X4 BiPRO Server Benutzerhandbuch

SoftProject

Die in dieser Dokumentation enthaltenen Informationen und die zugehörigen Programme können ohne besondere Ankündigung geändert werden. Für etwaige Fehler übernimmt SoftProject keine Haftung.

Diese Dokumentation und die zugehörigen Programme dürfen ohne schriftliche Zustimmung der SoftProject GmbH weder ganz noch teilweise kopiert, reproduziert, verändert oder in irgendeine elektronische oder maschinenlesbare Form umgewandelt werden.

Alle genannten Warenzeichen sind Warenzeichen der jeweiligen Eigentümer.

Kontakt

SoftProject GmbH

Am Erlengraben 3

D-76275 Ettlingen

Website: www.softproject.de

Vertrieb

Telefon: +49 7243 56175-0

vertrieb@softproject.de

SoftProject-Support

Telefon: +49 7243 56175-333

support@softproject.de

Inhaltsverzeichnis

1 Einleitung 4
2 Über den X4 BiPRO Server 5
2.1 Aufbau des X4 BiPRO Servers 5
2.1.1 Implementierte Sicherheits-Features 5
2.1.2 Aufbau des X4 BiPRO Servers 6
2.1.3 Funktionsweise der X4 BiPRO Services 7
2.2 Implementierungen 8
2.2.1 Grundlagen 8
2.2.2 Namespaces 9
2.2.3 Service Konfiguration (Service.xml) 9
2.2.4 Statische Validierung 17
2.2.5 Definition von BiPRO-Diensten 20
2.2.6 Prozessschritte 24
2.2.7 Anhänge und Optimierung mit MTOM 30
2.3 Backup 31
3 Entwickeln mit dem X4 BiPRO Server 32
3.1 Neues Projekt anlegen 32
3.1.1 Übersicht über die verschiedenen Vorlagenarten 33
3.2 Module 38
3.2.1 Modul BiPRO Norm 410 Security Token Service 38
3.2.2 Modul BiPRO Norm 410 Security Token Service (Legacy) 66
3.2.3 Modul BiPRO Norm 430 Transfer Service 78
3.2.4 Modul X4 BiPRO Import (Demo Projekt) 88
3.2.5 Modul X4 BiPRO Consumer Services 90

1 Einleitung

Kenntnisse über die grundsätzliche Entwicklung von ESB-Prozessen mit der X4 BPMS werden für die Prozessentwicklung mit dem X4 BiPRO Server vorausgesetzt!

Mit dem X4 BiPRO Server werden neben einer Systembibliothek auch die spezifischen Module in Form von X4-Templates ausgeliefert.

Diese Templates können sowohl als Implementierungsvorlagen als auch als eigenständige Module verwendet werden, um verschiedene BiPRO-Services zu implementieren. Sie bilden die Schnittstelle zum Produktkern und enthalten Beispiele, Demos und ReadMe-Dateien zur Arbeit mit dem X4 BiPRO Server.

Die Templates sind nach Erstellung voll editier- und veränderbar, so dass jede Komponente an die jeweiligen Projektvorgaben angepasst werden kann.
Beachten Sie, dass ein reguläres Update eines solchen aus einem X4 Template entstandenen Projektes nicht mehr möglich ist, da jede Implementierung auf Basis eines X4 Templates einzigartig ist.
Aus diesem Grund empfiehlt es sich, das jeweilige X4 Template als Basis zu nehmen und die benötigten Schnittstellenprozesse im Template über ein separates X4 Projekt anzusteuern. Dadurch kann bei jedem Update des X4 BiPRO Servers das jeweilige X4 Template einfach neu erstellt werden.
Für kundenseitig ausgeführte Upgrades kann SoftProject keine Gewähr übernehmen.

2 Über den X4 BiPRO Server

2.1 Aufbau des X4 BiPRO Servers

Der BiPRO Server stellt dem Prozessentwickler ein System zur Verfügung, mit dem BiPRO-Services deklarativ definiert und mit eigenen Prozessen verknüpft werden können. So sind nach außen angebotene BiPRO-Dienste immer BiPRO-konform, während alle Freiheiten der X4 Suite genutzt werden können, um die Dienste zu implementieren.

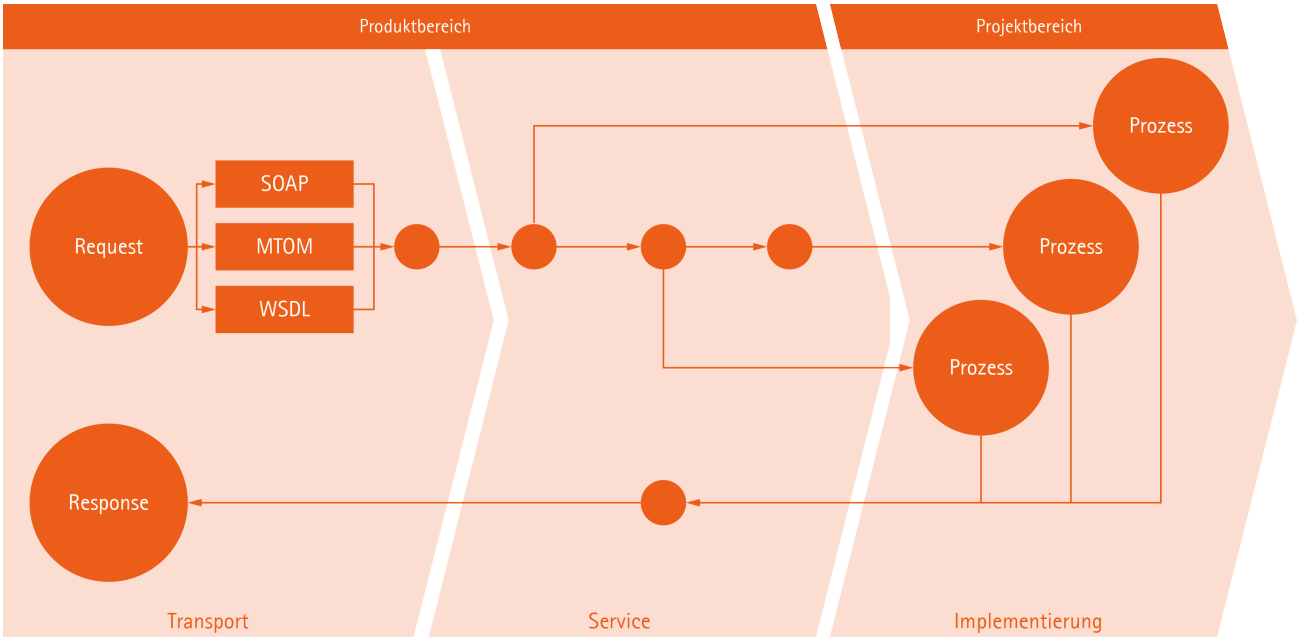
Der X4 BiPRO Server steuert die gesamte elektronische Kommunikation zwischen Versicherer und Vertriebspartnern und ermöglicht dabei, die fachliche Logik und Funktionalität auf Basis der BiPRO-Normen schnell umzusetzen. Basis hierfür ist eine Sammlung von Prozessen, Mappings und Steuerdateien auf Basis des X4 ESB.

2.1.1 Implementierte Sicherheits-Features

Der X4 BiPRO Server unterstützt folgende Sicherheitsvorgaben des BiPRO e. V.:

- Security Token Service zur Authentifizierung der Service Consumer für die Nutzung der Business Services (User+Password im Standard vorhanden; X509-Zertifikate und VDG-Tickets können sehr leicht kundenspezifisch implementiert werden)
- Verschlüsselung auf Transport-Ebene über SSL
- Authentifizierung und Integritätsprüfung auf Nachrichten-Ebene über Signaturen
- Umsetzung des Webservice Security Standards
- Validierung der zu transportierenden Nachrichten gegen die Schema-Definitionen des BiPRO e. V.

2.1.2 Aufbau des X4 BiPRO Servers



Die Anmeldedienste der *Normen 410 und 411* für Benutzername und Passwort, Zertifikate und VDG-Tickets stehen direkt nach der Installation als X4 Template zu Verfügung.
Der Transfer-Service der *Norm 430* sowie die Schnittstellen zum Import von Transfers und deren Bündelung als Lieferung stehen ebenfalls direkt nach der Installation als X4 Template zur Verfügung.

Der X4 BiPRO Server ist in drei Schichten eingeteilt. Jede Schicht unterliegt einem unterschiedlichen Zugriff und unterschiedlichen Zugriffsmöglichkeiten:

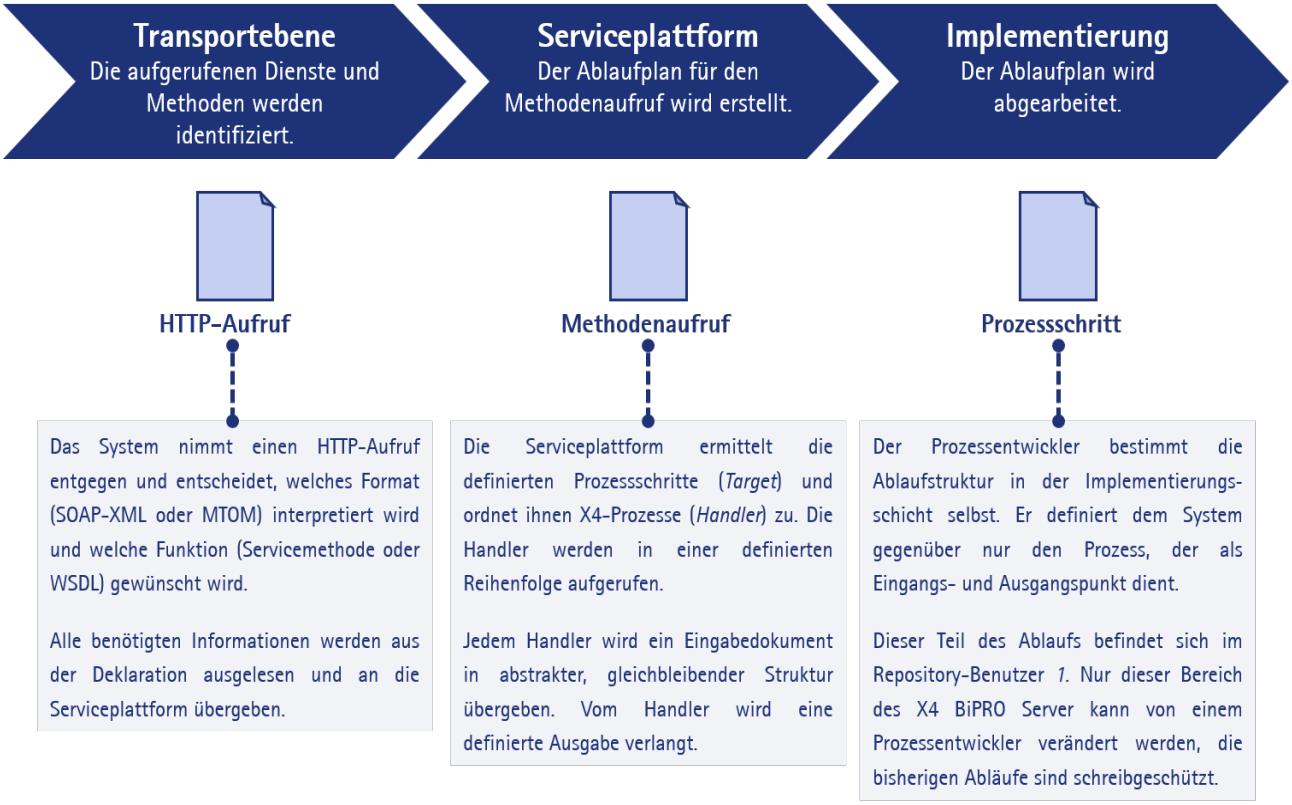
- **Transport-Schicht** (Produktbereich)
  Diese Schicht beinhaltet die Request-Behandlung (SOAP, MTOM, WSDL).
- **Service-Schicht** (Produktbereich)
  Diese Schicht beinhaltet die Grundgerüste der verschiedenen Services.
- **Implementierungs-Schicht** (Projektbereich, siehe Implementierungen)
  Diese Schicht beinhaltet die vom Prozessentwickler implementierten Prozesse. Diese Schicht enthält die *projektspezifischen Implementierungen* einzelner Normen.

Sowohl die Transport-Schicht als auch die Service-Schicht gehören zum **Produktkern**. Dieser befindet sich im Projekt `X4BiPROSystem`.

Die Implementierungs-Schicht bezeichnet den allgemeinen **öffentlichen Projektbereich** im Repository-User `1` . Er bildet sich aus den verschiedenen projektspezifischen Implementierungen.

Durch diese Aufteilung wird nicht nur der Update-Vorgang vereinfacht, sondern auch der der Fokus auf die kundenspezifischen Implementierungen der BiPRO Services gelegt.

2.1.3 Funktionsweise der X4 BiPRO Services



2.1.3.1 Vorverarbeitung

Das System versucht festzulegen, um welche Art von Aufruf es sich handelt. Dazu wird der HTTP-Aufruf, der vom ReST-Starter an den Prozess übergeben wird, nach folgender Entscheidungstabelle klassifiziert:

Content-Type Body → **HTTP-Methode ↓**	**application/xml** **text/xml**	**multipart/related**	**Sonstige**
POST	SOAP	MTOM	Fehler
GET	WSDL		
Sonstige	Fehler		

Wenn es sich um einen SOAP- oder MTOM-Request handelt, dann wird der Aufruf an die Serviceplattform weitergeleitet. Wenn es sich um einen MTOM-Request handelt, dann wird der Request zuvor dekodiert, indem die enthaltenen Anhänge in die SOAP-XML-Struktur integriert werden. Wenn ein WSDL-Request erkannt wird, dann wird der WSDL-Generator statt der Serviceplattform aufgerufen.

Im Fehlerfall antwortet der Dienst mit einer entsprechenden Fehlermeldung in Form eines SOAP-Fehlers.

