

Possible value: Any valid URL.

Example: `http://internalserver:8080/webaccess`

portalLoginRequired (WebAccess only)

Defines whether WebAccess requires login or not.

Possible values:

- `true` (default)
- `false`

portalIgnoresMaintenanceRoles (WebAccess only)

Defines whether the maintenance roles are ignored or not when it comes to filtering objects visible to a user.

Possible values:

- `true`
- `false` (default)

portalIgnoresMaintenanceRolesOnPreview (WebAccess only)

Defines whether the maintenance roles are ignored on preview or not when it comes to filtering objects visible to a user.

Possible values:

- `true`
- `false` (default)

qaSearchMode (WebAccess only)

Defines the QuickAccess search mode.

Possible values:

- `direct` searches only documents of the context specified by the QuickAccess application.
- `cascade` " uses multiple searches to allow more general results. E.g., if the specific context is "GEN;excel;cell formatting", the cascade mode will also search for the context "GEN;excel" if no documents were found for the specified context.
- `all` uses multiple searches defined by searchlists "qaDirectSearchList" and "qaIndirectSearchList" (default).
- `off` disables document search.

qaGlossarySearchMode (WebAccess only)

Enables QuickAccess to search within glossary entries.

Possible values:

- on (default)
- `off`

qaDirectSearchList (WebAccess only)

Sets a comma separated list of signature-level maps for a direct result list.

Example: `GEN:1110;URL:1110;SAP:11000`

"1110" means the first three levels of a document's signature must be equal to the signature currently searched for.

qaIndirectSearchList (WebAccess only)

A comma-separated list of signature-level maps for an indirect result list.

Example: `GEN:1100;URL:1100;SAP:00000`

qaSortResult (WebAccess only)

Sets sorting of the result list by explorative context.

Possible values:

- `yes` (default)
- no

qaSortAdjacentResult (WebAccess only)

Sets sorting of the adjacent result list by explorative content.

Possible values:

- `yes`
- no (default)

qaFilterChain (WebAccess only)

Sets the filter chain which filters the documents returned by the QuickAccess. If none is specified, the default filter chain is used.

Example: qaFilterChain

accessLatestReleased (WebAccess only)

Defines whether the latest released document version is displayed (`true`), or the latest version of a document (`false`) (without workflow engine).

Possible values:

- `true`
- `false` (default)

image.format.${domain}.${name}

Defines an image format by providing the domain and name of the format. Currently, provided domains are userprofile and newsfeedentry. The value specifies width first, and height last. If given both values, the image is scaled along the greater side; if given one value, the other one must be marked with "~".

Example:

```
image.format.userprofile.passport = width:50;height:150
image.format.newsfeedentry.default = width:500;height:~
```

image.upload.${domain}.maxsize

Defines the max size an image for the providing domain may have to get uploaded successfully. Currently, the only supported domain for this parameter is userprofile. The value is given as a decimal number directly followed by the unit kb or mb.

Default value: 1.6mb

Example:

```
image.upload.userprofile.maxsize = 128kb
image.upload.userprofile.maxsize = 1.44mb
```

image.formats

Defines the list of productive image formats as comma-separated list.

Example:

```
image.formats = userprofile.passport,newsfeedentry.resource
```

portalLinkBaseURL

Defines the base URL which shall be used for the portal link. This parameter is optional.

Example: `http://server:port/context`

ttkf.server.properties.reject.user.agents.patterns

This parameter is optional.

In chromium-based browsers deeplinks are opened in two tabs when clicked within Microsoft Office products, such as Word or Excel. Reason for this behavior is an internal request with an internal http Producer to check the link itself. With the following parameter it is possible, to reject requests made by an user agent containing the given comma-separated patterns. The strings are case-insensitive.

Example:

```
ttkf.server.properties.reject.user.agents.patterns = microsoft,ms-office
```

pushNotificationLanguageFallbackChain

Defines the language fallback chain for Push Notifications. Comma separated list of language tags. This parameter is optional. If not defined the default value is used.

Default: `de-de,en-us,fr-fr,es-es`

Example:

```
ttkf.server.properties.pushNotificationLanguageFallbackChain=de-de,en-us,fr-fr,es-
↪ es
```

ttkf.server.contentConfig.deactivated

By setting the parameter `ttkf.server.contentConfig.deactivated` to a comma separated list of configuration names, these configurations can be deactivated. They cannot be selected for

creating new documents anymore. However, playing content created with deactivated configs is still possible as well as editing and republishing these documents.

Example:

```
1 ttkf.server.contentConfig.deactivated=initial,my_custom_config,another_config
```

6.2 Data service

For tts server to be able to store data persistently, a connection to the previously created database is necessary. The Hibernate ORM framework is used to get object-oriented access to the relational database. Therefore, all defined properties are delegated to Hibernate and the connection pooling framework.

6.2.1 JNDI data source

We strongly recommend you configure the database connection by creating the data source in the application server. A data source is a logical database interface, providing (and hiding complex) information with regard to the database connection and connection pooling. Each data source can be accessed through a JNDI name.

Database configuration

If a database connection is provided through a JNDI data source, the JNDI name, the Hibernate dialect, and the name of the database schema have to be specified. Additional information, like the JDBC connection URL and the database user, are provided by the JNDI data source.

Service name: data (maintenanceInterval: 30)

hibernate.dialect

Defines the fully qualified Java class of the database dialect (type of database and used character set).

Possible values: For a complete list, please see the supported Hibernate dialects below.

Example: `de.tts.bd.business.data.UnicodeOracle12cDialect`

hibernate.default_schema

Sets the name of the default schema to use for SQL statements (usually the name of the database user).

Example: `TTS`

portal.hibernate.dialect

Defines the fully qualified Java class of the database dialect (type of database and used character set) for tts Server WebAccess.

Possible values: For a complete list, please see the supported Hibernate dialects below.

Example: `de.tts.bd.business.data.UnicodeOracle12cDialect`

portal.hibernate.default_schema

Sets the name of the default schema to use for SQL statements (usually the name of the database user) for TTS Server WebAccess.

Example: `TTS`

For Oracle, the default schema should always be specified in upper case.

The necessary parameter can be set via the *application-config.properties* file, for example:

```
1 # curator database configuration
2 ttkf.server.data.hibernate.dialect = de.tts.bd.business.data.
↪ UnicodeOracle12cDialect
3 ttkf.server.data.hibernate.default_schema = TTS
```

The Hibernate dialect ensures that the suitable syntax is used for initializing, updating, and accessing the database. Any later change of the dialect (for example from non-Unicode to Unicode) may cause problems.

Supported Hibernate Dialects:

- `de.tts.bd.business.data.UnicodeSQLServerDialect`
- `de.tts.bd.business.data.UnicodeOracle12cDialect` (suits 18c as well)
- `de.tts.bd.business.data.Oracle12cDialect`
- `de.tts.bd.business.data.SQLServerDialect`

The JDBC driver for Microsoft SQL Server requires the property select Method to be set to the value cursor. If there are performance issues, the value could be changed to direct.

6.3 Store service

The store service manages the runtime files used by the tts performance suite Server. The only property that needs to be set is the base path where the files will be stored. Within this directory, all sub-directories that are necessary for the single stores (config, document spool, etc.) will be created automatically.

In case you want to have separate stores (for spool, image, config), you can define them independent from the base path.

Temporary stores, as used by the report and import/export engine, will be located in a sub-directory of the data store base path. These stores cannot be defined separately from the data store.

Please make sure the data store provides enough disk space for storing temporary runtime files. See the requirements section for details.

For stores which are located under Minlo, s3 url are needed. For stores on a file system, a file url has to be provided.

Service name: store

store.base.directory

The base directory of the data store.

Example: `/home/ttn/datastore`

store.spool.url

Defines the spool directory where document contents are stored temporarily before being uploaded to the repository.

Examples:

- `file:///home/ttn/spool`
- `s3://bucket/ttn/spool`

store.config.url (optional)

Defines the directory where the version-controlled configurations of the tts performance suite Server Producer are stored.

Examples:

- `file:///home/ttn/config`
- `s3://bucket/ttn/config`

store.index.directory (optional, deprecated)

Defines the path where the search engine stores its index files.

Example: This is no longer used. Please Remove.

store.image.url (optional)

Defines the directory where images uploaded through the Curator are stored.

- `file:///home/ttn/image`
- `s3://bucket/ttn/image`

store.cre.url (optional)

Defines the directory where content runtime environments for the tts performance suite Desktop Producer is stored by the Server.

Examples:

- `file:///home/ttn/cre`
- `s3://bucket/ttn/cre`

store.report.url (optional)

Defines the directory where report generated by the tts performance suite Server are stored for download.

Example: `file:///home/ttn/report`

The report store cannot be an S3 path

6.4 User service for authentication and authorization

Service name: user

burst.retry.size

Sets the maximum permissible number of failed login attempts by a user before the system locks the user name for a specified time.

Possible value: Any integer value, e.g., 2

burst.denial.time

Defines the time (in seconds) a username for which the maximum number of failed login attempts was exceeded will be locked from the system.

Possible value: Any integer value, e.g., `30`

6.4.1 Login modules

Login modules are responsible to authenticate a user and provide user-specific information (user profile) to the application. They are categorized primarily by the point in time they are active during the authentication process. Modules which allow for an *explicit authentication* require a username and a password as their input. They are activated as soon as these bits of information have been provided. Modules that support *implicit authentication* derive their information from the first HTTP request to the server. These modules are primarily used for Single-Sign-On.

loginModules

A comma-separated list of fully qualified Java class names of login modules.

Possible values: One of the Login modules listed below.

Example: `de.tts.bd.business.login.ldap.LdapLoginModule`

When using the CookieLoginModule in combination with external users the following parameter has to be configured. Otherwise, the user object cannot be provided.

It is possible to support multiple login systems in the backend with cookie login. Mixed scenarios like database and external authentication are fully featured via cookie authentication. This is done by configuring multiple authorities populators in a comma-separated list. Per default the database authorities populator is always appended to the list of populators.

loginModules.CookieLoginModule.populator

Populators which are used to provide the user object.

