Das vorliegende Beispiel zeigt eine sehr einfache Implementierung der XTA2-Schnittstelle, mit dem Sie Nachrichten aus einem XTA2-Postfach abholen können. Dieses Beispiel stellt keine Best-Practice in Bezug auf die Java-Implementierung dar, noch wird jedwede Garantie hinsichtlich Funktionalität und Sicherheit der vorliegenden Implementierung übernommen.

Das Beispiel basiert auf OpenJDK 1.8. Es wird bei den folgenden Schritten davon ausgegangen, dass das JDK im `ressourcen`-Verzeichnis des Basisverzeichnisses des Beispiels abgelegt wurde.

Berücksichtigen Sie, dass die Beispiele von einer UNIX-Shell-Umgebung ausgehen. Wenn Sie dieses Beispiel unter Windows ausführen möchten, müssen Sie die Anweisungen zum Setzen der Umgebungsvariablen bzw. Pfadangaben entsprechend anpassen. Dies gilt auch für die Pfad-Angaben im Class-Path.

1. Legen Sie die Verzeichnisse `build`, `generated`, `ressourcen` und `src` in dem von Ihnen gewählten Basisverzeichnis an.

1) Kopieren Sie die Datei `Xta2Receiver.java` unter `src/de/berlin/bda/xta2/receiver`

2) Kopieren Sie die Dateien `SimpleKeyManager.java` und `XTAServiceManager.java` in den Pfad `src/de/xoev/transport/xta`

3) Laden Sie die XTA2-Schemata https://www.xoev.de/downloads-2316#XTA herunter, das vorliegende Beispiel geht von Version 3: http://xoev.de/transport/xta/211/ aus. Legen Sie das Paket im `ressourcen`-Verzeichnis im Verzeichnis `xta2` ab.

4) Im Beispiel wird zur Generierung des Webservice-Clients das Metro-Framework in der Version 2.3.1 verwendet: https://javaee.github.io/metro/. Laden Sie dieses Framework herunter und packen die ZIP-Datei im `ressourcen`-Verzeichnis aus.

5) Laden Sie die Jar-Datei https://mvnrepository.com/artifact/org.jetbrains/annotations/16.0.1 herunter und legen die Jar-Datei im Verzeichnis `ressourcen` ab.

6) Kopieren Sie die Datei `ressourcen/xta2/XTA.wsdl` nach `src`

7) Legen Sie die Datei `jaxp.properties` mit folgender Zeile unter `ressourcen/JDK-VERZEICHNIS/jre/lib` an:

```
javax.xml.accessExternalDTD = all
javax.xml.accessExternalSchema = all
```

*Alternativ zu dieser `jaxp.properties` Datei können Sie die Argumenten (siehe Schritt 9):
`-Djavax.xml.accessExternalSchema=all`
`-Djavax.xml.accessExternalDTD=all`

8) Setzen Sie die Umgebungsvariable `METRO_HOME` auf `./ressourcen/metro`

```
export METRO_HOME="./ressourcen/metro"
```

9) Erzeugen Sie den Webservice-Client aus der XTA2-WSDL, wechseln Sie hierzu vorher in Ihr Basisverzeichnis:

```
ressourcen/JDK-VERZEICHNIS/bin/java -classpath
"$METRO_HOME/lib/webservices-tools.jar:$METRO_HOME/lib/webservices-
api.jar:$METRO_HOME/lib/webservices-extra-
api.jar:$METRO_HOME/lib/webservices-extra.jar" -Xmx128M
com.sun.tools.ws.WsImport -s generated -extension -B-XautoNameResolution -
XadditionalHeaders -Xnocompile file:./src/XTA.wsdl
```

Hinweis: es werden einige Warnungen ausgegeben, diese können ignoriert werden. Beachten Sie, dass bei der Generierung Elemente aus dem Internet nachgeladen werden.

Wenn Sie die Meldung “Code wird generiert …” erhalten, konnte der Client erfolgreich generiert werden. Im generated-Verzeichnis finden sich die Java-Sourcen des generierten Clients.

10) Editieren Sie die Datei `Xta2Receiver` und setzen die Werte für Keystore und Truststore, ggf. auch die Adresse der VPS-Instanz

```
...
final XTAServiceManager manager = new XTAServiceManager(
    "MY_KEYSTORE.p12", null, "MY_KEYSTORE_PW",
    "MY_TRUSTSTORE.jks", "MY_KEYSTORE_PW”);

// Basisadresse des GCG der DEV-Umgebung
final String xtaBase = "https://cc01lsv2771.srz.lit.verwalt-berlin.de:8444/GovGateway211";
...
```

Ferner müssen die das Konto im GCG in der Anweisung `final String name = "USER";` hinterlegen, den Sie passend zum entsprechenden Schlüssel von Ihrem Ansprechpartner erhalten haben.

11) Kompilieren Sie die Sourcen

Webservice-Client bauen aus den generierten Sourcen der XTA2-WSDL. Wechseln Sie in das von Ihnen gewählte Basisverzeichnis:

```
ressourcen/JDK-VERZEICHNIS/bin/javac -d build -cp
./ressourcen/annotations-16.0.1.jar $(find ./generated | grep .java)
```

Abhol-Client bauen:

```
ressourcen/JDK-VERZEICHNIS/bin/javac -XDignore.symbol.file=true -d build -
cp ./build:./ressourcen/annotations-16.0.1.jar $(find ./src | grep .java)
```

12) Erstellen Sie einen Truststore mit den Root- bzw. Zwischenzertifikaten und setzen das Passwort, welches Sie in der oben angegeben haben. Sie erhalten die entsprechenden Zertifikate von Ihrem Ansprechpartner. Legen Sie den erstellten Truststore im build-Verzeichnis ab:

```
ressourcen/JDK-VERZEICHNIS/bin/keytool -import -alias pca2 -file
certificates/ssl_gcg_dev/ITPCA2.it.verwalt-berlin.de_BerlinPCA2.crt -
keystore truststore.jks

ressourcen/JDK-VERZEICHNIS/bin/keytool -import -alias ca004 -file
certificates/ssl_gcg_dev/ITCA004.it.verwalt-berlin.de_BerlinCA042016.crt -
keystore truststore.jks

ressourcen/JDK-VERZEICHNIS/bin/keytool -import -alias cc01lsv2771 -file
certificates/ssl_gcg_dev/2618_csr_cc01lsv2771.srz.lit.verwalt-
berlin.de.crt -keystore truststore.jks

ressourcen/JDK-VERZEICHNIS/bin/keytool -import -alias governikus -file
certificates/governikus-ssl.cer -keystore truststore.jks
```

13) Legen Sie den Keystore, welchen Sie von Ihrem Ansprechpartner bekommen haben, im build-Verzeichnis ab.

14) Führen Sie die Nachrichtenabholung aus, indem Sie in das `Build`-Verzeichnis wechseln. Beachten Sie, dass Sie sich ggf. Nachrichten zustellen müssen. Sie erhalten einen Test-Link von Ihrem Ansprechpartner.

```
ressourcen/JDK-VERZEICHNIS/bin/bin/java -cp .
de.berlin.bda.xta2.receiver.Xta2Receiver
```