2.1.3.2 Behandelnden Prozess selektieren

Wenn der Request an die Serviceplattform weitergeleitet wird, dann wird der Request mit dem Aufrufkontext angereichert. Der Aufrufkontext besteht zunächst aus einer Interpretation der URL nach einem bestimmten Muster. Dieses Muster wird aus den Vorgaben des BiPRO e. V. und der Basis-URL für den ReST-Starter gebildet.

1	Domain
2	Präfix
3	BiPRO-Norm (*Group*)
4	Service (*Endpoint name*)
5	Normversion ⓘ Entsprechend der BiPRO-Norm besteht die Normversion immer aus 5 Blöcken, z. B. `2.5.0.1.0`.

2.1.3.3 Service-Konfiguration

Der X4 BiPRO Server stellt im Repository-Benutzer *1* unter **X4BiPRO/Resources/Service.xml** einen zentralen Einstiegspunkt bereit. Wenn diese Datei existiert, dann werden alle dort angegebenen Service-Definitionen genutzt, um den verarbeitenden Service zu identifizieren.

Wenn aus dem Kontext keine Service-Definition abgeleitet werden kann, dann antwortet der Dienst mit einer Fehlermeldung.

Wenn nur keine passende Methodendefinition ermittelt werden kann, dann entspricht die Antwort BiPRO-konform der Fehlermeldung `00009` ("Methode nicht implementiert").

2.2 Implementierungen

2.2.1 Grundlagen

Für Serviceimplementierungen steht ein einfaches Konfigurationsmodell zur Verfügung. Dieses Konfigurationsmodell erlaubt die Integration von Prozessen in das System.

In diesem Konfigurationsmodell werden Dienstaktionen in drei Ebenen eingeteilt:

- Gruppe

- Endpunkt
- Implementierung

Gruppe und Endpunkt bestimmen jeweils eine Dienst-URL. Die Implementierung benennt eine Aktion oder Methode innerhalb eines Dienstes. Der Endpunkt hat nicht nur einen Namen, sondern auch eine Versionsnummer mit fünf Ziffern (vgl. BiPRO-Norm 190).

Beispiel

- Gruppe: 421_TAASUH
- Endpunkt: Service
- Endpunktversion: 2.6.1.1.0

Die URL wird anhand obiger Informationen bestimmt: *Basis-URL*/BiPRO/421_TAASUH/Service_2.6.1.1.0

Die Implementierung *getQuote* ergänzt den Aufrufpfad mit einer Servicemethode.

Weitere Informationen unter Definition von BiPRO-Diensten.

2.2.2 Namespaces

Die zentralisierte Servicemethode `ServiceLevelEntryPoint` bildet den Ausgangspunkt für eine projektspezifische Implementierung. Das Definitionsmodell für BiPRO-Services verwendet die folgenden Namespaces mit den genannten Präfixen:

Name	Präfix	Namespace	Beschreibung
BiPRO	`sp`	http://www.softproject.de/schemas/2018/x4bipro	Dieser Namespace beinhaltet die BiPRO-Standard-Namespaces.
Exception	`e`	http://www.softproject.de/schemas/2018/exception	Dieser Namespace beinhaltet die Exception-Schemas.

2.2.3 Service Konfiguration (Service.xml)

Die Datei `Service.xml` bildet eine übergeordnete Konfigurationsdatei im jeweiligen Ordner `Resources`.

In dieser Datei wird definiert, wo die Prozesse liegen, die beim Service-Aufruf verwendet werden sollen.

Beispiel für eine Service.xml

```
<Component>
    <Group id="..." standard="...">
        <Endpoint id="..." version="..." href="#...">
            <Component href="..." />
        </Endpoint>

        <Endpoint ...>
            <Component href="..." />
        </Endpoint>
        ...
    </Group>

    <EndpointComponent id="...">
        <Namespace prefix="..." uri="..." targetNamespace="true" />
        <Namespace prefix="..." uri="..." />
        ...
        <Schema xmlns:xsd="...">
            <xsd:import namespace="..." schemaLocation="... .xsd" />
        </Schema>
        <Implementation operation="..." href="#..." />
        ...
    </EndpointComponent>
    <ImplementationComponent id="...">
        <InputMessage name="..." type="..." />
        <OutputMessage name="..." type="..." />
        <Handler event="...">
            <Workflow href="..." />
        </Handler>
    </ImplementationComponent>
    ...
</Component>
```

2.2.3.1 Element <Component>

Attribut	Beschreibung
`href`	Policy-Verweis. xstore-Verlinkung zur Policy-Konfiguration. **Mögliche Werte:** String (URL)

2.2.3.2 Element <Group> (Dienstgruppe)

Das Group-Element fasst ein oder mehrere Endpoint-Elemente zusammen und bildet den implementierten Service wie z.B. den Security Token Service oder den Transport Service ab.

Innerhalb des Group-Elements sind alle Endpoints, also Ausprägungen des Services zusammengefasst, z. B. *x509* bzw. *User/Password-Login* für den Security Token Service.

Beispiel Group-Element

```
<Component>
    <Group id="default-sts" standard="410">
        <Endpoint>
            ...
        </Endpoint>
        ...
    </Group>
    ...
</Component>
```

Attribut	Beschreibung
`id`	Identifikator der Dienstgruppe **Mögliche Werte:** String
`standard`	BiPRO-Standard-Norm, die in der Dienstgruppe verwendet wird. **Mögliche Werte:** String

2.2.3.3 Element <Endpoint> (Endpunkt)

Das `Endpoint`-Element definiert den jeweiligen Endpunkt des Services. Das Group-Element enthält alle verwendeten Endpunkte. Die Endpunkte werden mit Verweisen zu den jeweiligen Policies versehen. Über den Wert des Attributs `id` erfolgt die Identifikation des Endpunktes. Das Attribut `version` definiert die verwendete BiPRO-Version. Die Endpoint-Elemente enthalten ein Component-Element mit einem Verweis auf die dazugehörigen Policy-Dateien.

Beispiel Endpoint-Element

```
<Component>
    <Group>
        <Endpoint id="UserPasswordLogin" version="2.5.0" href="#default-
sts:endpoint">
            <Component href="x4db://X4BiPRO/SecurityTokenService/Configuration/
Policies/UserPasswordLogin.xml" />
        </Endpoint>
        <Endpoint id="X509Login" version="2.5.0" href="#default-sts:endpoint">
            <Component href="x4db://X4BiPRO/SecurityTokenService/Configuration/
Policies/X509Login.xml" />
        </Endpoint>
        <Endpoint id="..." version="..." href="...">
            <Component href="..." />
        </Endpoint>
        ...
    </Group>


    <EndpointComponent id="default-sts:endpoint">
        <Namespace prefix="bipro" uri="http://www.bipro.net/namespace"
targetNamespace="true" />
        <Namespace prefix="wst" uri="http://schemas.xmlsoap.org/ws/2005/02/trust" />
        <Namespace prefix="wsse" uri="http://schemas.xmlsoap.org/ws/2002/12/secext" /
>
        <Schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://schemas.xmlsoap.org/ws/2005/02/trust"
schemaLocation="http://schemas.xmlsoap.org/ws/2005/02/trust/WS-Trust.xsd" />
        </Schema>
        <Export name="MySTS" url="http://www.provider.de/STS" />
        <Implementation operation="RequestSecurityToken" href="#default-
sts:requestSecurityToken" />
        <!-- <Implementation operation="ValidateSecurityToken" href="#default-
sts:validateSecurityToken" /> -->
    </EndpointComponent>
    ...
</Component>
```

Attribut	Beschreibung
id	Identifikator des Endpunktes **Mögliche Werte:** String

Attribut	Beschreibung
`version`	Implementierte BiPRO-Version, z. B. 2.5.0 **Mögliche Werte:** String ⓘ Die Versionsnummer kann für die 2019-04-12_14-49-18_WSDL-Generierung auf verschiedene Weise angegeben werden. Es können einstellige, zweistellige, dreistellige usw. Nummern angegeben werden. Beim Aufrufkontext wird immer die vollqualifizierte BiPRO-Version mit 5 Ziffernblöcken als Versionsnummer verwendet.
`href`	Verweis auf dazugehörige Endpunkt-Komponenten (`<EndpointComponent>`) mit Implementierungsverweisen **Mögliche Werte:** String (URL) ⓘ Vor die URL muss ein # gesetzt werden.
`forceSoap11`	Festlegung, ob das WSDL-Dokument SOAP 1.1 statt SOAP 1.2 als unterstütztes Protokoll ausweisen soll. **Mögliche Werte:** `true` / `false` ⓘ Es werden unabhängig von dieser Einstellung beide Versionen unterstützt. Die Angabe bezieht sich lediglich auf die Ausweisung im WSDL-Dokument.

2.2.3.4 Element <EndpointComponent>

In den einzelnen Endpunkt-Komponenten wird der Verweis auf die Implementierung definiert. Dazu wird der Namespace, die verwendeten Schemas und der Verweis auf die jeweilige Implementierungskomponente mit den entsprechenden Methodennamen (Operation) gesetzt. Das `EndpointComponent`-Element enthält die Elemente `Namespace`, `Schema` und `Implementation`.

Attribut	Beschreibung
`id`	Identifikator, der im Attribut `href` des Elements `<Endpoint>` angegeben werden muss. **Mögliche Werte:** String

2.2.3.5 Element <Namespace>

Das `Namespace`-Element definiert den verwendeten Namespace. Es kann innerhalb des Elements `EndpointComponent` mehrere Namespace-Elemente geben.

Attribut	Beschreibung
`prefix`	Verwendeter Namespace-Präfix **Mögliche Werte:** String
`uri`	URI des Namespace **Mögliche Werte:** String (URL)
`targetNamespace`	Definiert, ob der Namespace der Target Namespace ist. **Mögliche Werte:** `true` / `false`

2.2.3.6 Element <Schema>

Das `Schema`-Element definiert die verwendete Schema-Datei. Es kann innerhalb des Elements `EndpointComponent` mehrere `Schema`-Elemente geben. Das `Schema`-Element enthält das Element `<xsd:import>`.

Attribut	Beschreibung
`namespace`	Namespace unter dem das Schema angesprochen wird. **Mögliche Werte:** String
`schemaLocation`	URL der dazugehörigen XSD-Datei. **Mögliche Werte:** String (URL)

2.2.3.7 Element <Export>

Das `Export`-Element kann genutzt werden, um die Namen und/oder URLs innerhalb des WSDL-Dokuments zu beeinflussen.

Attribut	Beschreibung
name	Der Name, der für die Elemente <wsdl:portType>, <wsdl:port>, wsdl:binding und <wsdl:service> genutzt wird. An folgenden Stellen wird der Wert dieses Attributs genutzt: • <wsdl:portType name="[WERT]PortType"> • <wsdl:binding name="[WERT]Binding" type="bipro:[WERT]PortType"> • <wsdl:service name="[WERT]"> <wsdl:port name="[WERT]Port" binding="bipro:[WERT]Binding">
url	Die externe URL, unter der der Service erreichbar ist (sofern die im X4 BiPRO Manager festlegbaren Werte für die externe URL nicht ausreichen)

2.2.3.8 Element <Implementation>

Das Implementation-Element definiert die Servicemethode eines Endpunktes. Für jede Implementierung muss ein Implementation-Element angelegt werden. Dieses Implementation-Element enthält die Messages und die Handler mit Verweisen auf den dazugehörigen Prozess (*.wrf).

Attribut	Beschreibung
operation	verwendete Servicemethode

Beispiel Implementation-Element

```
<Component>
    ...
    <Implementation id="default-sts:requestSecurityToken">
        <InputMessage name="RequestSecurityTokenRequest" type="wst:RequestSecurityTok
en" />
        <OutputMessage name="RequestSecurityTokenResponse" type="wst:RequestSecurityT
okenResponse" />
        <Handler event="InboundSchemaValidation">
            <Workflow href="x4db://X4BiPRO/SecurityTokenService/Methods/
RequestSecurityToken/InboundSchemaValidation.wrf" />
        </Handler>
        <Handler event="ServiceImplementation">
            <Workflow href="x4db://X4BiPRO/SecurityTokenService/Methods/
RequestSecurityToken/MethodImplementation.wrf" />
        </Handler>
    </Implementation>
    <Implementation id="...">
        <!-- Implementation Component Content -->
    /Implementation>
    ...
</Component
```

2.2.3.9 Element <InputMessage>

Das `InputMessage`-Element definiert die Nachricht, die beim Input ausgegeben wird.

Attribut	Beschreibung
`name`	Bezeichnung der Input-Message. **Mögliche Werte:** String
`type`	Typ der Message **Mögliche Werte:** String

2.2.3.10 Element <OutputMessage>

Das `OutputMessage`-Element definiert die Nachricht, die beim Output ausgegeben wird.

Attribut	Beschreibung
`name`	Bezeichnung der Output-Message. **Mögliche Werte:** String
`type`	Typ der Message **Mögliche Werte:** String

2.2.3.11 Element <Handler>

Das `Handler`-Element bestimmt die Verweise auf aufzurufende Prozessschritte über das enthaltene Element `<Workflow>`.

Attribut	Beschreibung
`event`	Name des aufzurufenden Prozessschrittes (Event) **Mögliche Werte:** • Tracing • SecurityValidation • PolicyValidation • ServiceImplementation

2.2.3.12 Element <Workflow>

Das `Workflow`-Element bestimmt den aufgerufenen Prozessschritt.

Attribut	Beschreibung
`href`	URL des aufzurufenden Prozesses (`*.wrf`) **Mögliche Werte:** String (URL)

2.2.3.13 Element <Target>

Das `Target`-Element bestimmt die Ziel-URL des aufgerufenen Events.

Attribut	Beschreibung
`href`	Ziel-URL des Events **Mögliche Werte:** String (URL)
`state`	Status des Events **Mögliche Werte:** String

2.2.4 Statische Validierung

Manche BiPRO Services (insbesondere 42x TAA) erfordern ein hohes Maß an Performance und schnellen Reaktionszeiten.