Possible values:

- `de.tts.bd.business.login.ldap.ExtendedLdapAuthoritiesPopulator` (deprecated)
- `de.tts.bd.business.login.request.RequestElementsAuthoritiesPopulator`
- `ttn.bd.login.sso.windows.authorization.WindowsSSOAuthoritiesPopulator`

List of Login Modules

Module	Authentication	Remarks
`de.tts.bd.business.login.DefaultLoginModule` (explicit)	Authentication with user name + password against data stored in a local database.	
`de.tts.bd.business.login.CookieLoginModule` (implicit)	Login via user name and login-token contained in an HTTP cookie.	

Module	Authentication	Remarks
`com.tts.serverfoundation.authentication.ldap.LdapLoginModule` (implicit)	Authenticates against an LDAP directory.	When User with domain 'ldap' exists in database this user is logged in. Otherwise, the user is created at runtime by obtaining the profile from information stored in an LDAP record on the directory server.
`de.tts.bd.business.login.ldap.SimpleLdapLoginModule` (explicit) (deprecated)	Authenticates against an LDAP directory.	The user must also exist in the local database. Only the password may be omitted.
`de.tts.bd.business.login.ldap.ExtendedLdapLoginModule` (explicit) (deprecated)	Authenticates against an LDAP directory.	The user is created at runtime by obtaining the profile from information stored in an LDAP record on the directory server.
`de.tts.bd.business.login.request.RequestElementsLoginModule` (implicit)	Authenticates against an authentication value stored in the HTTP header or as a URL parameter.	The user is created at runtime by obtaining the profile from information stored in the HTTP request.
`ttn.bd.login.sso.windows.WindowsSSOLoginModule` (implicit)	Authenticates and authorizes against LDAP with Windows logon credentials (default). Since 2012 R2, it is possible to configure a different authorities populator.	The user is created at runtime by obtaining the profile from information stored in Windows principal object and LDAP record. Since 2012 R2, the user information can be retrieved from the local data- base as well using users with domain 'ldap'.
`com.tts.authentication.saml.SAMLLoginModule`	This login module is designed for Single-Sign-On based on SAML v2. The user authenticates against the configured identity provider.	When User with domain 'saml' exists in database this user is logged in. Otherwise, the user is created at runtime by obtaining the profile from information stored in the identity provider.

⚠ **Login modules may be combined to form module chains. As soon as one of the modules successfully authenticates the user, the user will be logged in.**

6.4.2 LDAP authentification

Supported scenarios

For LDAP-related login modules three scenarios are supported.

(A) Local user + LDAP authentication + without service user (deprecated)

This scenario is provided by the *SimpleLdapLoginModule*. It is recommended for directories with a flat structure combined with local user administration.

1. Load a user from the local database
2. Retrieve LDAP authentication information from the user object to find the user's Distinguished Name (DN) and LDAP server.
3. In case no LDAP authentication information is available, the DN is created through a template.
4. Bind the determined DN to the LDAP directory for authentication.

(B) Local user + LDAP authentication + with service user

This scenario is provided by the *SimpleLdapLoginModule*. It is recommended for directories with a complex structure combined with local user administration.

1. Load a user from local database (user needs 'ldap' in domain column, achieveable with user import)
2. Bind the service user to LDAP directory.
3. Ascertain the DN of the user's LDAP entry through a search operation.
4. Bind the determined DN to the LDAP directory for authentication

(C) Runtime user + LDAP authentication + with service user

This scenario is provided by the *LdapLoginModule*. It is recommended for directories with structures of any complexity and requires no local user management.

1. Ascertain the DN of the user's LDAP entry through a search operation.
2. Create a runtime user profile by using the data stored in the LDAP entry.
3. Bind the determined DN to the LDAP directory for authentication.

Properties for LDAP configuration only

ldap.default.base.dn

Sets the base DN used for all LDAP operations in TTS PERFORMANCE SUITE (root DN).

Example: `DC=teamtraining,DC=local`

ldap.default.manager.dn (scenarios B,C)

Sets the DN of the service user used for all LDAP search operations.

Example: `CN=ldapadmin,OU=Services,OU=TTS,DC=teamtraining,DC=local`

In case the DN contains a backslash "", this character must be escaped with" ", otherwise the value of this parameter might be parsed wrong.

Example: `CN=ldapadmin, \\ admin,OU=Services,OU=TTS,DC=teamtraining,DC=local`

ldap.default.manager.password (scenarios B,C)

Sets the password of the service user. In case the password contains a backslash "", this character must be escaped with "", otherwise the value of this parameter might be parsed wrong. Best way would be to avoid a backslash in the password and to use another special character.

ldap.default.manager.password.x (scenario B,C)

Sets the password of the service user. Other than `ldap.default.manager.password`, where the password is needed in cleartext, this parameter uses the BASE64 encoded version of the used password. If both parameters are set the encrypted version is used.

ldap.default.dialect

Fully qualified Java class name of dialect for LDAP server.

Possible values:

- `de.tts.bd.business.login.ldap.GenericLdapDialect` (default dialect for all LDAP services)
- `de.tts.bd.business.login.ldap.ActiveDirectoryDialect` (Microsoft Active Directory Service)

ldap.default.user.dn.template (SimpleLdapLoginModule)

Defines the template for DN's of user entries. The placeholder ${ttc:user} may be used instead of a username. During the login process, the placeholder is replaced by the actual username.

Possible value: `CN=${ttc:user},OU=User,OU=TTS, DC=teamtraining,DC=local`

ldap.default.mapper.userProfile *(ExtendedLdapLoginModule)*

Fully qualified name of Java class for mapping LDAP attributes to values of the user profile.

Possible value: `de.tts.bd.business.login.ldap.GenericLdapUserProfileMapper`

ldap.default.mapper.processRoles *(ExtendedLdapLoginModule)*

Fully qualified name of Java class for mapping LDAP groups of a user to process roles of the TTS PERFORMANCE SUITE server.

Possible value: `de.tts.bd.business.login.ldap.GenericLdapProcessRoleMapper`

ldap.default.mapper.maintenanceRoles *(ExtendedLdapLoginModule)*

Fully qualified name of Java class for mapping LDAP groups of a user to maintenance roles of the TTS PERFORMANCE SUITE server.

Possible value: `de.tts.bd.business.login.ldap.GenericLdapMaintenanceRoleMapper`

ldap.default.recurseIntoGroups

Should parent groups be considered?

Possible values:

- `true`
- `false` (default)

Example:

```
1 ttkf.server.user.ldap.default.recurseIntoGroups=true
```

ldap.default.group.base.dn

Base DN for group search.

Default value: Value of parameter ttkf.server.user.ldap.default.base.dn

Example:

```
1 ttkf.server.user.ldap.default.group.base.dn = DC=teamtraining,DC=local
```

ldap.default.dialect.group.filter

Filter query for group search. Template parameters can be used.

Default value: `(&(objectClass=group)(dn=${ttc:group}))`

Example:

```
ttkf.server.user.ldap.default.dialect.group.filter=(&(objectClass=group)(dn=${ttc:
group}))
```

ldap.default.dialect.group.attr.groupMembership

Name of the group membership attribute.

Default value: `memberOf`

Example:

```
ttkf.server.user.ldap.default.dialect.group.attr.groupMembership=memberOf
```

⚠ You can manage the mapping of LDAP groups to TTS PERFORMANCE SUITE roles (either process or author roles) in the *Assign authorizations to external users* dialog located in the administration section (>*Settings* >*Users*).

LDAP dialects

LDAP dialects are used to simplify the mapping of LDAP entries to values of the user profile. Since most LDAP directories are structured in a very particular way, a specific dialect may not be needed in many cases. At the moment, two dialect classes are supported. The *GenericLdapDialect* provides a set of mappings which may be common to a lot of directories and which should be sufficient in most cases. The *ActiveServerDialect* can be used in a Windows Server environment.

All properties must be prefixed with `ldap.default.dialect`, which is omitted in the following list for improved readability.

.user.filter

Defines the LDAP search filter used to find user entries (e.g. `(&(objectClass=user)(cn=${ttc:user}))`).

Default values:

- GenericLdapDialect: `(&(objectClass=user)(cn=${ttc:user}))`
- ActiveDirectoryDialect: `(&(objectClass=user)(userPrincipalName=${ttc:user}@${ttc:adsDomain}))`

.user.attr.name

Sets the name of the LDAP attribute containing the username.

Default value: `cn`

.user.attr.uniqueId

Sets the name of the LDAP attribute containing a unique identifier.

Default values:

- GenericLdapDialect: `distinguishedName`
- ActiveDirectoryDialect: `objectGUID`

.user.attr.uniqueId.encoding

Sets the encoding of the unique id (allowed here are “string” or “uuid_bin”, the latter denotes a binary UUID format).

Default values:

- GenericLdapDialect: `string`
- ActiveDirectoryDialect: `uuid_bin`

.user.attr.firstname

Defines the name of the LDAP attribute containing the first name.

Default value: `givenName`

.user.attr.lastname

Defines the name of the LDAP attribute containing the last name.

Default value: `sn`

.user.attr.email

Defines the name of the LDAP attribute containing the email address.

Default value: `mail`

.user.attr.language

Defines the name of the LDAP attribute containing the language.

Default values:

- GenericLdapDialect: `preferredLanguage`
- ActiveDirectoryDialect: `c`

.user.attr.language.encoding

Sets the encoding of the language (allowed here are “iso639”, “iso3166”, “rfc1766”, the value is converted automatically).

Default values:

- GenericLdapDialect: `iso639`
- ActiveDirectoryDialect: `iso3166`

.user.attr.editLanguage

Defines the name of the LDAP attribute containing the edit language (if language is not provided).

Default value: `preferredLanguage`

.user.attr.editLanguage.encoding

Sets the encoding of the edit language (allowed here are “iso639”, “iso3166”, “rfc1766”, the value is converted automatically).

Default value: `iso639`

.user.attr.interfaceLanguage

Defines the name of the LDAP attribute containing the interface language (if language is not provided).

Default value: `preferredLanguage`

.user.attr.interfaceLanguage.encoding

Sets the encoding of the interface language (allowed here are "iso639", "iso3166", "rfc1766", the value is converted automatically).

Default value: `iso639`

.user.attr.groupMembership

Sets the name of the LDAP attribute containing the group membership information (must contain distinguished names of the groups). This parameter offers a combination of LDAP-attributes, delimited by "#".

Default: `memberOf`

Example (with combined attributes): `#c#l results into "c=DE,l=Heidelberg"`

.adsDomain (ActiveDirectoryDialect only)

Sets the ADS domain name (e.g., teamtraining.local). If it is not specified, the search includes all domains. Only necessary if...`user.filter` has not been set.

