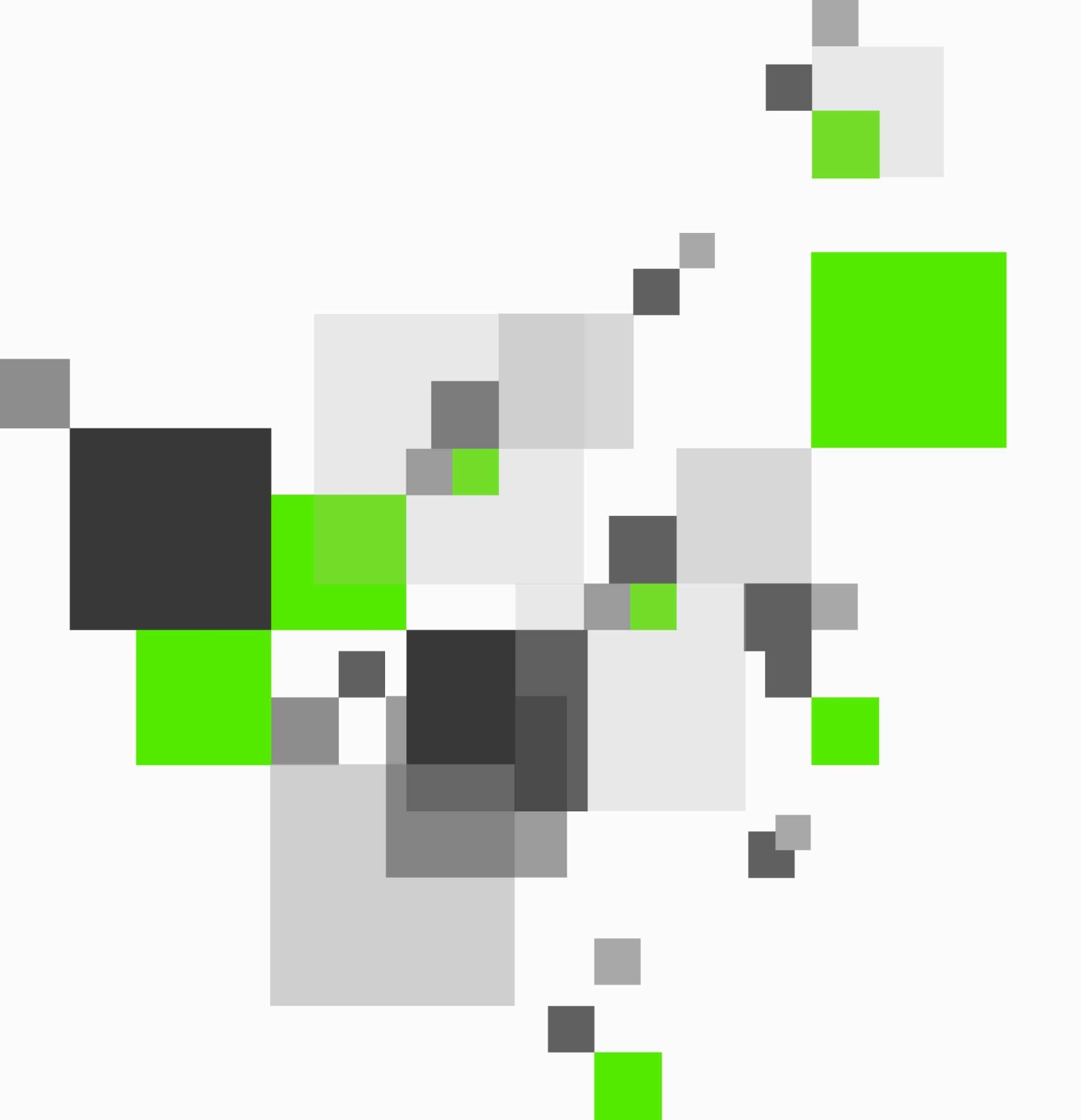
**Beskedfordeler client codesamples**



Changelog:

History of changes:

|  |  |  |
| --- | --- | --- |
| **Versionnumber** | **Date** | **Subject** |
| 1.0 | 30-09-2015 | Definition of codeexamples |
| 1.1 | 26-02-2018 | Updated for Beskedfordeler version 2.1 |
| 1.2 | 28-03-2018 | Fixed beskedfordelerHostname for 2.1  and Updated Gradlew to 4.6 |
| 1.3 | 09-05-2018 | STS-interface library updated from 1.0 to 2.2 |
| 1.3.1 | 16-05-2018 | STS-interface-extra library temporarily introduced to replace sts-bf-soap which was causing conflict. |
| 1.3.2 | 17-06-2018 | Documentation updated to include the option of providing console arguments rather than editing SamplesHelper. SamplesHelper.java has been updated to inform the user if he does not provide the keystore arguments. |
| 1.3.3 | 04-07-2018 | Removed legacy vaerdiliste code examples |
| 1.4 | 06-03-2020 | Added functionality that can validate the XML test message when sending a message.  Documentation updated with XML message examples. |