Aus diesem Grund bietet der BiPRO Server die Möglichkeit, zur Optimierung der BiPRO Schema-Validierung ein Pre-Loading zu aktivieren, um den Einfluss der Validierung bei jedem Serviceaufruf zu minimieren.

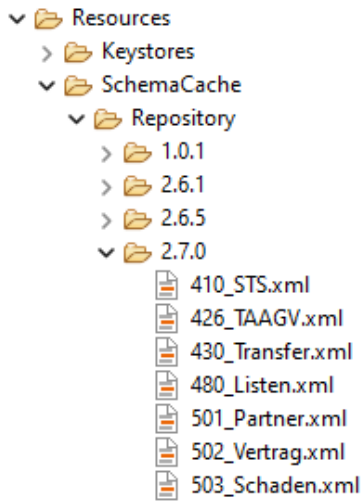
Die Aktivierung nimmt etwas Zeit zu jedem initialen Start des Servers in Anspruch, und je nach BiPRO-Norm und Version ist eine zusätzliche Konfiguration erforderlich

1. Um die statische Validierung zu verwenden, navigieren Sie zum Verzeichnis *X4BiPRO/Resources/SchemaCache* und öffnen die Datei **NormVersions.xml**.
2. Aktivieren Sie die jeweilige BiPRO Release Version, für die die Validierung gelten soll, durch Entfernen des jeweiligen Kommentars.

```
<?xml version="1.0" encoding="UTF-8"?>
<NormVersions>
    <Version>1.0.1</Version>
    <Version>2.6.1</Version>
<!--    <Version>2.6.5</Version> -->
    <Version>2.7.0</Version>
    <Version>2.7.1</Version>
    <Version>2.7.4</Version>
    <Version>2.8.0</Version>
<!--    <Version>2.8.1</Version> -->
<!--    <Version>2.8.2</Version> -->
<!--    <Version>2.8.3</Version> -->
<!--    <Version>2.8.4</Version> -->
</NormVersions>
<!--Created by X4 Designer, Copyright © SoftProject GmbH. All rights reserved.-->
```

Bei Bedarf können auch weitere Release Versionen hinzugefügt werden.

1. Legen Sie einen neuen Ordner im Verzeichnis *X4BiPRO/Resources/SchemaCache/Repository* an und benennen ihn nach der gewünschten Release Version (z.B. 2.7.0). Innerhalb dieses Ordners müssen sich die selben Dateien befinden, wie in den bereits vorhandenen Ordnern (z.B. 410_STS.xml, 426_TAAGV.xml, etc.).



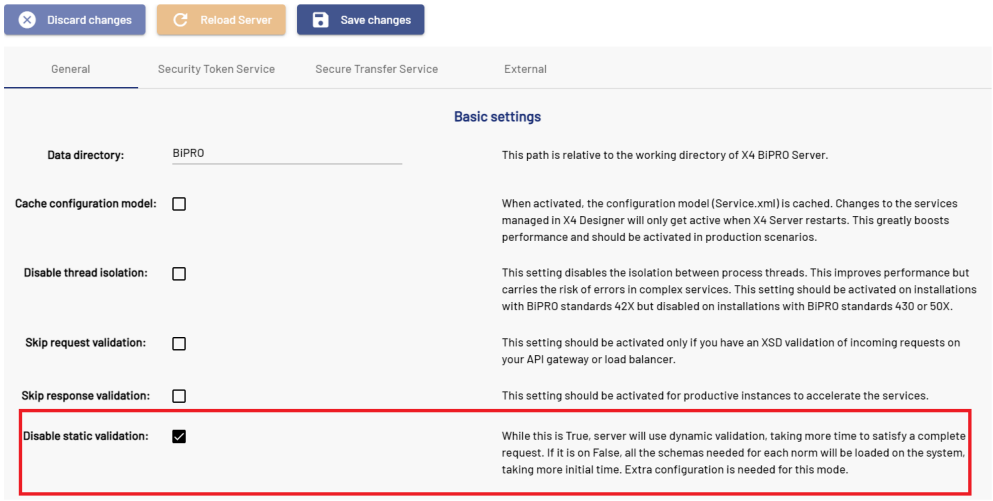
- 42x Normen haben mehrere unterschiedliche Suffixe, je nach Spezialisierung: 421_TAA SUH, 422_TAALV, 423_TAAKF, 424_TAAKV, 425_TAARS, 426_TAAGV. Daher muss beachtet werden, dass alle Attribute der jeweiligen Spezialisierung verwendet werden.

```
<sp:Bindings>
    <sp:EndpointBinding endpoint="Service"
                        group="426 TAAGV"
                        pattern="2.7.0"
                        version="2.7.0.1.0"/>
    <sp:SchemaBinding basePath="x4db://1/X4BiPRO/Resources/Schemas/2.7.(
    <sp:ContextBinding id="426"/>
    <sp:ImplementationBinding allowMutateResponse="true" operation="get(
```

  Die drei rot markierten Parameter werden verwendet, um eine MD5-Kennung zu erstellen, die als Schlüssel für den Zugriff auf die Gruppe von Schemas für diese spezifische Norm dient.
  Der Name der Datei, die hier definierten Attribute und auch die **Service.xml** im X4BiPRO-Ordner müssen übereinstimmen, um korrekt zu funktionieren.
  Im Abschnitt "Schema" können Sie weitere Schemas hinzufügen, wenn die jeweilige Norm dies erfordert.

```
<Schema xmlns="http://www.softproject.de/schemas/2018/x4bipro">
   <xsd:import namespace="http://www.bipro.net/namespace/nachrichten" schemaLocation="${schema.repositoryRoot}/bipro-nachrichten-${context.r
      <xsd:import namespace="http://www.bipro.net/namespace/allgemein" schemaLocation="${schema.repositoryRoot}/bipro-allgemein-${context.re
      <xsd:import namespace="http://www.bipro.net/namespace/versicherung/tarifierung" schemaLocation="${schema.repositoryRoot}/bipro-versich

      <!-- TODO: Include more schemas, depending on which BiPRO standard (421, 422, 423, ...) should be used -->
      <xsd:import namespace="http://www.bipro.net/namespace/basis" schemaLocation="${schema.repositoryRoot}/bipro-basis-${context.release.ba
      <xsd:import namespace="http://www.bipro.net/namespace/datentypen" schemaLocation="${schema.repositoryRoot}/bipro-datentypen-${context.
      <xsd:import namespace="http://www.bipro.net/namespace/partner" schemaLocation="${schema.repositoryRoot}/bipro-partner-${context.releas

      <xsd:import namespace="http://www.bipro.net/namespace/sachen" schemaLocation="${schema.repositoryRoot}/bipro-sachen-${context.release.
      <xsd:import namespace="http://www.bipro.net/namespace/versicherung/produktmodell/komposit" schemaLocation="${schema.repositoryRoot}/bi

      <xsd:import namespace="http://www.bipro.net/namespace/versicherung/haftpflicht" schemaLocation="${schema.repositoryRoot}/bipro-versich
      <xsd:import namespace="http://www.bipro.net/namespace/versicherung/sach" schemaLocation="${schema.repositoryRoot}/bipro-versicherung-s


      <!-- xsd:import namespace="http://www.bipro.net/namespace/sepa" schemaLocation="${schema.repositoryRoot}/bipro-sepa-${context.release}
      <xsd:import namespace="http://www.bipro.net/namespace/sepa" schemaLocation="${schema.repositoryRoot}/bipro-sepa-${context.release}.xsd


      <!-- Hier werden die Provider-spezifischen Erweiterungen (unter Provider-spezifischem Namespace) importiert. -->
         <xsd:import namespace="http://www.provider.de/namespace" schemaLocation="${schema.repositoryRoot}/provider-gewerbe-1.1.xsd" /> -->
</Schema>
```

2. Starten Sie nun den Prozess **RunStartupTasks.wrf** im Verzeichnis X4BiPROSystem/Processes/System, um das Pre Loading nun zu aktivieren.
   Dies kann entweder manuell oder durch Neustart des X4 Servers vorgenommen werden.
   Des weiteren ist auch eine Aktivierung bzw. Deaktivierung über den BiPRO Manager möglich (Flag **Statische Validierung deaktivieren**):



2.2.5 Definition von BiPRO-Diensten

Ein BiPRO-Dienst wird in der Datei `X4BiPRO/Resources/Service.xml` definiert und registriert. In diesem Abschnitt befinden sich einige Beispiele für Dienstdefinitionen mit zunehmender Komplexität.

2.2.5.1 Minimaldefinition

```
<Component>
    <!-- http://[...]/BiPRO/900_Beispiel/[...] -->
    <Group id="900_Beispiel">
        <!-- http://[...]/BiPRO/900_Beispiel/MeinService_2.5.0[.*.*] -->
        <Endpoint id="MeinService" version="1.0.0">
            <Implementation operation="GetWerte">
                <!-- ... -->
            </Implementation>
            <Implementation operation="GetNochMehrWerte">
                <!-- ... -->
            </Implementation>
        </Endpoint>
    </Group>
</Component>
```

Obiges Beispiel ist mindestens notwendig, um einen Dienst in Betrieb zu nehmen und eine Antwort von der URL zu erhalten.

In diesem Beispiel ist die Gruppe `900_Beispiel` und der Endpunkt `MeinService` mit dem Versionsnummernbereich `2.5.0` zu sehen.

> ⓘ Die nachfolgenden Versionsnummerstellen sind in dieser Definition beliebig. Dieser Dienst behandelt alle Aufrufe auf URLs, deren Endpunktversion mit `2.5.0` beginnt.

2.2.5.2 Beschreibung des Dienstschemas

Um das Dienstschema mithilfe der eingebauten WSDL-Generierung zu beschreiben, können am Endpunkt weitere Elemente genutzt werden.

```
<Component>
    <Group id="900_Beispiel">
        <Endpoint id="MeinService" version="2.5.0">


            <!-- Target-Namespace definieren -->
            <Namespace prefix="bipro" uri="http://www.bipro.net/namespace"
targetNamespace="true" />


            <!-- XSD-Schema definieren -->
            <Schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="htt
p://www.bipro.net/namespace" elementFormDefault="qualified" attributeFormDefault="qua
lified">
                <xsd:import namespace="http://www.bipro.net/namespace/nachrichten"
schemaLocation="${context.baseUrl}/${context.schemaPath}/${context.release}/bipro-
nachrichten-${context.release.base}.xsd" />
                <xsd:import namespace="http://www.bipro.net/namespace/allgemein"
schemaLocation="${context.baseUrl}/${context.schemaPath}/${context.release}/bipro-
allgemein-${context.release.base}.xsd"/>
                <!-- ... -->
            </Schema>
            <!-- ... -->
        </Endpoint>
    </Group>
</Component>
```

In diesem Beispiel erhält der Dienst ein Target-Namespace, in dem alle Schema-Elemente liegen. Zusätzlich wird ein Schema definiert, dass die BiPRO-Namespaces `Allgemein` und `Nachrichten` importiert.

Eine Beschreibung der im Attribut `schemaLocation` verwendeten Platzhalter findet sich in Platzhalter in schemaLocation.

2.2.5.3 Platzhalter in schemaLocation

Im Attribut `schemaLocation` können Platzhalter verwendet werden. Diese zeigen auf die entsprechende Schemadatei im Schemapaket der BiPRO-Version, für die dieser Endpunkt den Dienst bereitstellt. Folgende Platzhalter können verwendet werden:

Platzhalter	Beschreibung
`context.baseUrl`	Externe Basis-URL (aus Reverse-Proxy-Konfiguration) **Beispiel:** http://www.versicherung.de
`context.servicePath`	Externer Basis-Pfad der BiPRO-Services (aus Reverse-Proxy-Konfiguration) **Beispiel:** /service/bipro
`context.schemaPath`	Externer Basis-Pfad der BiPRO-Schemas (aus Reverse-Proxy-Konfiguration) **Beispiel:** /schemas/bipro
`context.endpointUrl`	Externe URL zum aktuellen Endpunkt **Beispiel:** http://www.versicherung.de/service/bipro/430_Transfer/Service_2.6.1.1.0
`context.endpointName`	ID des aktuellen Endpunkts **Beispiel:** Service
`context.endpointFullName`	ID und vollqualifizierte Version (5-stellig) des aktuellen Endpunkts getrennt mit einem Unterstrich **Beispiel:** Service_2.6.1.1.0
`context.endpointVersion`	Vollqualifizierte Version (5-stellig) des aktuellen Endpunkts **Beispiel:** 2.6.1.1.0
`context.release`	BiPRO-Release-Version (3-stellig) des aktuellen Endpunkts **Beispiel:** 2.6.1
`context.release.base`	BiPRO-Release-Version (3-stellig) des aktuellen Endpunkts; die letzte Stelle ist immer `0`. **Beispiel:** 2.6.0

Platzhalter	Beschreibung
`schema.repositoryRoot`	Der Basispfad der Schemadateien, die für das im Serviceaufruf geltende Release herangezogen werden. Im Kontext einer Validierung beginnt die URL mit "xstore://" für bekannte Schemadateien. Im Kontext der Servicebeschreibung (WSDL) beginnt die URL immer mit "http://"

2.2.5.4 Implementierungsparameter

Mit Hilfe des `ImplementationParameter`-Elements können dem Serviceprozess weitere Informationen mitgeteilt werden. Das ist beispielsweise dann hilfreich, wenn ein Prozess für mehrere Einträge in der Servicedefinition verwendet wird. Konkret wird in der Standardimplementierung der Norm 410/411 derselbe Prozess für die Behandlung der Anfragen verwendet. Jedoch legt ein Implementierungsparameter "tokenType" die Art des Security-Tokens fest, welches der Prozess erzeugt.

X4 BiPRO Server nutzt einige vordefinierte Parameter, um das Verhalten der Serviceinfrastruktur zu steuern. Nachfolgend befindet sich eine Auflistung dieser vordefinierten Parameter.