Default value: `teamtraining.local`

Configuration of LDAP over SSL

To ensure a secure communication between the TTS PERFORMANCE SUITE server and the LDAP server, the use of SSL (Secure Sockets Layer) is strongly recommended. Only two additional steps are necessary to enable SSL:

1. The provider-url must match the URL scheme of ldaps, for example `ldaps://secureldap.teamtraining.local:636`.
2. The SSL certificate of the secure LDAP server must be stored in a keystore of trusted certificates and provided to the JVM or the application server. TTS PERFORMANCE SUITE server checks the certificate's availability at runtime

You can import the certificate into the JVM using *keytool*:

```
1 keytool -import -v -trustcacerts -alias ldapserver -file ldapserver_cert.cer -
↪ keystore trust-store.ks -storepass secret
```

Then, start the JVM with the keystore by setting the JVM parameters *javax.net.ssl.trustStore* and *javax.net.ssl.trustStorePassword*:

```
1 java -Djavax.net.ssl.trustStore=truststore.ks -Djavax.net.ssl.trustStorePassword=
↪ secret
```

⚠ If configuration of the JVM parameter is not possible, the keystore of the system user (`${user.home}/.keystore`) or the keystore of the JVM (`${ja- va.home}/lib/security/cacerts`) may be used instead.

Migration of deprecated LDAP-configuration (6.3.2 and earlier)

Both properties, `ldap.server` and `ldap.user_dn`, are deprecated and not supported anymore (since version 6.4.1). Please adapt your configuration in the following way:

- `SimpleLdapLoginModule` has to be registered
- `ldap.server` must be replaced with `ldap.default.provider.url`
- `ldap.user_dn` must be replaced with `ldap.default.user.dn.template`

Known restrictions of the LDAP interface

- Only "internal" authentication with username and password is supported. Alternative mechanisms in accordance with SASL (RFC 2222) are not possible.
- Start-TLS (RFC-2830) is currently not supported.
- LDAP groups have to be associated to user entries. Users associated with groups are not supported.

6.4.3 Single-Sign-On (SSO)

Request-based Single-Sign-On

Many SSO systems use elements of the HTTP request (headers, parameters) to indicate if a user is signed in or not. With this knowledge in mind, the generic login module already supports a large number of SSO products.

To enable SSO as authentication mode, the login module `de.tts.bd.business.login.request.RequestElementsLoginModule` must be registered in the property `loginModules` of the user service.

Some of the configuration properties of the SSO authentication have to meet a specific syntax to get the values extracted properly from the HTTP request. Below, the extractors for particular elements of the HTTP request are described.

Configuration of LDAP over SSL

To ensure a secure communication between the TTS PERFORMANCE SUITE server and the LDAP server, the use of SSL (Secure Sockets Layer) is strongly recommended. Only two additional steps are necessary to enable SSL:

1. The provider-url must match the URL scheme of ldaps, for example `ldaps://secureldap.teamtraining.local:636`.
2. The SSL certificate of the secure LDAP server must be stored in a keystore of trusted certificates and provided to the JVM or the application server. TTS PERFORMANCE SUITE server checks the certificate's availability at runtime

You can import the certificate into the JVM using *keytool*:

```
keytool -import -v -trustcacerts -alias ldapserver -file ldapserver_cert.cer -keystore trust-store.ks -storepass secret
```

Then, start the JVM with the keystore by setting the JVM parameters *javax.net.ssl.trustStore* and *javax.net.ssl.trustStorePassword*:

```
java -Djavax.net.ssl.trustStore=truststore.ks -Djavax.net.ssl.trustStorePassword=secret
```

⚠ If configuration of the JVM parameter is not possible, the keystore of the system user (`${user.home}/.keystore`) or the keystore of the JVM (`${ja- va.home}/lib/security/cacerts`) may be used instead.

Migration of deprecated LDAP-configuration (6.3.2 and earlier)

Both properties, `ldap.server` and `ldap.user_dn`, are deprecated and not supported anymore (since version 6.4.1). Please adapt your configuration in the following way:

- `SimpleLdapLoginModule` has to be registered
- `ldap.server` must be replaced with `ldap.default.provider.url`
- `ldap.user_dn` must be replaced with `ldap.default.user.dn.template`

Known restrictions of the LDAP interface

- Only "internal" authentication with username and password is supported. Alternative mechanisms in accordance with SASL (RFC 2222) are not possible.
- Start-TLS (RFC-2830) is currently not supported.
- LDAP groups have to be associated to user entries. Users associated with groups are not supported.

6.4.4 Single-Sign-On (SSO)

Many SSO systems use elements of the HTTP request (headers, parameters) to indicate if a user is signed in or not. With this knowledge in mind, the generic login module already supports a large number of SSO products.

To enable SSO as authentication mode, the login module `de.tts.bd.business.login.request.RequestElementsLoginModul` must be registered in the property `loginModules` of the user service.

Some of the configuration properties of the SSO authentication have to meet a specific syntax to get the values extracted properly from the HTTP request. Below, the extractors for particular elements of the HTTP request are described.

⚠ The value of an extractor contains two elements separated through a dot - the first representing the type of extractor, the second the value to extract. For example, the configuration ***parameter.user*** creates an extractor which reads the value of the URL parameter user.

Extractor: Parameter

Extracts a value from a URL parameter.

Parameter: Name of a request parameter

Values: /webaccess/index.do?user=musterman
(extractor parameter.user determines the value "musterman")

Extractor: parameterValues

Extracts a list of values from all URL parameters with the same name.

Parameter: Name of a request parameter

Values: /webaccess/index.do?group=sales&group=accounting
(extractor parameterValues.group determines the values {"sales","accounting"})

Extractor: Header

Extracts a value from an HTTP header.

Parameter: Name of a request header

Values:
GET /webaccess/index.do
...
X-User: musterman
...
(extractor header.X-User determines the value “musterman”)

Extractor: Cookie

Extracts a value from a Cookie.

Parameter: Name of a Cookie

Values:
GET /webaccess/index.do
...
Cookie: email=musterman@mustercompany.local; username=musterman
...
(extractor cookie.email determines the value”musterman@mustercompany.local”)

Extractor: Constant

Simply extracts the value given to the parameter, not from the HTTP request.

Parameter: Constant value

Values: extractor constant.en-us determines the value “en-us”

⚠ The configuration options of the SSO login module are divided between properties concerning the authentication and properties used to map value to the user profile.

request.auth.value

Extracts a value to validate a user. If the value is available and could be validated successfully, the user is authenticated.

Example: `header.user`
(extractors returning a value list as *parameterValues* are not allowed)

request.auth.validator

Defines the validator which validates the returned value of property *request.auth.value*.

Example: `notEmpty`
(currently, only the notEmpty validator exists. Thus, a user is authenticated if the value of *request.auth.value* is not empty)

request.attr.name

Extracts the (login) name of a user.

Example: `parameter.user`
(extractors returning a value list as *parameterValues* are not allowed)

request.attr.uniqueId

Extracts a unique identifier (e.g., email address or the username again).

Example: `parameter.email`

request.attr.firstname

Extracts the first name of a user.

Example: `parameter.firstname`

request.attr.lastname

Extracts the last name of a user.

Example: `parameter.lastname`

request.attr.email

Extracts the e-mail address of a user.

Example: `parameter.email`

request.attr.language

Extracts the language of the user which will be used as the edit and the interface language within the TTS PERFORMANCE SUITE server.

Allowed encoding: ISO 639, ISO 3166, RFC 1766

Example: `constant.en`

request.attr.language.encoding

Specifies the encoding of the language provided by the *request.attr.language** property.

Allowed values: `iso639`, `iso3166`, `rfc1766`

Example: `iso639`

request.attr.editLanguage

Extracts the edit language of the user which will be used within TTS PERFORMANCE SUITE. This value overrides the value of *request.attr.language*.

Allowed encoding: ISO 639, ISO 3166, RFC 1766

Example: `constant.en`

request.attr.editLanguage.encoding

Extracts the encoding of the edit language provided by property *request.attr.editLanguage*.

Allowed values: `iso639`, `iso3166`, `rfc1766`

Example: `iso639`

request.attr.interfaceLanguage

Extracts the interface language of the user which will be used within TTS PERFORMANCE SUITE. This value overrides the value of *request.attr.language*.

Allowed encoding: ISO 639, ISO 3166, RFC 1766

Example: `constant.en`

request.attr.interfaceLanguage.encoding

Extracts the encoding of the interface language provided by property *request.attr.interfaceLanguage*.

Allowed values: `iso639`, `iso3166`, `rfc1766`

Example: `iso639`

request.attr.groupMembership

Extracts the group membership of a user. The determined group names serve as input values to identify maintenance and process roles within TTS PERFORMANCE SUITE server.

The mapping of group names is done in the administration section of the Curator (*Assign authorizations to external users*). The extracted group names must match the ones set up in the Curator (case sensitive).

Example: `parameter.group`

request.attr.groupMembership.separator

Defines the way in which the return value of *request.attr.groupMembership* is separated. If a value list was returned (parameterValues extractor), this property will be ignored.

Allowed values: `comma`, `whitespace`, `semicolon`

Example: `comma`

Example:

```
1  ttkf.accelerator.user.loginModules = de.tts.bd.business.login.request.
↪ RequestElementsLoginModule
2  ttkf.accelerator.user.request.auth.value = header.User
3  ttkf.accelerator.user.request.auth.validator = notEmpty
4  ttkf.accelerator.user.request.attr.name = header.User
5  ttkf.accelerator.user.request.attr.uniqueId = header.User
6  ttkf.accelerator.user.request.attr.firstname = parameter.fname
7  ttkf.accelerator.user.request.attr.lastname = parameter.lname
8  ttkf.accelerator.user.request.attr.email = cookie.email
9  ttkf.accelerator.user.request.attr.language = constant.en
10 ttkf.accelerator.user.request.attr.language.encoding = iso639
11 ttkf.accelerator.user.request.attr.groupMembership = parameterValues.group
```

With this sample configuration, a user is authenticated successfully if the Request header "User" exists and it is not empty. The user profile is built from request parameters, the e-mail address is read from a cookie and the language is set to English. A list of parameter values provides information on the user's group membership.