Table of contents

[1 Codeexamples for Støttesystem Beskedfordeler 3](#_Toc34399206)

[1.1 AfsendBesked.java 3](#_Toc34399207)

[1.2 AfhentBesked.java 4](#_Toc34399208)

[1.3 AfleverBesked.java 4](#_Toc34399209)

[2 Java Code Examples Documentation 4](#_Toc34399210)

[2.1 Sts-interface-2.2-SNAPSHOT.jar 4](#_Toc34399211)

[2.2 Prerequisites 5](#_Toc34399212)

[2.3 Before building and running the samples 5](#_Toc34399213)

[2.4 Building the samples 6](#_Toc34399214)

[2.5 Running the samples from inside gradle 6](#_Toc34399215)

[2.6 Running the JAR executables 6](#_Toc34399216)

[2.7 AfhentBesked.java & AfsendBesked.java using Security Token Service 7](#_Toc34399217)

[2.8 RabbitMQ Message Delivery 7](#_Toc34399218)

[2.9 Wildfly SSL Configuration 7](#_Toc34399219)

[2.10 Web service client security with Apache CXF 7](#_Toc34399220)

[2.11 XML message examples 8](#_Toc34399221)

# Codeexamples for Støttesystem Beskedfordeler

The code examples show how a client can access the services Afsend\_Besked, Afhent\_besked and Aflever\_besked in the Støttesystem Beskedfordeler. The code examples exist in a Java version.

In order to compile the examples, in the unzipped "samples" directory, execute

./gradlew clean build

## AfsendBesked.java

This code example shows how to use the Afsend Besked service when sending messages to the Støttesystem Beskedfordeler. To access the service, the client must first obtain a security token in form of a SAML-assertion issued from the Støttesystem Security Token Service. The example shows that a cached token in the client can be used appropriately.

The client establishes a TLS secured AMQP connection to the Støttesystem Beskedfordeler. When the client establishes the connection the service sends a SASL challenge for the client to present a valid security token. After establishing the connection, the example shows how the client can send multiple messages over the connection. Each message is accompanied by a valid security token.

When the client sends a message it will receive an asynchronous answer on a reply queue. The example shows that using the objects generated from the schema definition, valid message structures can be programmatically constructed.

If the connection fails or the message is rejected by the service, the examples shows how to handle this and how it is possible to save a message temporarily before sending so it is not lost.

## AfhentBesked.java

This code example shows how the Afhent Besked service can be used to receive messages from the Støttesystem Beskedfordeler. To access the service, the client must first obtain a security token in form of an SAML assertion issued from the Støttesystem Security Token Service. The example shows that If an earlier token has been cached in the client this can be used.

The client establishes a TLS secured AMQP connection to the Støttesystem Beskedfordeler. When the client establishes the connection the service sends a SASL challenge for the client to present a valid security token.

After establishing the connection, the example show how the client can receive multiple messages over the connection. Each message must be acknowledged before the next message can be received. The example shows how the client after receiving a message can save this message.

## AfleverBesked.java

The codeexamples show how the Aflever Besked service can be constructed to receive messages from the Støttesystem Beskedfordeler. The Client sets up a REST interface to be called form the Støttesystem Beskedfordeler.

The examples show how the client can receive a message and after receiving it save this message. If the connection fails, the Støttesystem Beskedfordeler will call the interface again and try to deliver the next message in the queue.

# Java Code Examples Documentation

A brief introduction to setting up, configuring, building and running the code examples.

## Sts-interface-2.2-SNAPSHOT.jar

The file contains the necessary XSD components to build the code examples.

. About the samples

The samples support 2 tools:

- afsend besked, which retrieves a token from the Security Token Service and publishes a message

- afhent besked, which retrieves a token from the Security Token Service and retrieves one or more messages

The file "SamplesHelper" contains URL for Security Token Service and host/port for Beskedfordeler.

The token is retrieved using the STS REST API and used in the AMQP connection establishment and AMQP calls.

These samples are created, pointing at the eksterntest environment. Values in SamplesHelper, will have to be changed,

in order to use a different environment.

## Prerequisites

You must have a JDK installed in order to build the project, STS is using functionality from Java 8, so make sure to download and install version 8. You must have a suitable keystore with private key and certificate, known to the Security Token Service. These keystores are NOT provided as part of the samples and must be obtained for the specific environment.

A service agreement (Serviceaftale) must exist to allow publishing (send) and retrieving (afhent) messages. This service agreement must be created using STS Administrationsmodul (or any replacement).