Parameter	Beschreibung
responseType	Mit diesem Parameter kann der Inhalt des Content-Type-Headers festgelegt werden, der in Antworten des jeweiligen Services enthalten ist.
doNamespaceOptimization	Durch die Festlegung auf *true* kann die aktive Optimierung der XML-Namespaces in den SOAP-Antworten aktiviert werden.

Beispiel

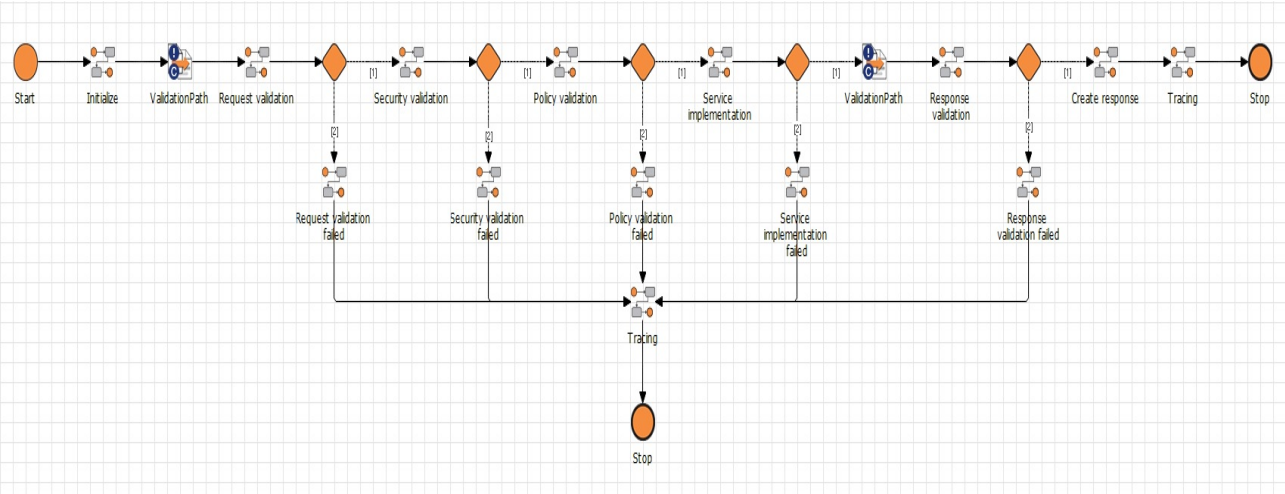
In diesem Beispiel wird ein registrierter BiPRO Norm 430 Transfer Service in der Service.xml um den Implementierungsparameter *doNamespaceOptimization* erweitert.

Dies hat zur Folge, dass in jeder Service-Response die Namespaces im Root-Element deklariert werden (dies ist jedoch nicht BiPRO-konform!):

```
...
    <Component href="xstore://430_Transfer/Resources/Service.xml" />
    <Group id="430_Transfer">
        <ImplementationParameter name="doNamespaceOptimization">true</
ImplementationParameter>

        <!-- Activates MTOM -->
        <Optimization behavior="default">
            <BinaryContainer local-name="Daten" namespace-uri="http://
www.bipro.net/namespace/allgemein" />
        </Optimization>

        <Endpoint id="Service" version="2.6.1" href="#430_Transfer" />
        <Endpoint id="Service" version="2.7.1" href="#430_Transfer" />
        <Endpoint id="Service" version="2.8.0" href="#430_Transfer" />
    </Group>
...
```

2.2.6 Prozessschritte

Die Struktur der internen Systemprozesse:



2.2.6.1 Sicherheitsprüfung (SecurityValidation)

> **Die Sicherheitsprüfung wird durch automatische Vorgänge ergänzt**
> Das übergebene BiPRO- oder SAML-Token wird standardmäßig zusätzlich intern geprüft. Wenn die zusätzliche Prüfung nicht durchgeführt werden soll, dann kann sie im Bereich *Security Token Service* im X4 BiPRO Manager abgeschaltet werden, indem die Einstellung **Standard-Validierung abschalten** verwendet wird.

2.2.6.1.1 Eingabestruktur

Eingabestruktur

```
<Request>
    <Service>
        <Name><!-- Endpunkt-Name --></Name>
        <Context><!-- BiPRO-Norm --></Context>
        <Version><!-- BiPRO-Version --></Version>
    </Service>
    <CorrelationId>
        <!-- Eindeutiger Bezeichner (PID) des Serviceaufrufes -->
    </CorrelationId>
    <Timestamp>
        <!-- Timestamp bei Request-Eingang -->
    </Timestamp>
    <Parameters>
        <!-- Implementation-Parameter aus der Servicekonfiguration -->
    </Parameters>
    <Stages>
        <!-- Ergebnisse der vorangegangenen Prozessschritte -->
    </Stages>
    <Headers>
        <!-- Nachrichtenheader -->
    </Headers>
    <Body>
        <!-- Nachrichtenbody -->
    </Body>
</Request>
```

2.2.6.1.2 Ausgabestruktur

Wenn die Sicherheitsprüfung erfolgreich war, dann kann eine beliebige Ausgabestruktur zurückgegeben werden.

Wenn bei der Sicherheitsprüfung ein Fehler auftritt, dann muss eine BiPRO-Exception zurückgegeben werden. Die zurückgegebene BiPRO-Exception muss nicht vollständig sein.

Ausgabe im Fehlerfall

```
<nac:BiproException xmlns:nac="http://www.bipro.net/namespace/nachrichten">
    <nac:Status>
        <nac:Meldung>
            <nac:ArtID>Fehler</nac:ArtID>
            <nac:Text>Der Vorgang ist fehlgeschlagen.</nac:Text>
            <nac:MeldungID>99999</nac:MeldungID>
            <nac:Identifikationsnummer>F_900_99999</nac:Identifikationsnummer>
        </nac:Meldung>
    </nac:Status>
</nac:BiproException>
```

2.2.6.2 Fachliche Prüfung (PolicyValidation)

2.2.6.2.1 Eingabestruktur

Eingabestruktur

```
<Request>
    <Service>
        <Name><!-- Endpunkt-Name --></Name>
        <Context><!-- BiPRO-Norm --></Context>
        <Version><!-- BiPRO-Version --></Version>
    </Service>
    <CorrelationId>
        <!-- Eindeutiger Bezeichner (PID) des Serviceaufrufes -->
    </CorrelationId>
    <Timestamp>
        <!-- Timestamp bei Request-Eingang -->
    </Timestamp>
    <Parameters>
        <!-- Implementation-Parameter aus der Servicekonfiguration -->
    </Parameters>
    <Stages>
        <!-- Ergebnisse der vorangegangenen Prozessschritte -->
    </Stages>
    <Headers>
        <!-- Nachrichtenheader -->
    </Headers>
    <Body>
        <!-- Nachrichtenbody -->
    </Body>
</Request>
```

2.2.6.2.2 Ausgabestruktur

Wenn die fachliche Prüfung erfolgreich war, dann kann eine beliebige Ausgabestruktur zurückgegeben werden.

Wenn bei der fachlichen Prüfung ein Fehler auftritt, dann muss eine BiPRO-Exception zurückgegeben werden. Die zurückgegebene BiPRO-Exception muss nicht vollständig sein.

Ausgabe im Fehlerfall

```
<nac:BiproException xmlns:nac="http://www.bipro.net/namespace/nachrichten">
    <nac:Status>
        <nac:Meldung>
            <nac:ArtID>Fehler</nac:ArtID>
            <nac:Text>Der Vorgang ist fehlgeschlagen.</nac:Text>
            <nac:MeldungID>99999</nac:MeldungID>
            <nac:Identifikationsnummer>F_900_99999</nac:Identifikationsnummer>
        </nac:Meldung>
    </nac:Status>
</nac:BiproException>
```

2.2.6.3 Service-Implementierung (ServiceImplementation)

2.2.6.3.1 Eingabestruktur

Eingabestruktur

```
<Request>
    <Service>
        <Name><!-- Endpunkt-Name --></Name>
        <Context><!-- BiPRO-Norm --></Context>
        <Version><!-- BiPRO-Version --></Version>
    </Service>
    <CorrelationId>
        <!-- Eindeutiger Bezeichner (PID) des Serviceaufrufes -->
    </CorrelationId>
    <Timestamp>
        <!-- Timestamp bei Request-Eingang -->
    </Timestamp>
    <Parameters>
        <!-- Implementation-Parameter aus der Servicekonfiguration -->
    </Parameters>
    <Stages>
        <!-- Ergebnisse der vorangegangenen Prozessschritte -->
    </Stages>
    <Headers>
        <!-- Nachrichtenheader -->
    </Headers>
    <Body>
        <!-- Nachrichtenbody -->
    </Body>
</Request>
```

2.2.6.3.2 Ausgabestruktur

Wenn die Serviceimplementierung erfolgreich war, dann kann eine beliebige Ausgabestruktur zurückgegeben werden.

Wenn die Ausgabe genau gesteuert werden soll (z. B. Ausgabe mit Nachrichten-Header), dann kann eine Struktur verwendet werden, die der Eingabestruktur ähnelt.

Detaillierte Ausgabe

```
<Request>
    <Headers>
        <!-- Nachrichtenheader Ausgabe -->
    </Headers>
    <Body>
        <!-- Nachrichtenbody Ausgabe -->
    </Body>
</Request>
```

Wenn bei der Serviceimplementierung ein Fehler auftritt, dann muss eine BiPRO-Exception zurückgegeben werden. Die zurückgegebene BiPRO-Exception muss nicht vollständig sein.

Ausgabe im Fehlerfall

```
<nac:BiproException xmlns:nac="http://www.bipro.net/namespace/nachrichten">
    <nac:Status>
        <nac:Meldung>
            <nac:ArtID>Fehler</nac:ArtID>
            <nac:Text>Der Vorgang ist fehlgeschlagen.</nac:Text>
            <nac:MeldungID>99999</nac:MeldungID>
            <nac:Identifikationsnummer>F_900_99999</nac:Identifikationsnummer>
        </nac:Meldung>
    </nac:Status>
</nac:BiproException>
```

2.2.6.4 Aufzeichnung der Prozess-Nachrichten (Tracing)

Ausgehende Nachrichten werden asynchron aufgezeichnet.
Andere Prozessschritte haben keinen Zugriff auf die Ausgabe dieses Prozessschrittes. Die Aufzeichnung der Prozess-Nachrichten hat keinen Einfluss auf die Antwortzeit.

2.2.6.4.1 Eingabestruktur

Eingabestruktur

```
<Request>
    <Service>
        <Name><!-- Endpunkt-Name --></Name>
        <Context><!-- BiPRO-Norm --></Context>
        <Version><!-- BiPRO-Version --></Version>
    </Service>
    <CorrelationId>
        <!-- Eindeutiger Bezeichner (PID) des Serviceaufrufes -->
    </CorrelationId>
    <Timestamp>
        <!-- Timestamp bei Request-Eingang -->
    </Timestamp>
    <Parameters>
        <!-- Implementation-Parameter aus der Servicekonfiguration -->
    </Parameters>
    <Stages>
        <!-- Ergebnisse der vorangegangenen Prozessschritte -->
    </Stages>
    <Headers>
        <!-- Nachrichtenheader -->
    </Headers>
    <Body>
        <!-- Nachrichtenbody -->
    </Body>
</Request>
```

2.2.7 Anhänge und Optimierung mit MTOM

Mit dem Element `<Optimization>` kann gesteuert werden, wie sich das System verhält, wenn es die Ausgabestruktur optimiert. Dieses Verhalten kann mit dem Element `<Optimization>` auf Gruppen-, Endpunkt- oder Aufrufebene gesteuert werden. Während der Optimierung werden Datenblöcke als Anhang markiert und in einer MIME/Multipart-Nachricht an den Aufrufer übergeben.

Konfiguration aller Dienste der Norm 430 als MTOM-Dienst

```
<Group id="430_Transfer">
    <!-- Diese Einstellung (MTOM) kann man auch am Endpunkt festlegen -->
    <Optimization behavior="strictlyBinary">
        <BinaryContainer local-name="Daten" namespace-uri="http://www.bipro.net/
namespace/allgemein" />
    </Optimization>
</Group>
```

2.2.7.1 Optimierungsmodus Element <behavior>

Wert des Attributs	Verhalten
`default`	Wenn die Gesamtgröße der Nachricht kleiner wird, dann ist die Antwort des Aufrufs eine MIME-Multipart-Nachricht (Standard). Soll auch hier eine MIME-Multipart-Nachricht als Antwort erzwungen werden, so muss der Platzhalter "forceMtomValue" auf -300 gesetzt werden.
`strictlyBinary`	Die Antwort des Aufrufs ist immer eine MIME-Multipart-Nachricht.
`disable`	Die Antwort des Aufrufs ist niemals eine MIME-Multipart-Nachricht.

2.2.7.2 Datenblockmarkierung

Datenblöcke werden erkannt, wenn ihr qualifizierter Name in einem der `BinaryContainer`-Elemente unterhalb des Elements `<Optimization>` enthalten ist.

2.3 Backup

Um ein vollständiges Back-up des X4 BiPRO Server zu erstellen, müssen alle X4-Datenbanken und das Verzeichnis BiPRO (Speicherort siehe unten) gesichert werden.

Je nach Projektimplementierung müssen auch die Daten aus dem Projektbereich gesichert werden.

Die Daten für den X4 BiPRO Server liegen standardmäßig unter *`Installationspfad_X4_Server/BiPRO`*.

Folgende Daten werden in diesem Verzeichnis abgelegt:

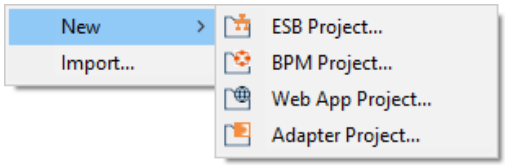
- `biproInstanceState.xml`: Betriebsparameter der X4 BiPRO Server-Installation
- `Logs/Transfers`: Transfer-Objekte, die beim Import-Prozess zu Fehlern geführt haben
- `Temp`: Ablageort von temporären Schema-Dateien

> Der ContentStore wird ab der Version 6.0 nicht mehr unterstützt. Alle bisherigen Dateien, die im ContentStore gespeichert wurden, können migriert werden (siehe Kapitel Migration).