The following HTTP request authenticates manager and salesman Mario Musterman and populates his profile with information:

```
1  GET /webaccess/index.do?fname=Mario&lname=Musterman&group=Sales&group=Management
2  Host: www.ttkfserver.musterfirma.local
3  User-Agent: Mozilla/5.0
4  Cookie: email=Mario.Mustermann@musterfirma.local
```

```
5 ...
```

Windows-based Single-Sign-On

Purpose of Windows-based SSO is to provide a Single-Sign-On for end users when accessing the TTS PERFORMANCE SUITE WebAccess. The end user is authenticated against an active directory and authorized by permissions mapped to its LDAP-profile.

If authentication fails, an HTTP-401 standard error message is displayed in the browser. In case the user was authenticated successfully, but is missing authorization, an exception is thrown and access is denied.

An advantage of Single-Sign-On, besides automatic login, is the use of existing user information and structures just by mapping LDAP groups to TTS PERFORMANCE SUITE-internal permissions.

This feature uses NTLM v2 and/or Kerberos for negotiation in the authentication process. In the configuration, you can define which protocol is to be used to authenticate the user.

Restrictions

The current implementation restricts the usage of the Windows-based-Single-Sign-On in two ways:

- Current SSO does not provide any fallback mechanism to other login methods as database authentication.
- If authentication fails, the browser automatically sends a HTTP-401 error message combined with a pop-up requesting for basic authentication. The user might login with valid credentials, even though the Single-Sign-On has failed (for example due to different domain location).

Requirements

- Windows OS: This feature supports Windows OS only, since the Account lookup works locally and in Active Directory via Win32 API.
- The WebAccess and/or Curator site must be in the Intranet Zone.
- Integrated Windows Authentication must be enabled in the browser preferences.
- Domain: The application server must be in the corresponding domain.

Configuration

To enable the Windows-based-SSO, a few configuration steps are necessary. First, the login module `ttn.bd.login.sso.windows.WindowsSSOLoginModule` must be defined in user service.

Next, the LDAP needs to be specified. Without these bits of information, a successful login is not possible.

Last but not least, the Windows-based-SSO contains a few parameters that must be set.

loginModules.WindowsSSOLoginModule.populator

Specifies the populator of the login module, populating user profile and authorities. If not configured, the default populator (LDAP) is used.

Value: `ttn.bd.login.sso.windows.authorization.WindowsSSODatabase-AuthoritiesPopulator`

login.sso.windows.principalFormat

Specifies the name format for the principal.

Values:

- `fqn`: Fully qualified names, such as domain\username. When unavailable, a SID is used. (default)
- `sid`: SID in S-format.
- `both`: Both a fully qualified name and a SID in the S- format. The fully qualified name is placed in the list first.

login.sso.windows.roleFormat

Specifies the name format for the role.

Values:

- `fqn`: Fully qualified names, such as domain\username. When unavailable, a SID is used. (default)
- `sid`: SID in S-format.
- `both`: Both a fully qualified name and a SID in the S- format. The fully qualified name is placed in the list first.

login.sso.windows.allowGuestLogin

Allows guest login. When true and the system's Guest account is enabled, any invalid login succeeds as Guest.

Values:

- `true`
- `false` (default)

login.sso.windows.securityFilterProviders

Defines a list of security filter providers.

Value: `waffle.servlet.spi.NegotiateSecurityFilterProvider`

login.sso.windows.protocols

A comma-separated list of security protocols supported by the NegotiateSecurityFilterProvider. May be `Negotiate` or `NTLM` (or a combination of both).

Example: `NTLM,Negotiate`

Example:

```
ttkf.accelerator.user.loginModules=ttn.bd.login.sso.windows.WindowsSSOLoginModule
ttkf.accelerator.user.login.sso.windows.principalFormat = both
ttkf.accelerator.user.login.sso.windows.roleFormat = both
ttkf.accelerator.user.login.sso.windows.securityFilterProviders = waffle.servlet.
↪ spi.NegotiateSecurityFilterProvider
ttkf.accelerator.user.login.sso.windows.protocols = NTLM
ttkf.accelerator.user.ldap.defaultprovider.url = ldap://server.domain.local:port/
ttkf.accelerator.user.ldap.defaultbase.dn = OU=User,OU=TTS,DC=teamtraining,DC=
↪ local
```

```
ttkf.accelerator.user.ldap.defaultmanager.dn = CN=admin,OU=Sonstige,OU=User,OU=TTS
↪ ,DC=tts,DC=local
```

```
ttkf.accelerator.user.ldap.defaultmanager.password = password
ttkf.accelerator.user.ldap.defaultdialect = de.tts.bd.business.login.ldap.ActiveDirectoryDialect
ttkf.accelerator.user.ldap.defaultdialect.adsDomain = teamtraining.local
ttkf.accelerator.user.ldap.defaultdialect.user.filter = (&(objectClass=user)(userPrincipalName=${ttc:user}@${ttc:adsDomain}))
```

6.4.5 SAML Single-Sign-On

With SAML, a new SSO (Single Sign-on) variant is supported for the WebAccess and Curator. To activate SAML, you have to enable the login module in the application-config.properties file:

```
ttkf.server.user.loginModules=com.tts.authentication.saml.login.SamlLoginModule
```

6.4.5.1 application-config Configuration

ttkf.server.saml.config.connections.path

Specifies the location of the SAML YAML configuration file.

Relative paths point to a `saml` folder in the store base path: `${storeBasePath}/saml.`

S3 URLs can also be configured, e.g. `s3://bucket/path/saml-config.yaml.`

Default Value: `saml-config.yaml`

ttkf.server.saml.config.connections.check.cron

How often the configuration file should be reloaded.

Default value: `*/10 * * * * *` (every 10 seconds)

ttkf.server.saml.config.request.timeout

Time in seconds to wait for a response from the IDP.

Default value: `3600`

6.4.5.2 saml-config.yaml

The *saml-config.yaml* contains the SAML configuration in detail. Multiple IDPs can be connected at the same time. Comprehensive documentation with a detailed description of the configuration options is available if required. Please feel free to contact us.

6.4.5.3 Additional configuration

userlist.workflowcheck

For better collaboration the mechanism for owners and assignees has been refactored. Instead of showing either all external users or none of them, now the external users are filtered based off their persisted permissions.

The requirement for this feature is to import all authors from the external identity provider (e.g., ldap), see chapter *user import*.

Default value: `true`

Values:

- `true`: filter owners and assignees based off persisted permissions
- `false`: filter local users, show external users

6.5 Logging

The tts performance suite server uses log4j2 to log to different media, e.g., console or logfile. These logs are very important to monitor the application at runtime and to identify causes of errors, instabilities, or other malfunctions.

By default, the tts performance suite server logs with the log level INFO to the standard out only. To customize the logging behavior a separate `log4j2.properties` file must be included with one of the following methods.

- As a jvm parameter:

```
-Dlog4j.configurationFile=C:\path\to\log4j2.properties
```

- Via catalina.properties:

```
log4j.configurationFile=C:\\path\\to\\log4j2.properties
```

- Provide the parameter via the context parameter `log4jConfiguration` e.g., in the context.xml or server.xml

```
<Context>
...
<Parameter name="log4jConfiguration"
value="file:///C:/path/to//log4j.application.properties" />
</Context>
```

The last method is recommended. It allows to use different configuration files for Curator and WebAccess and thus e.g., to log to different logfiles respectively.

The following section shows a few examples on how to configure the tts performance suite logging for specific use cases. Additional information can be found in the official log4j2 documentation (see https://logging.apache.org/log4j/2.x/manual/configuration.html#Properties)

Example 1

```
monitorInterval=30
rootLogger.level = INFO
rootLogger.appenderRef.stdout.ref = STDOUT
appender.console.type = Console
appender.console.name = STDOUT
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = %-5p [${web:servletContextName:-}][%d] %c [%t]
- %X - %m%n
```

Example 1 shows a sample logging configuration, that prints all log messages of level INFO and above to the standard out using the given pattern. Note that for runtime configuration

changes, the `monitorInterval` must be present in the properties file. The application will check for changes in the logging configuration in the specified interval (in seconds).

Example 2

```
appender.rolling.type = RollingFile
appender.rolling.name = RollingFile
appender.rolling.fileName = c:/temp/curator.log
appender.rolling.filePattern = c:/temp/curator-%d{MM-dd-yy-HH}-%i.log.gz
appender.rolling.layout.type = PatternLayout
appender.rolling.layout.pattern = %-5p [$${web:servletContextName:-}] [%d] %c [%t]
- %X - %m%n
appender.rolling.policies.type = Policies
appender.rolling.policies.size.type = SizeBasedTriggeringPolicy
appender.rolling.policies.size.size=100MB
appender.rolling.strategy.type = DefaultRolloverStrategy
appender.rolling.strategy.max = 5
logger.upload.name = com.tts.document.upload
logger.upload.level = DEBUG
```

Example 2 shows how to add an additional logger. In this example, it logs all log messages from the package `com.tts.document.upload` with level DEBUG and above to separate files using a rolling file appender.

6.6 Version control and Workflow service

6.6.1 Version control

The Release service provides version control for documents and their content. Any change to a document's properties or its content results in a new version. When combined with the Workflow engine, transitions of the workflow status are version controlled as well.

⚠ You can roll back and delete a version, as well as compare it with other revisions or with the current version.

⚠ With version control enabled, the hard disk space used by the repository and the size of the database will increase significantly. For further details, see the hardware requirements section.

Service name: releaseService (maintenanceInterval:10)

revisionDeleteEnabled

Allows for the deletion of document revisions.

Possible values:

- `true`
- `false` (default)

maxUpdates

Defines the maximum number of updated documents within the maintenance interval.

Possible values:

- Any positive integer

- `50` (default)
- `0` (all buffered)

updateAllOnStartup

Defines whether to recalculate the latest released version for all documents on startup or not.

Possible values: - `true` - `false` (default)

6.6.2 Workflow service

The Workflow service allows for creating and administering workflows that are based on document types. In addition to that, workflow transitions are permission controlled through assignable author roles.

Workflow functions

Workflow functions are actions that will be executed when a document's workflow status is changed. To define which functions are triggered by which transition, use the workflow administration section (>*Settings* >*Operational structuring* >*Workflows*). Read on for a brief description of the currently available functions:

IncreaseVersionFunction

Increases the version of the document.