The service agreement must identify the system using the certificate and private key used when running these samples. The service agreement must include sending and retrieving roles, and message types (beskedtyper), and authorized by necessary municipalities (kommuner) supplied by CVR ids (cvr numre).

## Before building and running the samples

To get more log-information, change the logback.xml file, currently set to INFO.

There are values that must be set in SamplesHelper.java and then there are values that can be optionally set in SamplesHelper.java. Those that can optionally set in SamplesHelper are still required to run the samples but can be instead be provided by arguments to the JAR executables described in **Error! Reference source not found.**. Make sure that the values below with the (\*) tag has been setup correctly in SamplesHelper.java, the values without (\*) can be set by arguments or have a default value set in SamplesHelper.java.

The following values are service

agreement-specific;

requestCVRNumber

keyStoreFile

keyStorePassword

The following values are environment-specific;

trustStoreFile

trustStorePassword

beskedfordelerHostname (\*)

beskedfordelerAfhentServiceURI (\*)

beskedfordelerAfsendServiceURI (\*)

## Building the samples

Build using gradle (<https://gradle.org>). See tasks: sh ./gradlew tasks

Build all: sh ./gradlew dist\_all

(For Windows, replace "sh ./gradlew" with "gradlew.bat")

## Running the samples from inside gradle

The build.gradle file supports 2 tasks for running the samples:

- afsendbesked (which publishes a message)

- afhentbesked (which retrieves one or more messages)

Run like so:

sh ./gradlew afsendbesked

sh ./gradlew afhentbesked

Command-line arguments can be supplied using an argument: -Dexec.args="" passes whatever is in the ""s to the tool

Example:

sh ./gradlew afsendbesked -Dexec.args="-help"

sh ./gradlew afhendbesked -Dexec.args="-help"

You must at least supply keystore and password, CVR number and message for sending.

You must at least supply keystore and password, CVR number and queue UUID for retrieving.

Command line arguments:

|  |  |
| --- | --- |
| Argument | Action |
| -keyfile "PATH\_TO\_File" | keyStoreFile |
| -keypass "Password " | Password to keyStoreFile |
| -trustfile "PATH\_TO\_FILE " | trustStoreFile |
| -trustpass "Password" | Password to trustStoreFile |
| -cvr "CVR" | requestCVRNumber |

Table 1 JAR command line arguments

## Running the JAR executables

Run like so:

java -jar build/jars/afsendbesked-client-1.3.2-SNAPSHOT.jar

java -jar build/jars/afhentbesked-client-1.3.2-SNAPSHOT.jar

Supply parameters as usual, i.e.:

java -jar build/jars/afsendbesked-client-1.3.2-SNAPSHOT.jar -help

All possible arguments are provided by the “-help” argument and are listed in Table 1.

## AfhentBesked.java & AfsendBesked.java using Security Token Service

SfwClient.java shows how to setup a token-request using the performTokenCall method. This example uses the REST API from the Security Token Service.

## RabbitMQ Message Delivery

RabbitMQ (and AMQP) message delivery semantics have been described here in detail:

<https://www.rabbitmq.com/api-guide.html>

<https://www.rabbitmq.com/reliability.html>

<https://www.rabbitmq.com/direct-reply-to.html>

## Wildfly SSL Configuration

In order for a deployed REST application to use SSL, the appropriate configuration must be done on the container (Wildfly, in our example) so that outgoing connections use TLS.

The following documentation indicates how this can be done:

<https://docs.jboss.org/author/pages/viewpage.action?pageId=66322705>

## Web service client security with Apache CXF

We are using the Apache CXF framework in STS for web application security (service providers).

Essential WS-TRUST features are implemented in CXF and the relevant document linked below showhow a web service client can contact and obtain assertions from a Security Token Service, for example

<https://cxf.apache.org/docs/ws-trust.html>

## XML message examples

Valid XML message:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns2:Haendelsesbesked xmlns="urn:oio:sagdok:3.0.0"

xmlns:ns2="urn:oio:besked:kuvert:1.0"

xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"

xmlns:ns4="urn:oio:sts:1.0.0">

Invalid XML message:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<ns2:Haendelsesbesked xmlns="urn:oio:sagdok:3.0.0"

xmlns:ns2="urn:oio:ns2:kuvert:1.0"

xmlns:ns3="http://www.w3.org/2000/09/xmldsig#"

xmlns:ns4="urn:oio:sts:1.0.0">

This message uses an incorrect namespace in line 3. This will result in the following error:

Execution failed for task ':AfsendBesked.main()'.

> Process 'command 'C:/Program Files/Java/jdk1.8.0\_161/bin/java.exe'' finished with non-zero exit value 1

[main] ERROR 2020-03-06 15:03:17,077 [AfsendBesked] - cvc-elt.1: Cannot find the declaration of element 'ns2:Haendelsesbesked'.