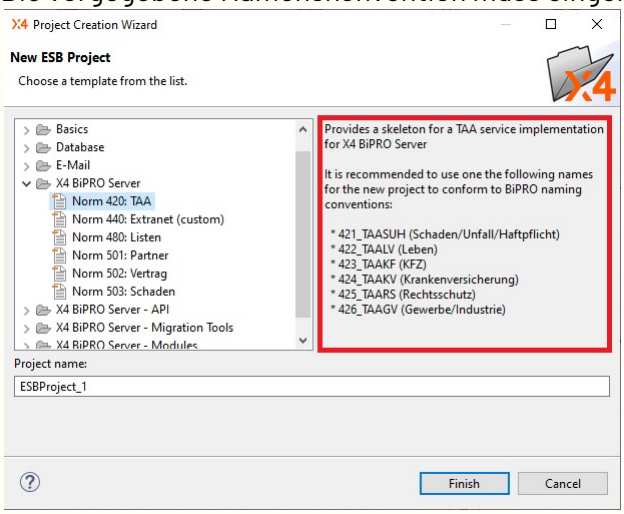
3 Entwickeln mit dem X4 BiPRO Server

3.1 Neues Projekt anlegen

1. Öffnen Sie den X4 Designer.
2. Klicken Sie im Repository rechts und wählen Sie **New** > **ESB Project**.



3. Wählen Sie eine Projektvorlage aus einem der Ordner *X4 BiPRO Server*, *X4 BiPRO Server - API*, *X4 BiPRO Server - Migration Tools* oder *X4 BiPRO Server - Modules* aus.
   Eine detaillierte Erläuterung finden Sie im nachfolgenden Abschnitt **Übersicht über die verschiedenen Vorlagenarten**.
4. Geben Sie einen Projektnamen ein und bestätigen Sie mit **Finish**.

> ⚠ Für die Namen der Projekte gelten bestimmte Namenskonventionen, die im Infobereich anzeigt werden, wenn die entsprechende Projektvorlage ausgewählt wird.
> Die vorgegebene Namenskonvention muss eingehalten werden.
>
> 
>

Das Projekt wird angelegt.

5. Beachten Sie die `Readme`-Datei unter `Projektname/Resources`.

> ✓ Der X4 BiPRO Server bietet über den X4 Designer eine Autovervollständigung an. Diese kann über die Tastenkombination `Strg + Leer` aufgerufen werden.

3.1.1 Übersicht über die verschiedenen Vorlagenarten

Ab dem Release-Linie 6 bietet der X4 BiPRO Server eine erweiterte Vorlagen-Auswahl für die verschiedenen Service-Implementierungen an, die in vier Bereiche gegliedert ist:

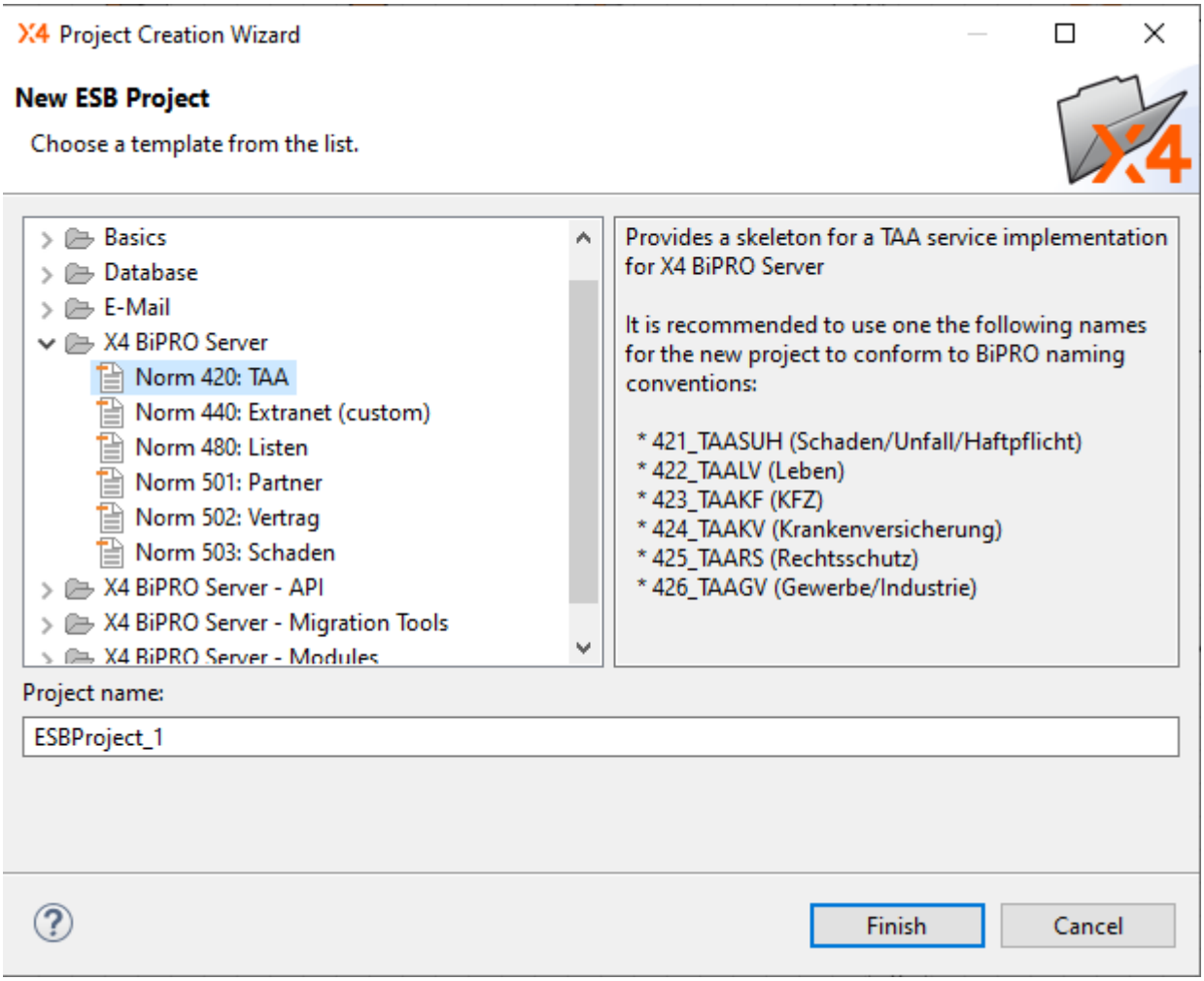
- X4 BiPRO Server
- X4 BiPRO Server - API
- X4 BiPRO Server - Migration Tools
- X4 BiPRO Server - Modules

Von besonderer Bedeutung ist hierbei der "Modules"-Bereich, da dieser die Module für eine Norm 410 und eine Norm 430 Implementierung beinhaltet.

3.1.1.1 X4 BiPRO Server

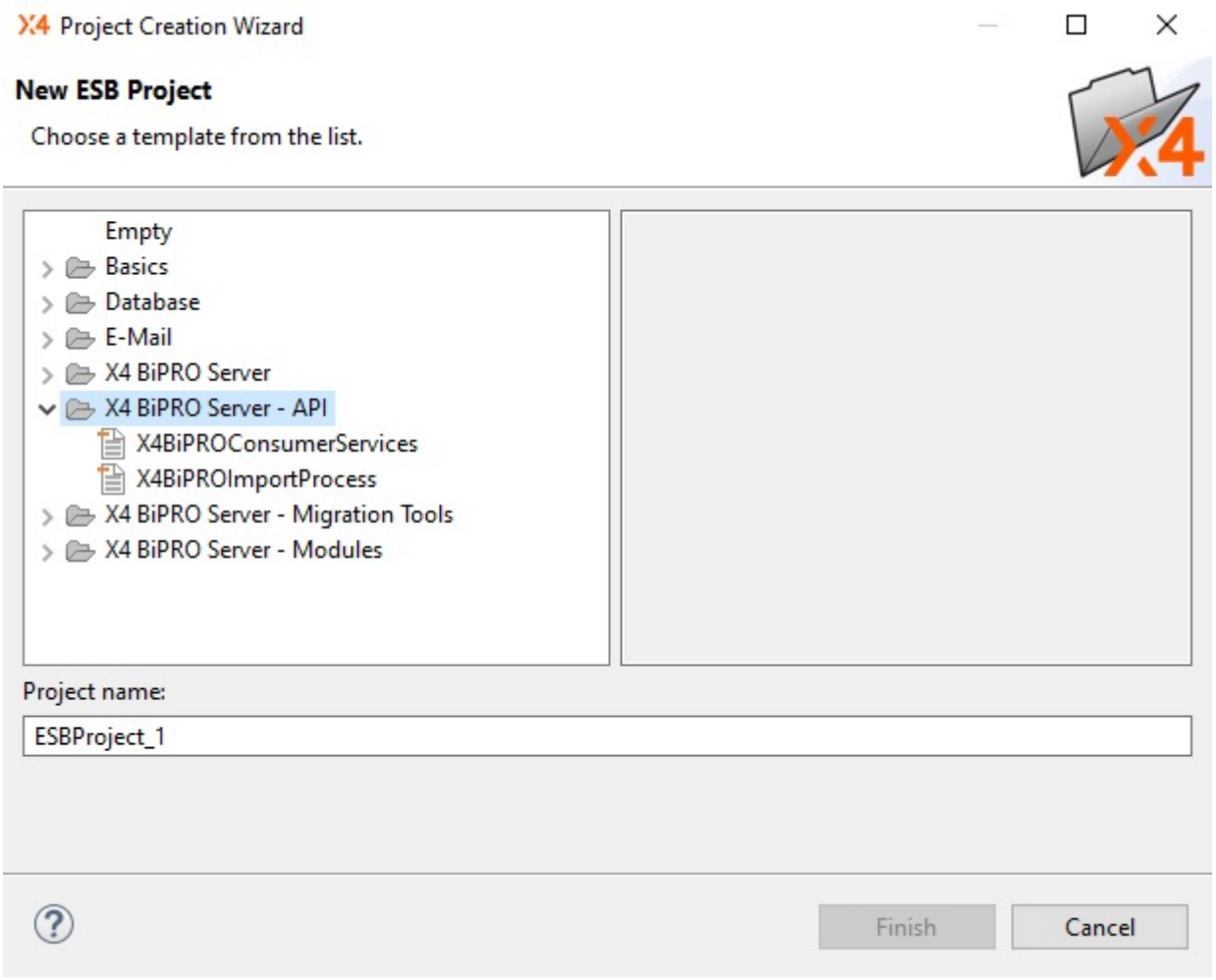
Dieser Bereich beinhaltet die bereits aus vorherigen Versionen des X4 BiPRO Servers bekannten X4 Templates, die ein Grundgerüst für eine BiPRO Service-Implementierung bieten.

Jedes dieser Templates beinhaltet neben den benötigten Service-Methoden in Prozessform auch eine zugehörige Konfigurationsdatei (Service.xml), um den Service bereitstellen zu können.



3.1.1.2 X4 BiPRO Server - API

Dieser Bereich beinhaltet unter anderem aus älteren BiPRO Server Versionen bekannte API-Projekte.



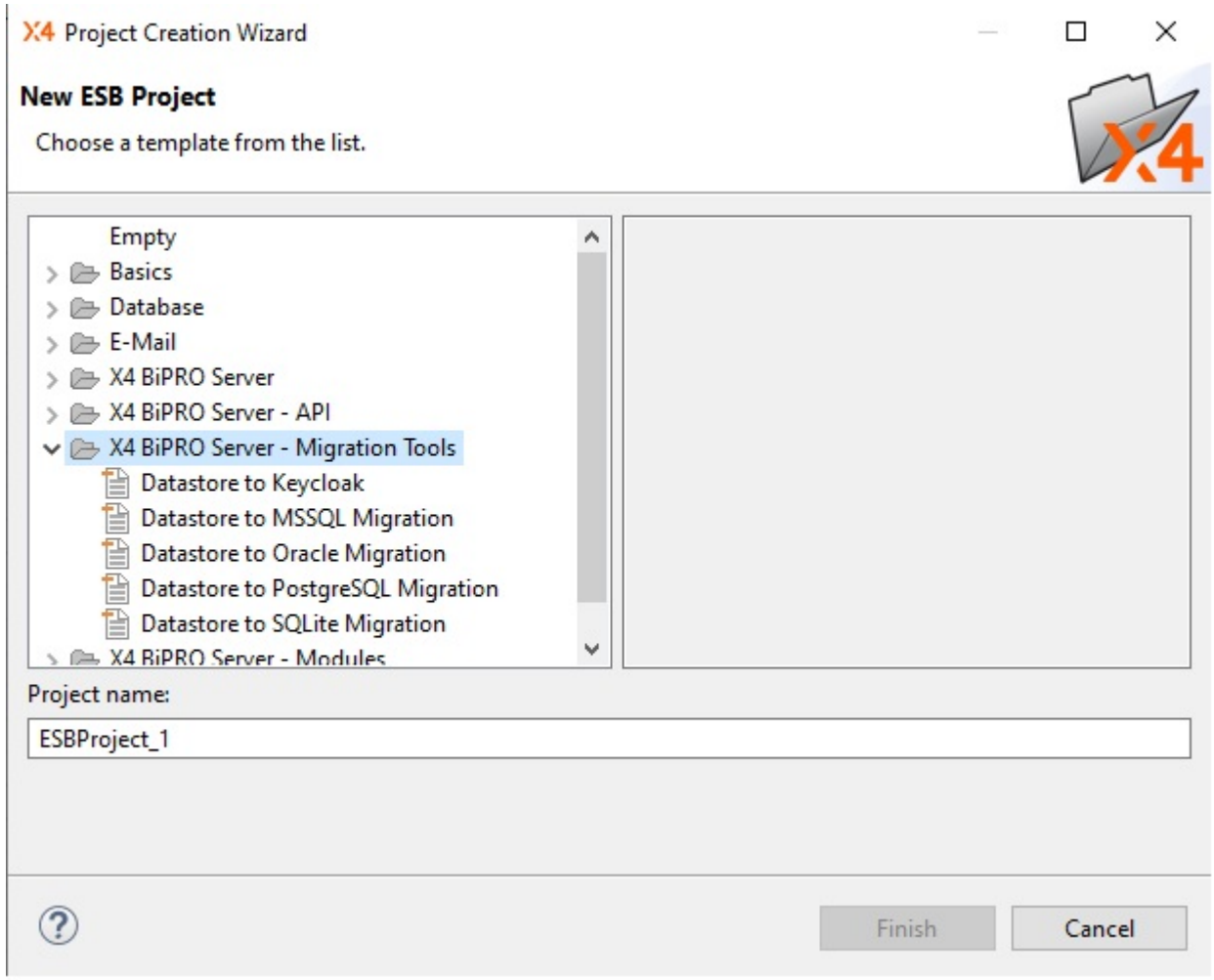
X4BiPROConsumerServices	Diese Vorlage stellt verschiedene Prozesse bereit, mit denen unterschiedliche externe BiPRO Services angebunden werden können
X4BiPROImportProcess	Die in dieser Vorlage enthaltenen Prozesse ermöglichen es, Bestandsdaten BiPRO-konform zu Lieferungen zu gruppieren, in einer Datenbank zu persistieren und für Consumer bereitzustellen

3.1.1.3 X4 BiPRO Server - Migration Tools

Dieser Bereich beinhaltet Vorlagen, mit denen Migrationen aus älteren X4 BiPRO Server Produktlinien durchgeführt werden können. Im Speziellen betrifft dies den damals inkludierten internen Datastore, der in der Vergangenheit als BiPRO Token Speicher oder als initiale Datenbank für den TransferService genutzt wurde.