NotifyOwnersFunction

Sends e-mails to the document's owner and assignee when the workflow status changes.

NotifyAssigneeOnlyFunction

Sends an e-mail to the document's assignee when the workflow status changes.

NotifyOwnerOnlyFunction

Sends an e-mail to the document's owner when the workflow status changes.

NotifyMrolesFunction

Sends e-mails to all role members of document's target status after a status change.

CleanupVersionsFunction

All superfluous versions of a document will be deleted upon calling this function, keeping only versions that have been 'published' at some point, as well as all versions that are newer than the last published version.

RevokePublishingFunction

Removes the release flag of a document so the document won't appear in the documentation portal.

⚠ For an in-depth documentation of the version control and workflow service features, please refer to the Administrator manual.

6.7 Cache service

This service provides a distributed caching facility. The use of caches plays a key role in achieving high performance by minimizing the number of roundtrips to the database. Distributed caches are used to share data with other applications, for instance a WebAccess.

Changes made to a distributed cache are reflected in each application, even if they are running on different machines.

⚠ **Caching may fail if the server has multiple network adapters attached and at least one adapter is disabled or not working. In this case, replace "localhost" with the network adapter's IP in the configuration you want to use. If the WebAccess and the Curator are running on the same server, 127.0.0.1 should be used**

⚠ You might face some warning messages concerning the buffer size of the operating system:

```
1 WARN [2012-09-26 16:25:34,825] [Startup] org.jgroups.protocols.UDP
2 [main] - send buffer of socket java.net.DatagramSocket@1e247e2 was
3 set to 640KB, but the OS only allocated 131.07KB. This might lead to
4 performance problems. Please set your max send buffer in the OS
5 correctly (e.g. net.core.wmem_max on Linux)
```

⚠ In this case, you need to adapt the OS settings, since those problems may result in performance issues. Please ask your system administrator to help you with these settings.

Service name: cache

cacheDomain

Defines the name of the domain in which this application node is located.

Possible values:

- `workbench`
- `docportal`

cachePreload

Enables preloading via bootstrap loader.

Possible values:

- `true` (default)
- `false`

`peer.discovery`

Sets either multicast or unicast cache peer discovery mode.

Possible values:

- `multicast` (default)
- `unicast`

peer.listener.host

Defines the host name on which the listener is listening.

Example: `localhost`

peer.listener.port

Sets the port on which the listener is listening (necessary in unicast mode only).

Example: `7800`

peers.multicastGroupAddress (multicast only)

Sets the address of the multicast group this application should join in order to receive cache information.

Example: `228.8.8.8`

peers.multicastGroupPort (multicast only)

Defines the port of the multicast group this application should join in order to receive cache information.

Example: `45566`

peers.timeToLive (multicast only)

Specifies how far the packages are propagated (0-255).

Possible values:

- `0`: the same host
- `1`: the same subnet (default)
- `32`: the same site
- `64`: the same region
- `128`: the same continent
- `255`: unrestricted

peers.workbench (unicast only)

Comma-separated list of host names with ports of peers located in the Curator cache domain.

Example: `workbench:7800`

peers.docportal (unicast only)

Comma-separated list of host names with ports of peers located in the WebAccess cache domain.

Example: `webaccess1:7800,webaccess2:7800`

The EHCacheProvider offers additional configuration options. Please refer to project's website: http://ehcache.sourceforge.net/documentation/configuration.html (section ehcache.xml and Other Configuration Files)

If running on an OS with active IPv6, be sure to start the application server with IPv4 preference by means of the JVM parameter "-Djava.net.preferIPv4Stack=true".

6.8 Repository service

The repository service provides access to the document repositories where the content of each document is stored. When uploading document content, the files are first spooled to a local

directory (spool store). Within a defined maintenance interval, the files will then be uploaded to their repositories asychronously.

Service name: repository (maintenanceInterval:5)

repository.maintenance.interval

Verifies the availability of the specified document repositories in an interval given in seconds.

Possible value: any possible value

Example: `300`

spool.maxRetries

Defines how often a failed upload will be retried before the document is marked as "upload failed".

Possible value: any positive integer value

Example: `5`

repository.url

S3-URL pointing to the repository behind MinIO. The URL contains bucket and root path.

Possible value: S3-URL

Example: `s3://<bucket>/<path>`

repository.endpoint

The local address of the MinIO server. No trailing slashes allowed.

Possible value: URL

Example: `http://localhost:9000`

repository.endpoint.external

The address of a used load balancer or proxy. No trailing slashes allowed.

Possible value: URL

Example: `http://proxy`

repository.accessKeyId

The accessKeyId of MinIO.

repository.secretKey

The secretKey of MinIO.

⚠ By default, there exist repository providers for file system and URL. Providers for other kinds of repositories are available upon request.

6.9 Notification service

The Notification service automatically sends e-mails when certain events occur in the lifecycles of monitored documents. E.g., users may set up to be notified when documents they are interested in get modified.

The service queues all e-mails and sends them periodically within a defined interval (maintenance interval).

⚠ This service is disabled by default.

Service name: email (maintenanceInterval:10)

enabled

Enables or disables the service.

Possible values:

- `true`
- `false` (default)

mail.smtp.host

Sets the name or IP address of the SMTP server.

Example: `smtp.mymailserver.com`

mail.smtp.port

Sets the port of the SMTP server.

Example: `25`

mail.smtp.auth

True if the provided SMTP server requires authentication.

Possible values:

- `true` (default)
- `false`

mail.smtp.user

Defines the username to authenticate against the SMTP server.

Example: `tts`

mail.smtp.pass

Defines the password to authenticate against the SMTP server.

Example: `ttspass`

mail.smtp.starttls.enable

Enables STARTTLS command (if supported by the SMTP server) to switch the connection to a TLS-protected connection before issuing any login commands.

Note that an appropriate trust store must be configured so that the Producer will trust the server's certificate. Disabled by default.

Possible values:

- `true`
- `false` (default)

mail.smtp.connectiontimeout

Represents the connection timeout in milliseconds.

Possible value: any positive integer

Example: `5000`

mail.smtp.timeout

Represents the timeout to send data in milliseconds.

Possible value: any positive integer

Example: `30000`

mail.smtp.sendpartial

Sends the e-mail even if one of the recipients is invalid.

Possible values:

- `true` (default)
- `false`

default.email.dispatcher

To specify the e-mail address of the sender

Possible value: Any valid e-mail address

Example: `curator@domain.com`

default.email.bcc

Defines an e-mail address that is used for blind carbon copy messages.

Possible value: Any valid e-mail address

Example: `bcc@domain.com`

⚠ For further information about possible parameters, see the following website: http://-java.sun.com/products/javamail/javadocs/com/sun/mail/smtp/package- summary.html

6.10 Language service

Language-related functions in the tts Server are provided by the Language service. Languages are expressed either as lowercase ISO 639 language codes (interface language), or as RFC 1766 compliant language tags (edit language).

⚠ **The server's license controls which languages are available. The settings below are used to restrict the list of available languages. You cannot add languages that are not covered in the license.**

available.edit.languages

Restricts the number of available edit languages defined in the license file.

Possible values:

- `*`: No restriction
- Comma separated RFC 1766 language tags, e.g. `de-de, en-us`

default.edit.language

Sets the default edit language in RFC 1766 format.

Example: `de-de`

available.interface.languages

Sets the available interface languages in ISO 639 format.

Possible values: Comma separated ISO 639 language codes, e.g. `de, en, es`

default.interface.language

Sets the default interface language in ISO 639 format.

Example: `de`

6.11 Template service

The WebAccess is basically a smart template engine that can be configured via the template service. All relevant web pages of the WebAccess may be customized to match a customer's requirements, e.g., with regard to the corporate design.

Configurations concerning customer-specific behavior of user login, handling of process, and maintenance roles (standard and preview mode) are also managed by this service.

Templates are defined in a file named *templates.xml* located in $TTPS_HOME/WEB- INF. tts provides services to support you in defining a custom portal layout and behavior. Please contact your Key Account Manager to receive an offer.

The experienced tts customizing team will be happy to offer professional customizing services.

Service name: templates

config

Defines where the service configuration file is located.

Possible value: `/WEB-INF/templates.xml`

configCustom

Sets the path including a wildcard for all customer configurations.

Possible value: `/WEB-INF/templates.*.xml`, e.g. `/WEB-INF/templates.customername.xml`

configAutoReload

Enables automatic configuration reload when the configuration file has been changed.

Possible values:

- `true`
- `false` (default)

emptyRolesAllowed

Enables login to the WebAccess even if a user has no process roles assigned or there are no roles supplied to the WebAccess.

Possible values:

- `true` (default)
- `false`

defaultRoles

Comma separated list of technical names of process roles to display if a user has no process roles assigned. Works only if the property *emptyRolesAllowed* is set to `false`.

Possible values: any valid technical names of process roles

Example: `worker, manager`

emptyRolesAllowedOnPreview

Allows preview for a user to whom no process roles were assigned.

Possible values:

- `true` (default)
- `false`

defaultPreviewRoles

Comma separated list of technical names of process roles to display on preview if a user has no process roles assigned. Works only if the property *emptyRolesAllowed* is set to `false`.

Possible values: any valid technical names of process roles

Example: `worker, manager`

ignoreMaintenanceRolesOnPreview

Defines whether the maintenance roles are ignored on preview or not by filtering objects visible to a user.

Possible values:

- `true` (default)
- `false`

userSessionTimeout

Sets a session timeout (in seconds) for accessing the WebAccess.

Possible value: any positive integer

Example: `1800`

previewSessionTimeout

Sets a session timeout (in seconds) for accessing the WebAccess preview within Curator.

Possible values: any positive integer

Example: `600`

6.12 Scheduler service

This service handles any asynchronous job execution, which includes the maintenance jobs of all other services.

6.13 Configuration service

This service offers settings for the administration of multiple configurations used in Curator and WebAccess.

⚠ **For more information about multiple configurations, please see the Design Manager guide.**

Service name: config

defaultConfiguration

Defines the default configuration when there are multiple configurations.

Possible value: name of the configuration

Example: `initial`

portalConfigurations

Defines the names of available portal customizings in a comma separated list. These pieces of information are used to provide a list box of customizings within the preview, allowing users to switch the customizing.