Mit dem Release 6 des X4 BiPRO Servers wurde dieser nun entfernt, um entweder KeyCloak oder eine Datenbank als Speicher zu nutzen.

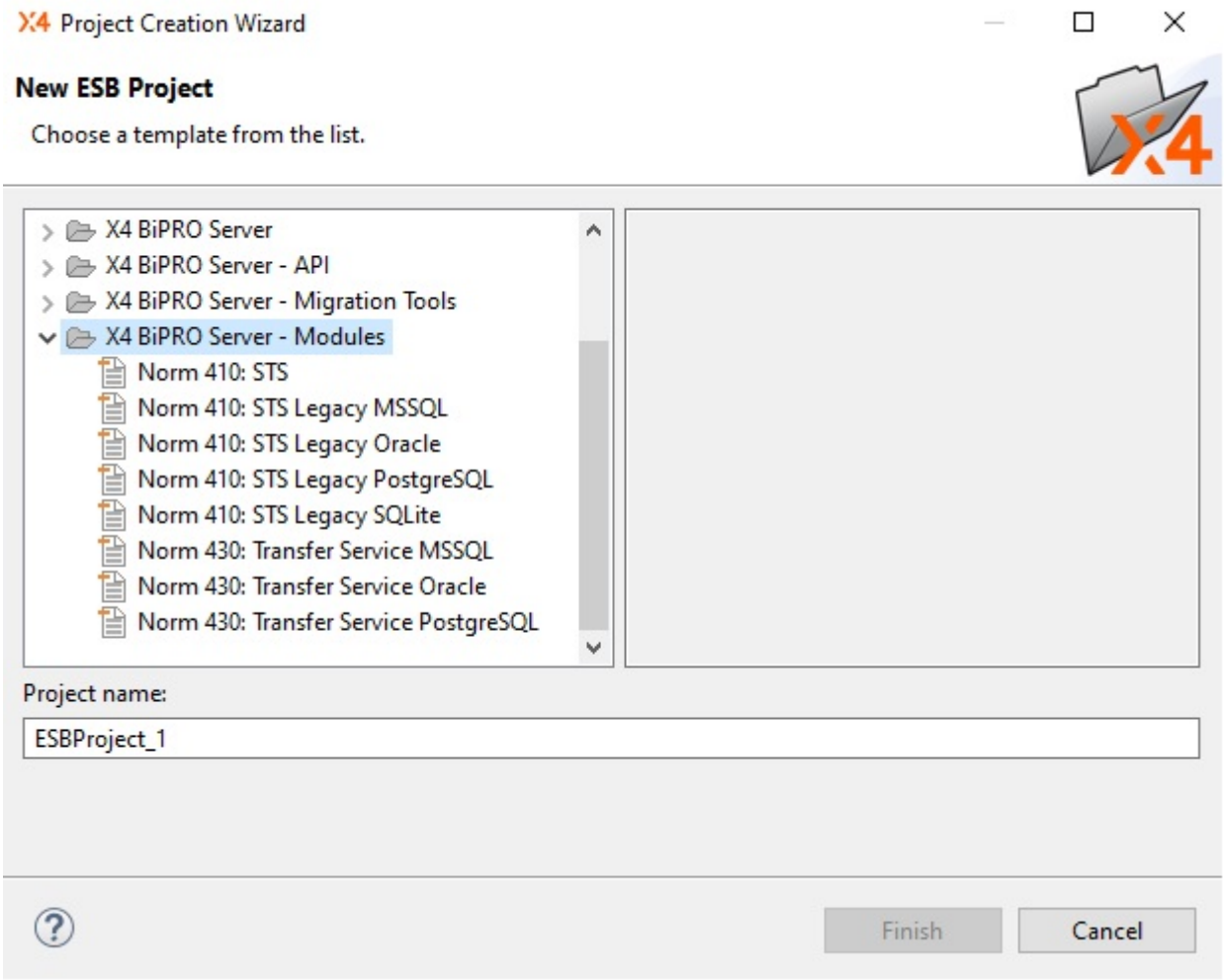
Damit bereits bestehende Datastore-Bestände auch in der neuesten Version des X4 BiPRO Server genutzt werden können, wird über den Bereich *X4 BiPRO Server - Migration Tools* eine Auswahl an Vorlagen zur Durchführung einer Datenmigration angeboten.



Datastore to KeyCloak	Mit diesen Migrationsprozessen können die Datastore-Bestände nach KeyCloak migriert werden.
Datastore to <Database> Migration	Mit diesen Migrationsprozessen können die Datastore-Bestände in eine unterstützte Datenbank migriert werden: • MSSQL • PostgreSQL • OracleDB • SQLite

3.1.1.4 X4 BiPRO Server - Modules

Dieser Bereich beinhaltet die Kernkomponenten des X4 BiPRO Servers: Module, die einen vollwertigen BiPRO Service out-of-the-box anbieten.



Dabei werden für jede unterstütze Datenbank spezifische Vorlagen zur Verfügung gestellt.

Norm 410: STS	Dieses Modul beinhaltet eine vollwertige Implementierung eines BiPRO Norm 410 Security Token Service, der an KeyCloak angebunden wird. Es werden die folgenden Authentifizierungsmethoden für den Security Token Service unterstützt: • User/Password • X.509 Zertifikat

Norm 410: STS Legacy	Dieses Modul beinhaltet eine vollwertige Implementierung eines BiPRO Norm 410 Security Token Service, der an eine Datenbank angebunden wird. • User/Password • X.509 Zertifikat • VDG Ticket Die jeweils anzubindende Datenbank wird anhand des Vorlagennamen-Suffixes gekennzeichnet: • MSSQL • PostgreSQL • OracleDB • SQLite
Norm 430: TransferService	Dieses Modul beinhaltet eine vollwertige Implementierung eines BiPRO Norm 430 TransferServices Die jeweils anzubindende Datenbank wird anhand des Vorlagennamen-Suffixes gekennzeichnet: • MSSQL • PostgreSQL • OracleDB

3.2 Module

3.2.1 Modul BiPRO Norm 410 Security Token Service

Dieses Modul bietet eine "Out-of-the-box"-Lösung eines BiPRO Norm 410 Security Token Service zur Anbindung an KeyCloak.

Neben der standardmäßigen Service-Methode `RequestSecurityToken` wird auch eine generische **SecurityValidation**-Implementierung zur Verfügung gestellt.

3.2.1.1 Installation

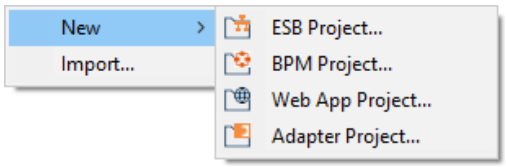
Die Installation des Moduls erfolgt durch die Erstellung eines neuen ESB Projektes, das auf der zugehörigen Projektvorlage **Norm 410: STS** basiert.

Nachdem das Projekt angelegt ist, erfolgt die Konfiguration des Moduls.

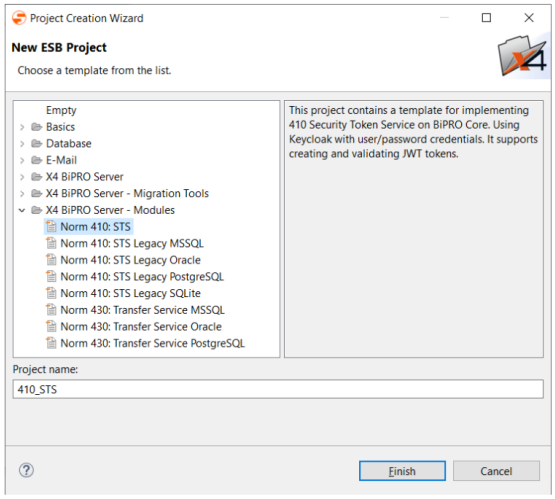
3.2.1.1.1 Vorgehen

Wenn bereits ein 410 STS Legacy installiert wurde, so muss dieser vor Einrichtung des 410 STS mit KeyCloak-Anbindung entfernt werden, da es sonst zu Konflikten kommen kann.

1. Öffnen Sie den X4 Designer.
2. Klicken Sie im Repository rechts und wählen **New** > **ESB Project**.



3. Wählen Sie die Projektvorlage **Norm 410: STS** aus dem Ordner *X4 BiPRO Server - Modules* aus.



4. Geben Sie den Namen für das neue Projekt ein (Empfehlung: *410_STS*) und bestätigen Sie die Eingabe.

3.2.1.2 Konfiguration

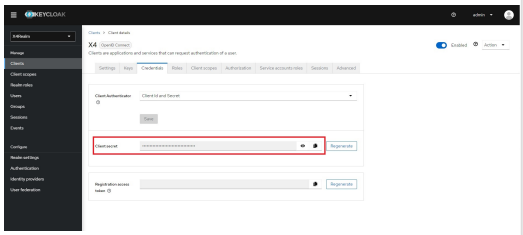
Das Modul muss sowohl im X4 Designer, als auch in KeyCloak konfiguriert werden.

3.2.1.2.1 Modulkonfiguration im X4 Designer

Bevor das Modul genutzt werden kann, müssen die **Custom Placeholders** für den BiPRO Norm 410 Security Token Service definiert werden.

Diese setzen sich wie folgt zusammen:

Name	Beschreibung	Hinweis
scheme	Protokoll, unter dem der KeyCloak Server erreichbar ist	Standard: **http**
host	Hostname des KeyCloak Servers	Standard: **localhost**
port	Portnummer des KeyCloak Servers	Standard: **8085**
pathBase	Die KeyCloak Operation	Standard: **/auth/**

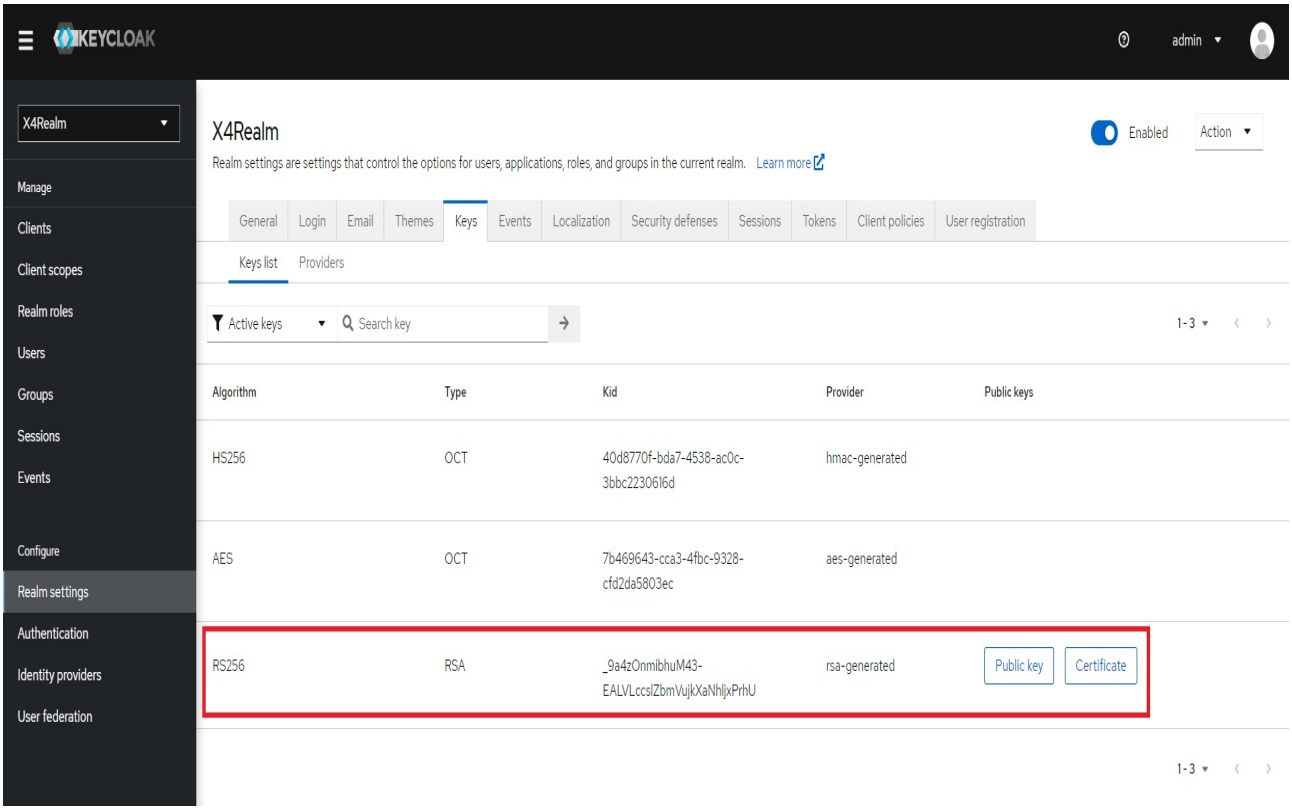
Name	Beschreibung	Hinweis
pathRealm	beinhaltet ausschließlich den Pfad zur Spezifizierung des KeyCloak Realms	Standard: **realms/**
realm	Name des BiPRO KeyCloak Realms	Standard: **BiPRORealm**
clientid	ID des BiPRO KeyCloak Clients	Standard: **BiPRO**
pathProtocol	Protokollpfad	Standard: **/protocol/**
protocol	Protokoll-Typ	Standard: **openid-connect**
accessType	Wie erfolgt der Zugriff?	Standard: **confidential**
clientSecret	Schlüssel zur Authentifizierung am Client	Den Wert des Platzhalters **clientSecret** erhalten Sie über den Bereich **Credentials** des X4 Clients. Hier befindet sich das Client Secret (rot umrandet). Über den **Kopieren**-Button kann der Wert in die Zwischenablage kopiert und anschließend als Wert des Placeholders verwendet werden. 