Example: `initial, tts, customizing1, customizing`

mapping.browser.$BrowserName

Defines the configuration mapped to the given browser name, where $BrowserName stands for the name of the browser.

Example:

```
1 mapping.browser.OPERA11 = desktop_config
2 mapping.browser.MOBILE_SAFARI = mobile_config
```

Possible values for $BrowserName (snippet): `IE`, `IE11`, `IE10`, `IE9`, `EDGE`, `EDGE14`, `EDGE_MOBILE`, `IEMOBILE10`, `IEMOBILE11`, `FIREFOX`, `FIREFOX48`, `FIREFOX_MOBILE`, `SAFARI`, `MOBILE_SAFARI`, `CHROME,CHROME51`, `CHROME_MOBILE`

mapping.operatingSystem.$OperatingSystemName

Defines the configuration mapped to the given operating system name, where $OperatingSystemName stands for the name of the OS.

Example:

```
1 mapping.operatingSystem.WINDOWS = desktop_config
2 mapping.operatingSystem.IOS = mobile_config
```

Possible values for $OperatingSystemName (snippet): `WINDOWS`, `WINDOWS_10`, `WINDOWS_8`, `WINDOW_7`, `ANDROID`, `ANDROID6`, `ANDROID6_TABLET`, `ANDROID5`, `ANDROID5_TABLET`, `WEBOS`, `PALM`, `IOS` `MAC_OS`, `MOBILE_SAFARI`

⚠ More information on how to use the browser and operating system-specific configurations can be found in the user manual and the administrator's guide.

6.14 Feature service

This service provides a facility to control and configure the behavior of several components (or features) of the application. Because of the complexity of the configuration, this service uses its own configuration file.

Service name: feature

config

Context-relative name of the configuration file for this service or absolute file path introduced by prefix "file:"

Default value: `/WEB-INF/feature-config.xml`

Possible value for absolute path: `file:/configfiles/feature-config.xml`

6.15 Webaccess Gateway Configuration

The WebAccess Gateway allows you to proxy requests from WebAccess to external services, such as REST APIs. Each endpoint is configured with its own URL, headers, and a list of HTTP headers to skip when forwarding requests.

6.15.1 Configuration properties

```
# URL of the proxied service
ttkf.server.gateway.endpoints.<name>.url=<url>

# Custom header sent to the proxied service
ttkf.server.gateway.endpoints.<name>.headers.<header-name>=<value>

# Comma-separated list of headers to skip for this endpoint
ttkf.server.gateway.endpoints.<name>.skippedHeaders=<comma_separated_list>
```

`<name>` is used as the path segment in the gateway URL.

The proxied service is accessible via: `http://localhost:8080/webaccess/gateway/<name>`

6.15.2 Default skipped headers

The following headers are skipped by default and not forwarded to the proxied service:

```
accept-encoding
authorization
connection
content-length
cookie
host
keep-alive
sec-fetch-dest
sec-fetch-mode
sec-fetch-site
sec-fetch-user
transfer-encoding
upgrade
```

6.15.3 Customizing skipped headers

To override the global list of skipped headers: `ttkf.server.gateway.skippedHeaders=<commaseparated list>`

To set endpoint-specific skipped headers: `ttkf.server.gateway.endpoints.<name>.skippedHeaders=<commaseparated list>`

If a header is listed as skipped but explicitly set in the endpoint configuration (e.g., authorization), the configured value will be sent.

Example: `ttkf.server.gateway.endpoints.<name>.headers.authorization` will override the skipped authorization header.

6.15.4 Example configuration

```
1 openai.key=fookey
2 ttkf.server.gateway.endpoints.openai.url=https://api.openai.com/v1
3 ttkf.server.gateway.endpoints.openai.headers.authorization=Bearer ${openai.key}
```

With this configuration, the OpenAI API is accessible via: `http://localhost:8080/webaccess/gateway/openai` To retrieve all supported AI models, call: `http://localhost:8080/webaccess/gateway/openai/models`

6.16 Webaccess redirect specific files to presigned URLs

The WebAccess can be configured to redirect file download requests to presigned URLs. Default behavior is to serve specific file types via presigned URLs, while other file types are served directly by the WebAccess.

Cache-Control Header is set to `no-store` for presigned url to prevent browser/proxy caching.

PresignUrl duration is set to 12 hours.

```
1 ttkf.server.download.content.presign.url.enabled=<true|false> default true
2 ttkf.server.download.content.presign.url.for.fileExtensions=<comma separated>
↪ default .mp4 and .webm
3 #example:
4 #ttkf.server.download.content.presign.url.for.fileExtensions=.mp4,.avi
```

6.17 Miscellaneous parameters

These configuration parameters do not belong to a specific service:

ttkf.integrator.collectionContentHandler.customSupportedTypes

Defines supported mime-types for files within collections (e.g., zip-archives). The mime types have to be added in the format `{prefix}.{mimetype}={priority}`.

prefix: ttkf.integrator.collectionContentHandler.customSupportedTypes

mimetype: any valid mime-type

priority: any natural number greater than 9

Example:

```
ttkf.integrator.collectionContentHandler.customSupportedTypes.text/javascript=10
ttkf.integrator.collectionContentHandler.customSupportedTypes.text/richtext=11
ttkf.integrator.collectionContentHandler.customSupportedTypes.text/plain=12
```

ttkf.server.queue.failure.delay

This parameter is optional.

Defines the delay between the next repetition of a failed command.

Example:

```
ttkf.server.queue.failure.delay=100
```

ttkf.server.loginTokenLifetimeMin

This parameter is optional.

Defines the lifetime of a loginToken in minutes (for WebAccess and Curator).

The default value -1 will lead to an infinite lifetime.

Default value: `-1`

Example:

```
ttkf.server.loginTokenLifetimeMin = 10
```

ttkf.server.otp.expiration.seconds

This parameter is optional.

Defines the time in seconds until *One Time Passwords (OTPs)* expire.

Default value: `60`

Example:

```
ttkf.server.otp.expiration.seconds=30
```

ttkf.server.content.security.policy.restrictSVG

This parameter is optional.

If set to true, a content-security-policy is added as a HTTP response header to mitigate the threat of XSS attacks in SVG files.

Default value: `true`

Example:

```
ttkf.server.content.security.policy.restrictSVG=false
```

ttkf.server.content.security.policy.restrictXML

This parameter is optional.

If set to true, a content-security-policy is added as a HTTP response header to mitigate the threat of XSS attacks in XML files.

Default value: `true`

Example:

```
ttkf.server.content.security.policy.restrictXML=false
```

6.18 Search service

6.18.1 Solr Search Service

The search service is used to provide multilingual search capabilities for the most important objects in the tts server. You can search in the titles and properties of the following objects:

- processes
- topics
- courses
- documents
- glossary entries

In the Curator you find additionally:

- users

The search service is based on Apache Solr, a text search engine written entirely in Java. In order to find results quickly and accurately, the metadata of the mentioned objects are stored in separate indexes. These are implementation details from Solr.

The indexes are created automatically after database initialization or after tts server updates (containing internal database updates).

⚠ A full recreation and re-indexing can be forced over the installation page. There exists a link “Rebuild search index”, which will rebuild the entire index asynchronously. Administrator privileges are necessary.

Solr mandatory parameter

ttkf.server.search.server.url

This mandatory parameter defines how the Curator and WebAccess can reach the Solr server.

```
ttkf.server.search.server.url={full context path to the Solr instance}
```

Example:

```
ttkf.server.search.server.url=http://127.0.0.1:8983/solr/tts-server
```

6.18.2 Search parameter

ttkf.server.search.max.result.size

This optional parameter defines the maximum number of unfiltered search results which should be returned by a query.

By default, each search request will be limited to 10.000 results.

Default value: `10000`

Example:

```
1 ttkf.server.search.max.result.size=5000
```

ttkf.server.search.disjuntion.tie

This optional parameter specifies the *tiebreaker* for the *DisMax* parser in the SolR search engine.

When searching multiple fields, this factor defines the impact that additional matches will have on the score of a result item. For example, when a query matches the title and the description of a document, with the best match found in the title, the score of the additional match in the description will be multiplied by the tie factor and added to the score of the best match in the title of the document.

With a tie factor of 0.0, additional matches will be ignored.

With a tie factor of 1.0, the overall score of a result item will be the sum of all field matches. Usually, the tie factor should be much less than 1.0.

Possible value: Between 0 and 1

Default value: `0.5`

Example:

```
1 ttkf.server.search.disjunction.tie=0.0
```

See https://solr.apache.org/guide/solr/9_5/query-guide/dismax-query-parser.html for more details.

6.18.3 Highlighting parameter

The following configuration parameters for the highlighting of query results are optional. Each highlighting parameter has a direct equivalent in the SolR syntax.

See https://solr.apache.org/guide/solr/9_5/query-guide/highlighting.html for more details.

ttkf.server.search.highlighting.snippets

Maximum number of highlighted snippets per field.

Possible value: Positive integer

Default value: `1`

Example:

```
1 ttkf.server.search.highlighting.snippets=2
```

ttkf.server.search.highlighting.fragsize

Size in characters for highlighted fragments.

Possible value: Positive integer

Default value: `256`

Example:

```
ttkf.server.search.highlighting.fragsize=256
```

ttkf.server.search.highlighting.simple.pre

HTML markup to insert at the start of a match in the fragment.

Default value: `<em>`

Example:

```
ttkf.server.search.highlighting.simple.pre=<p>
```

ttkf.server.search.highlighting.simple.post

HTML markup to insert at the end of a match in a fragment.

Default value: `</em>`

Example:

```
ttkf.server.search.highlighting.simple.post=</p>
```

ttkf.server.search.chunksize

Defines the number of elements, which shall be indexed by Solr at the same time. It is an optional parameter

Default value: `100`

Example:

```
ttkf.server.search.chunksize = 50
```

ttkf.server.search.epss.max.group.count

Defines the maximum of groups Solr should return in a search response. This is an optional parameter.

Default value: `50`

Example:

```
ttkf.server.search.epss.max.group.count = 100
```

ttkf.server.search.epss.group.limit

Defines the maximum limit of rows Solr should return in a search response. This is an optional parameter.

Default value: `1000`

Example:

```
ttkf.server.search.epss.group.limit = 100
```

ttkf.server.search.optimize.solr.index.interval

Defines the interval to optimize the solr search index. Value is given in hours, default value is 24 hours.