3.2.1.2.2 Public Key einfügen

Neben dem Client Secret wird auch der Public Key des X4 Realms benötigt.

Dieser findet sich unter **Realm settings** und dort unter dem Reiter **Keys**.

Mit einem Klick auf den Button **Public key** kann der Key angezeigt und der String in die Zwischenablage kopiert und anschließend in die Datei **PublicKey.txt** unter *Resources/Keys/PublicKey.txt* eingefügt werden.

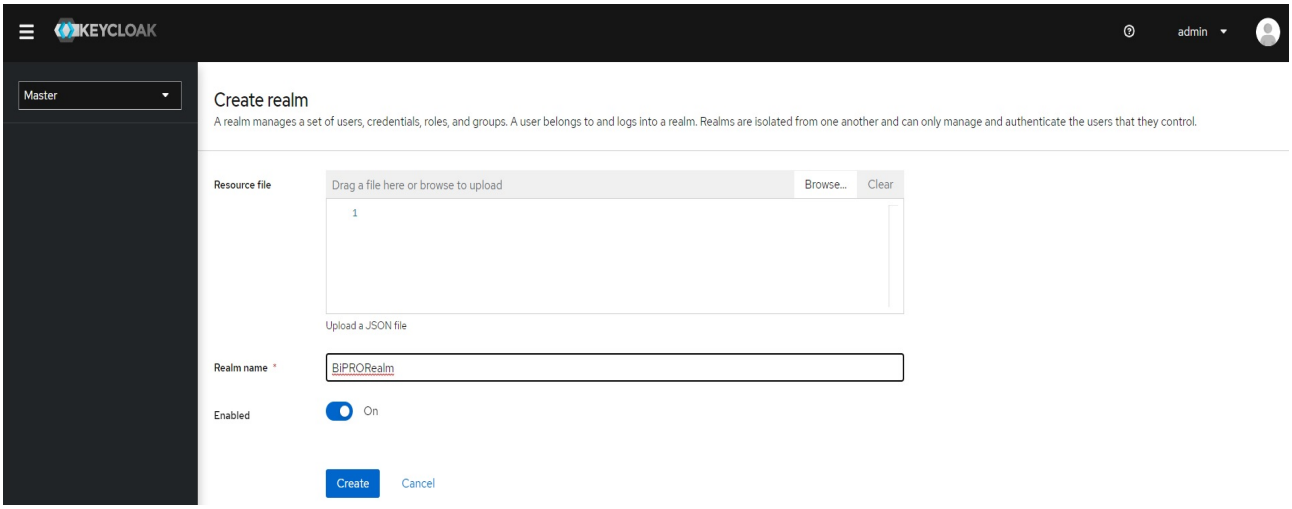


Beispiel des Public Keys:

```
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0iMUhXaUUMcvPsArH94IrxHe05uxVEIRGZCI
ShnFvx431r9ZqoaaFzpj9B2jTvCXuzhbcwn8uuShpwyG8spSpIjuTpGYpO8OXDa9NowhckkILXmHX+Ij4
81DnDLSS6OzkZSn3nJ5PO25Va6L0vzwe86ZmtBZ8zs7WLBvi6LcBh0rp3ZLhf1sgZ4iWSVioApFyEGUJbh
ru3aWUVTK1jyH72bJ1N+JHqZ/j/kk9wZKuvkQ5yAhWj9dJxjTf0iv4TtnyGD25qJsjZh98MkgBqo2gjAgVZb
5fuXbAM7LGFG8Aeb5M4wNSG13jdJfdQg3EgORMRCIR3KfqKCMEFOumZ4u3QIDAQAB
```

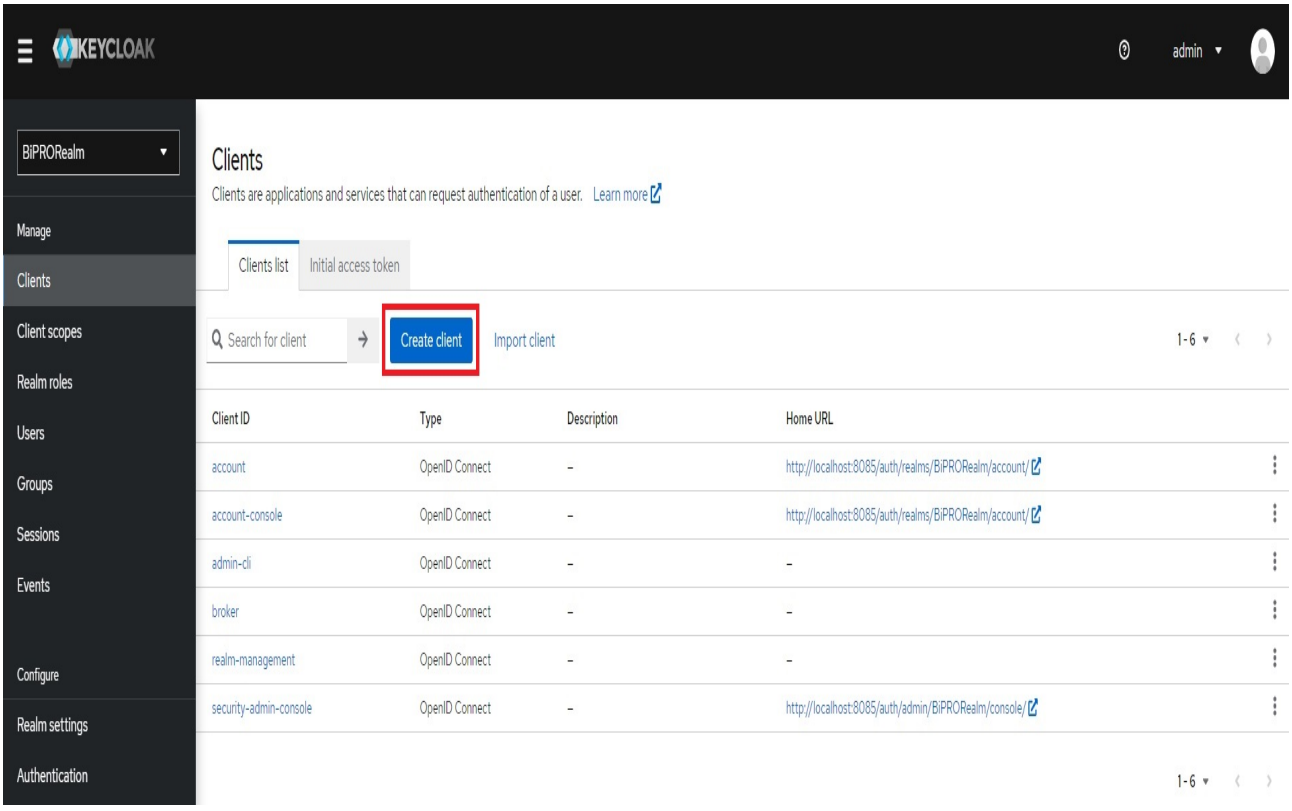
3.2.1.2.3 KeyCloak Konfiguration

Legen Sie im KeyCloak ein neuer Realm mit dem Namen **BiPRORealm** an.

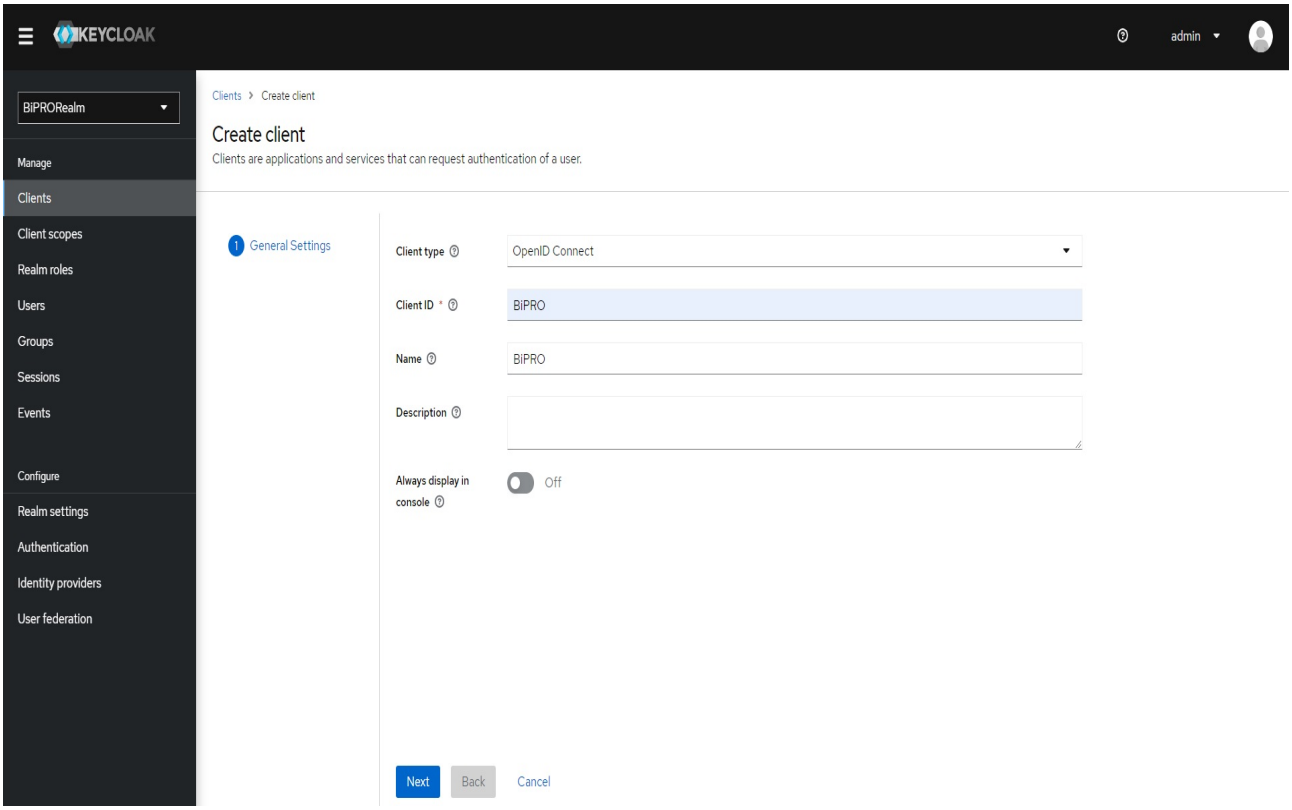


Legen Sie anschließend innerhalb des im vorherigen Schritt erstellten Realms einen neuer Client mit dem Namen **BiPRO** an.

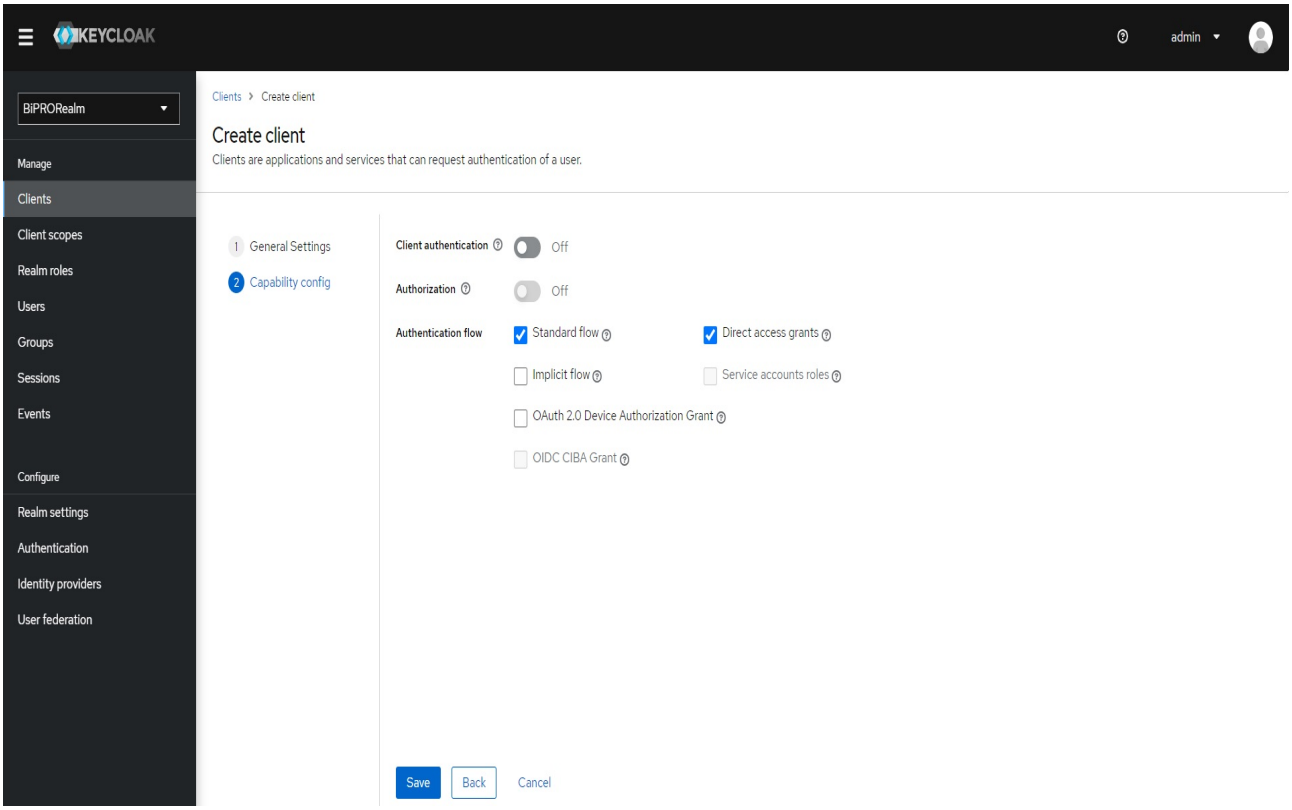
Dazu klicken Sie im Menü links auf "Clients" und anschließend auf den Button "Create client".