Example:

```
ttkf.server.search.optimize.solr.index.interval = 24
```

6.18.4 Search Index API

The server provides an endpoint to trigger the recreation of the Solr search index. This API is based on **token** authorization. Requests with missing or invalid token will be rejected.

The token is configured with the following property:

```
ttkf.server.properties.search.reindex.token=<your_token>
```

The API can then be called with:

```
https://url/curator/mvc/searchIndexAdmin/reindex?token=<your_token>
```

6.19 Security configuration

Password security

ttkf.server.security.password.reset.active

This is the global option to enable (`true`) or disable (`false`) the resetting of passwords, both initial passwords or expired ones.

Possible values:

- `true`
- `false` (default)

ttkf.server.security.password.history.size

Specifies the number of passwords in the history. A new user password must be different from the ones contained in the history. Each user has their own history.

Possible value: 1-N

Default value: 4

The server provides restrictions for new passwords. The following restrictions exist:

- Minimum character length (default 7)
- Username may not be contained in password
- First and last name may not be contained in password.
- RegEx condition. Default settings are:
  - At least one lower case character.

  - At least one upper case character.
  - At least one number.
  - At least one special character.

By default, 3 of the 4 conditions must be met.

- The last N passwords of a user cannot be re-used as new password.

The following optional settings exist to configure the password restrictions:

ttkf.server.security.password.min.length

Minimum character length for a new password.

Default value: `7`

ttkf.server.security.password.regex.condition.enable

Activate the regex conditions for new passwords. The conditions are:

- At least one lower case character.
- At least one upper case character.
- At least one number.
- At least one special character.

Possible values:

- `true` (default)
- `false`

ttkf.server.security.password.regex.condition.min.matches.to.pass

At least n regex condition(s) must match for a new password. (see the previous setting for more information about regex conditions)

Possible value: 0-4

Default value: `3`

ttkf.server.security.password.expiration.days

Specifies for how long (days) a password is valid. `-1` disables the expiration check.

Possible values: `-1` to `365`

Default value: `90`

6.19.1 Content Security Policy

A Content Security Policy (CSP) is a header that consists of several directives, e.g.

```
1 Content-Security-Policy: default-src 'none'; script-src 'self' 'unsafe-inline';
↪ img-src 'self' www.acme.com
```

and is used to tell the browser what it should do when it encounters different types of content from different sources.

A concrete example of this is `script-src`. This tells the browser who can execute scripts on our webpage. The example above says only scripts downloaded from our domain (self), or added

inline may be executed. We can also give it a list of trusted domains, for instance for collecting analytics.

There are currently three different cases where the Server sets a CSP:

- If the content type is JSON (e.g., an API endpoint): *Cannot be configured*
- If the content type is SVG or XML (e.g., showing or downloading an image)
- Otherwise, the frame-ancestors directive is set

6.19.1.1 SVG & XML files - anti cross-site scripting

A restrictive CSP is returned when fetching an image that prevents scripts from executing anywhere except when downloaded from the application itself. This also prevents unsafe inline scripts, thereby reducing the chances of cross-site scripting.

It can be disabled, if necessary, with the following parameters:

```
1 ttkf.server.content.security.policy.restrictSVG = false
2 ttkf.server.content.security.policy.restrictXML = false
```

6.19.2 frame-ancestors

At the moment, the default CSP is `frame-ancestors 'self'` and forbids the embedding of our application anywhere except in our own pages. You may extend this list with a comma-separated list of domains:

```
1 ttkf.server.content.security.policy.frame.ancestors.allowed = *.company.de, www.
↪ acme.com
```

This is useful/required if using the QuickAccess for Web or embedding the tts performance suite in an external LMS.

6.19.2.1 Reporting violations

If required, CSP violations can be reported to an endpoint. This is useful for trying out a new CSP or detecting possible security intrusions. It can be done with the following parameter:

```
1 ttkf.server.content.security.policy.report.uri = ...
```

This will add the following directive to the end of all CSPs listed above:

```
1 Content-Security-Policy: ...; report-uri <your endpoint will be here>;
```

6.19.3 Content-Disposition header for downloads

ttkf.server.security.download.attachment.mimetypes

The most common Microsoft Office mimetypes will be delivered with the Content-Disposition header set to "attachment" for any download requests from IE11. If any more mimetypes should be delivered this way they can be added by using this parameter.

Possible value: comma separated list of mimetypes without any blanks

Default value: no default (parameter is optional)

6.20 Security Recommendations

This section describes the latest recommendations for installing a tts performance suite Server securely. Any potential security issues that can be solved with infrastructure changes will be listed here.

These recommendations may depend on the specific environment and not be relevant to all customer setups.

6.20.1 HTTP Strict Transfer Security

If the Curator is, for instance, accessible under the SSL-secured URL https://example.com/curator, an attacker with access to the network can provide a user with the link http://example.com/curator and (given that SSL is no longer used) must not verify their identity and can pretend to be the Curator (e.g. for phishing).

To avoid that, we can send the HTTP header Strict-Transport-Security (HSTS, read more here) which tells the user's browser that they should only access the Curator (or WebAccess) over https. This can be done with Tomcat by enabling the following filter in the `web.xml`:

```
<filter>
    <filter-name>httpHeaderSecurity</filter-name>
    <filter-class>
        org.apache.catalina.filters.HttpHeaderSecurityFilter
    </filter-class>
    <async-supported>true</async-supported>
    <init-param>
        <param-name>hstsMaxAgeSeconds</param-name>
        <param-value>31536000</param-value>
    </init-param>
    <init-param>
        <param-name>hstsIncludeSubDomains</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>antiClickJackingEnabled</param-name>
        <param-value>false</param-value>
    </init-param>
    <!--
    <init-param>
        <param-name>antiClickJackingOption</param-name>
        <param-value>SAMEORIGIN</param-value>
    </init-param>
    -->
</filter>
```

And un-commenting this *filter-mapping*:

```
<filter-mapping>
    <filter-name>httpHeaderSecurity</filter-name>
    <url-pattern>/*</url-pattern>
```

```
    <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

In addition to `Strict-Transport-Security`, this filter sets the `X-Content-Type-Options` header:

- `X-Content-Type-Options: nosniff`
  Prevents the browser from overriding the MIME-type and potentially executing .exe disguised as images.

6.20.1.1 Configuration options for HttpHeaderSecurityFilter

max-age
The above max age (31,536,000 s) is equivalent to a year. The following recommendations are given:

- Wikipedia: "sites should set a period of several days or months depending on user activity and behavior".
- HSTS Preload list requires at least 1 year.
- Qualys (SSL Labs): "It is advisable to assign the max-age directive's value to be greater than 10368000 seconds (120 days) and ideally to 31536000 (one year)."

Returning a value of 0 will cause the Browser to forget the HSTS entry.

hstsIncludeSubDomains

If there are other web services on the same domain which do not use SSL, you may wish to set `hstsIncludeSubDomains` to `false`.

One possibility is to ramp up the `hstsMaxAgeSeconds` value, checking if any problems occur.

antiClickJackingEnabled

This option is not directly related to HSTS but prevents the application from being embedded in foreign websites (and thus prevents Clickjacking).

Feel free to enable this and set `antiClickJackingOption` to `SAMEORIGIN` if:

- Using an installation between 2020 r2 and 2021 r2 (inclusive)
- Not using the QuickAccess for Web
- Supporting IE11

The option has been deprecated from the 2022 release in support of the default Content Security Policy.

All mentioned HTTP headers may alternatively be set in a reverse proxy but the configuration is out-of-scope of this document; please consult the relevant documentation for setting headers in your reverse proxy.

6.20.2 Preventing HTTP Request Smuggling

- This only applies to **Tomcat** installations behind a **reverse proxy** using HTTP/1.1.

On HTTP/1.1, it is possible for web servers to send and receive packages using a length or using chunks of data. They determine this by using headers:

- Content-Length: X
- Transfer-Encoding: chunked

If, however, you send both headers, some web servers will behave differently from others. Technically the Transfer-Encoding header should take priority. However, some web servers don't support this, and others can be tricked into not using it.

If you are using two servers in a chain (e.g., in a reverse proxy scenario), it can be possible to 'smuggle' a second request by manipulating the headers and appending our own request. The smuggled request is then interpreted as a second request from the original user, e.g.

```
POST /login HTTP/1.1
Host: tt-s.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Transfer-Encoding: chunked

16
login=xxx&password=xxx
0
```

```
GET /404 HTTP/1.1
X-Foo: bar
```

The mitigation will depend on the setup and web servers involved. Two possibilities are:

6.20.2.1 Disable Keep-Alive

Separate requests from the same Producer will create new connections. This will increase overhead. To disable Keep-Alive, add `maxKeepAliveRequests="1"` to the `<Connector />` tag.

6.20.2.2 Use HTTP/2

HTTP/2 no longer uses Content-Length preventing ambiguity between web servers.

Even if it not possible to use HTTP/2 between the Reverse Proxy and the users, HTTP/2 can be used in the back end between the Reverse Proxy and Tomcat. To do this in Tomcat, please add the following to your tag:

```
<UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol"/>
```

e.g.

```
<Connector
connectionTimeout="20000"
port="8080"
protocol="HTTP/1.1"
redirectPort="8443"
useBodyEncodingForURI="true">
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol"/>
```

```
</Connector>
```

You will have to refer to the documentation for your reverse proxy for using HTTP/2.

6.20.3 Solr

6.20.3.1 Authentication & Authorization

Ensure some form of authentication is enabled in Solr (as described here: https://-solr.apache.org/guide/solr/9_5/deployment-guide/authentication-and-authorization-plugins.html). This will prevent malicious users who gain access to the machine from accessing (confidentiality) or modifying (integrity) the index in Solr.

Also ensure that Solr is run with a dedicated user with limited privileges (principle of least privilege). This will mitigate any damage if a malicious user does gain access to Solr.

6.20.3.2 Disable Solr Config-API

The Solr Config-API is enabled by default. This feature is not required by the tts performance suite Server and can allow vulnerabilities to be exploited. To disable it, supply the system property `-Ddisable.configEdit=true` at start time. This can usually be added to the `SOLR_OPTS` environment variable in your solr.in.sh script file.

The following command can be used to validate it is disabled:

```
curl https://localhost:8983/solr/tts-server/config -H "Accept:application/json" -H
↪  "Content-type:application/json" -d "{'set-user-property' : {'variable_name':'
↪ some_value'}}"
```

Adjust the URL and the core name where appropriate. A 403-status code will be returned if it has been successfully disabled.

6.21 Dashboard

The Dashboard provides a graphical overview of the usage data in the portal, which is collected and staged by Piwik. It is available via the “Dashboard” link in the portal (provided it is contained in the license and the user has sufficient rights).

6.21.1 License and user rights

The dashboard is a license component and thus must be contained in the license to be available.

Further the user must be assigned an author role which has the right to see the dashboard. This can be configured in the edit author role dialogue under the tab “portal”.

If the portal is available without login, the dashboard and the corresponding link are not visible by default.

6.21.2 Configuration of the Piwik connection

The connection to the Piwik server can be configured in the WEB-INF/templates.CUSTOM.xml used in the scenario.

The following values can be set:

```
<property name="piwikServer">piwik-server-url</property>
<property name="piwikToken">789ASDASD</property>
<property name="piwikSiteIdr">17</property>
<property name="piwikCustomDimensionContentType">1</property>
<property name="piwikDebug">false</property>
```

6.21.3 Specific error pages

If the dashboard is not available due to missing authorization or the like, an error page is displayed instead. This page is located at templates/dashboard/dashboard.error.jsp and distinguishes the following cases:

- The dashboard is not contained in the license.
- The user does not have sufficient rights to see the dashboard.
- The dashboard was called although the portal is available without login (authorization failure).

6.22 User Import Process

User import files (XSD file)

You can find this XSD file available at a public url in your deployed Curator. See resources.

User import files (XML file)

The format has changed and you can verify the validation of the configuration file with the xsd. The difference with the previous format is that only already existing process roles and maintenance roles will be added to an user. Not existing roles will be silently ignored.

User import files (Excel file)

We now manage the domain field differently. There are 3 values authorized:

- local, when the user is authentified with tts performance suite
- ldap, when the user is authentified with ldap
- saml, when the user is authentified with saml

Known issues and limitation

No warnings in reports.

Currently no hints are available in the generated report, if a user is successful created but with warnings.

For example:

- Role assignment failed, because the role does not exist

Excelfile - Username should not contain any whitespaces. Also, no leading or trailing whitespaces. - Excelfile should not contain the same username more than once. Username which are only distinct in upper or lower case will be handled as same username.

When values in field are not supported, you get description of errors in the excel report.

Resources

http(s)://:/curator/schemas/user/excelimport/v1/schema.xsd

6.23 Overview of the login modules

DefaultLoginModule

Login behavior: user from database

Authentication: username / password

LdapLoginModule

Default populators: LdapAuthoritiesPopulator

Configuration parameter: `com.tts.serverfoundation.authentication.ldap.LdapLoginModule`

Login behavior: user from database / temporary runtime user

Authentication: LDAP

ExtendedLdapLoginModule (deprecated)

Default populators: ExtendedLdapAuthoritiesPopulator

Configuration parameter:
`de.tts.bd.business.login.ldap.ExtendedLdapLoginModule`

Login behavior: temporary runtime user

Authentication: LDAP

SimpleLdapLoginModule (deprecated)

Configuration parameter:
`de.tts.bd.business.login.ldap.SimpleLdapLoginModule`

Login behavior: user from database

Authentication: LDAP

WindowsSSOLoginModule

Default populator: WindowsSSOAuthoritiesPopulator

Configuration parameter:
`ttn.bd.login.sso.windows.WindowsSSOLoginModule`

Login behavior: user from database / temporary runtime user

Authentication: Windows SSO

WindowsSSOLoginModule

Populator: ExtendedLdapAuthoritiesPopulator (deprecated)

Configuration parameter:
`de.tts.bd.business.login.ldap.ExtendedLdapAuthoritiesPopulator`

Login behavior: temporary runtime user

Authentication: Windows SSO

6.24 Adapt message for incompatible server & Producer

In a staged rollout of the tts performance suite Producer during an update there might be instances where for a certain period of time incompatible Producers are still installed on the authors' PCs. If an author now tries to login to the Curator a message is being shown which informs the person of the incompatibility.

As of tts performance suite 2018 this message can be supplemented with custom information. For instance, a custom message can now be shown with a point of contact which helps the author solve this issue, thereby reducing unnecessary helpdesk inquiries.

The message can be adapted on server level in the datastore in the file `config/client/version-mismatch-custom.json`

If the file does not exist yet, create a new one and add e.g., the following to this json-file:

```
1 {"message": "Unfortunately, your Producer is not compatible to the tts performance
↪  suite server. Please get in contact with IT@yourcompany.com to request an update
↪ of your Producer."}
```

Currently, only one language is supported.

In case personal information are shown in the message (e.g.: the name of the assigned IT support person) please make sure that you get the approval of those affected. Additionally, please bear in mind that the display of this message does not require a tts performance suite login. Therefore, any sensible information must not be included.

6.25 Configure WebSocket Support for Apache Webserver with AJP

6.25.1 Missing WebSocket Support In AJP

As for now, AJP 1.3 does not support WebSockets. This means, communication via WebSocket is not possible when using AJP between Apache Webserver (e.g., as proxy or reverse proxy) and Apache Tomcat Application Server. This leads to malfunctions within the ribbon (buttons won't be update concerning the current document state) and the user import feature.

⚠ The following error in the log files states missing WebSocket support exception is java.lang.UnsupportedOperationException: HTTP upgrade is not supported by this protocol

6.25.2 How to configure Apache Webserver?

The following modules must be activated within httpd.conf:

```
1 LoadModule rewrite_module modules/mod_rewrite.so
2 LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

Next the urls, which are to be upgraded to websocket protocol must be redirected via rewriting:

```
1 RewriteEngine On
2 RewriteCond %{HTTP:Upgrade} =websocket [NC]
3
4 RewriteRule /curator/ui/document/(.*) ws://localhost:8080//curator/ui/document/$1
↪ [P,L]
```

```
RewriteRule /curator/user/import/websocket(.*) ws://localhost:8080//curator/user/import/websocket/$1 [P,L]
```

6.26 Copilot Connector

The Copilot Connector integrates the tts performance suite server with Microsoft 365 Copilot. It indexes content (Topics, Processes, Courses, and Documents) into a Microsoft Copilot external connection so that the content is searchable and retrievable via Microsoft Copilot.

The connector runs a scheduled job that detects changes to authoring content and synchronizes them with the Microsoft Graph API.

6.26.1 Prerequisites

Before configuring the Copilot Connector, the following items must be set up in Microsoft Azure:

1. A **Microsoft Entra ID (formerly Azure AD) application registration** with the following application permissions granted:

- `ExternalConnection.ReadWrite.OwnedBy`
- `ExternalItem.ReadWrite.OwnedBy`
- `ExternalConnection.Read.All`
- `ExternalConnection.ReadWrite.All`
- `ExternalConnection.ReadWrite.All`
- `User.Read`

2. A **client secret** for the application registration.
3. A **Microsoft Graph external connection** created for the Copilot integration, providing a connection ID.

6.26.2 Configuration

To enable the Copilot Connector, add the following parameters to `application-config.properties`:

```
ttkf.server.ai.copilotconnector.enabled=true
ttkf.server.ai.copilotconnector.clientId=<azure-client-id>
ttkf.server.ai.copilotconnector.clientSecret=<azure-client-secret>
ttkf.server.ai.copilotconnector.tenantId=<azure-tenant-id>
ttkf.server.ai.copilotconnector.connectionId=<graph-connection-id>
```

ttkf.server.ai.copilotconnector.enabled

This parameter is required to activate the module.

Set to `true` to enable the Copilot Connector. If omitted or set to `false`, the connector is disabled.

Default value: `false`

ttkf.server.ai.copilotconnector.clientId

This parameter is required.

The client ID (application ID) of the Azure AD application registration.

ttkf.server.ai.copilotconnector.clientSecret

This parameter is required.

The client secret of the Azure AD application registration.

ttkf.server.ai.copilotconnector.tenantId

This parameter is required.

The tenant ID of the Microsoft Entra ID (formerly Azure AD) tenant used by the application registration.

ttkf.server.ai.copilotconnector.connectionId

This parameter is required.

The ID of the Microsoft Graph external connection that the Copilot Connector writes items to.

6.26.3 Scheduled job configuration

The connector synchronizes content changes on a fixed-rate schedule.

ttkf.server.ai.copilotconnector.job.fixed.rate.in.minutes

This parameter is optional.

Defines the interval in minutes between synchronization runs.

Default value: `5`

Example:

```
1 ttkf.server.ai.copilotconnector.job.fixed.rate.in.minutes=10
```

6.26.4 File extension filtering

The connector filters which files are indexed based on their file extension. Non-content files (e.g., stylesheets, images, fonts) are excluded by default.

ttkf.server.ai.copilotconnector.job.file.extension.blacklist

This parameter is optional.

A comma-separated list of file extensions to exclude from indexing. When configured, this replaces the default blacklist entirely.

Default blacklist: `css,scss,sass,less,png,jpg,jpeg,gif,bmp,ico,svg,webp,tiff,tif,woff,woff2,ttf,otf,eot,map,js`

Example:

```
1 ttkf.server.ai.copilotconnector.job.file.extension.blacklist=css,png,jpg
```

ttkf.server.ai.copilotconnector.job.file.extension.without.allowed

This parameter is optional.

If set to `true`, files without a file extension are also indexed.

Default value: `false`

Example:

```
ttkf.server.ai.copilotconnector.job.file.extension.without.allowed=true
```

6.26.5 Schema Generator

The connector needs a schema generator to create the schema for the Microsoft Graph external connection.

6.27 Translate Service

To use the on-demand translation feature in the QuickAccess, there must be a translation endpoint in the ttkf-server available. This can be configured via the application configuration parameters (depending on the desired translation service)

```
# For Deepl
ttkf.server.translation.service=deepl
ttkf.server.translation.secret=<key>

# For Azure*
ttkf.server.translation.service=azure
ttkf.server.translation.secret=<key>
ttkf.server.translation.azure.region=<region>
ttkf.server.translation.azure.endpoint=<endpoint>
```

* QuickStart: Azure AI Translator REST APIs Prerequisites

To disable translation caching, set:

```
tkf.server.feature.translation.caching = false
```