Im sich nun öffnenden Wizard legen Sie nun zunächst den Namen des neuen Clients fest (z.B. "BiPRO").



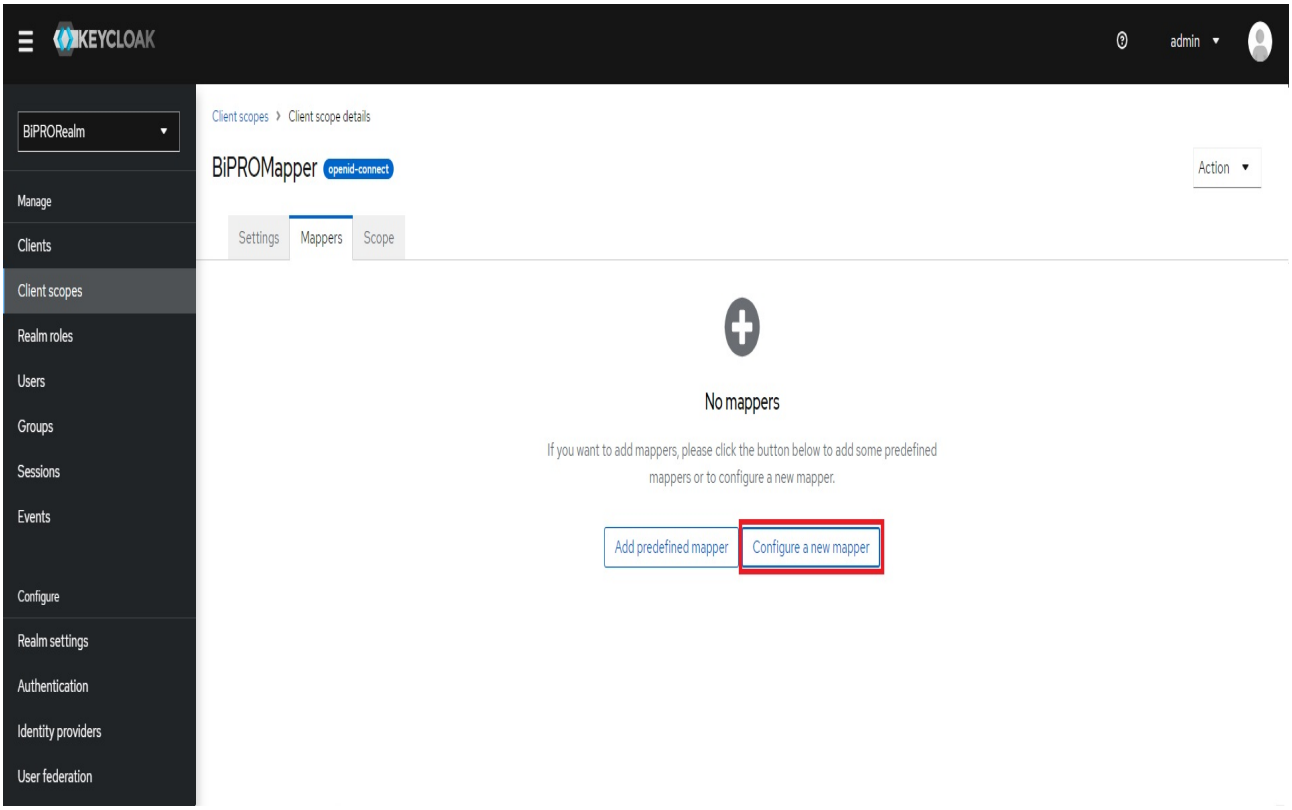
Im nächsten Schritt setzen Sie den Regler bei **Authorization** auf ON und speichern anschließend den neuen Client.



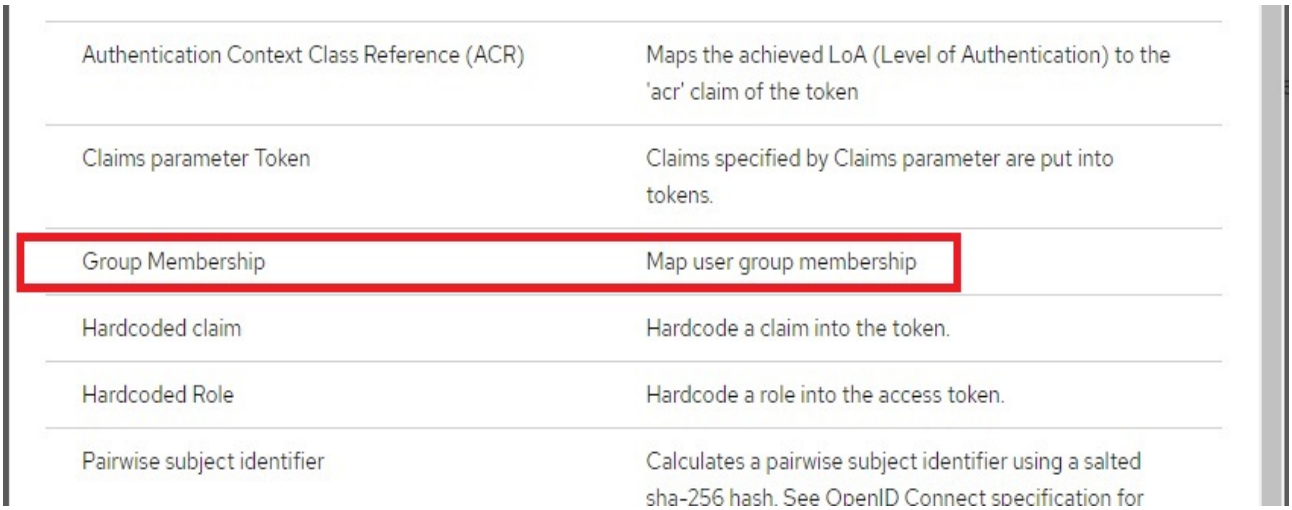
Anschließend benötigen Sie einen Mapper, um Gruppen/Organisationen verwenden zu können.

Dazu erstellen Sie im **BiPRORealm** über den Bereich **Client scopes** einen neuen Scope mit der Bezeichnung **BiPROMapper**.

Anschließend erstellen Sie dort unter dem Reiter **Mappers** einen neuen Mapper mit Klick auf **Configure new mapper**.



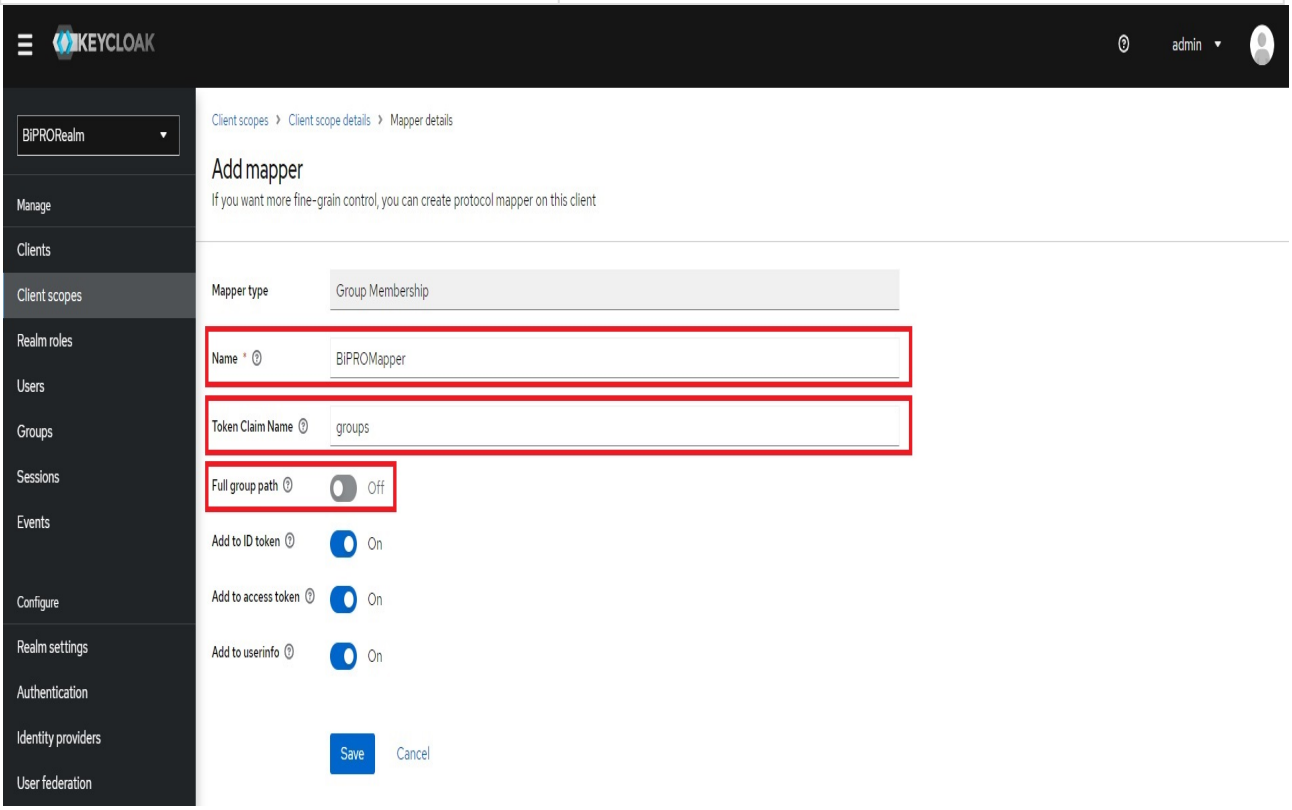
Im nun erscheinenden Auswahlmenü wählen Sie den Eintrag **Group Membership** aus.



Anschließend nehmen Sie die nachfolgenden Anpassungen für den künftigen Mapper vor und speichern anschließend:

Mapper Name	Der Name des Mappers (Bspw. *BiPROMapper*)

Token Claim Name	Als Bezeichner muss der Begriff *groups* eingetragen werden
Full group path	Der Status muss auf *OFF* gesetzt werden



Der Mapper ist nun angelegt und es können Gruppen/Organisationen für den 430 Kontext angelegt werden.

3.2.1.3 Benutzerverwaltung über KeyCloak

Mit erfolgreicher Einrichtung und Konfiguration des BiPRO Norm 410 Security Token Services können nun in KeyCloak das User Management vorgenommen werden.

Hierbei ist zu beachten, dass die KeyCloak-Entitäten **Benutzer** (User) bzw. **Gruppen** (Groups) im Kontext des X4 BiPRO Servers jeweils eine besondere Bedeutung haben, insbesondere bei der Verwendung des BiRO Norm 430 TransferService.

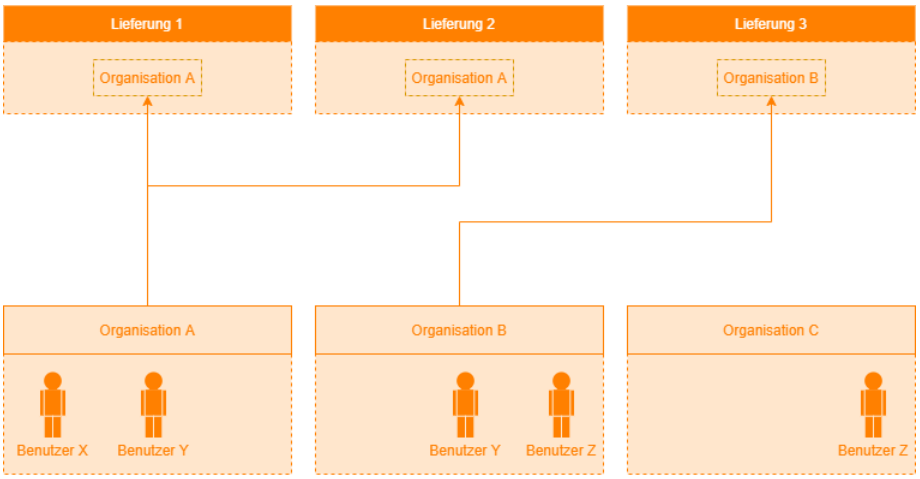
Benutzer (User)	Jeder in KeyCloak angelegte Benutzer repräsentiert einen Endnutzer, der die jeweils verfügbaren BiPRO Services nutzt. Dieser Endnutzer kann dabei sowohl eine reale Person, als auch ein technischer Nutzer sein.

Gruppen (Groups)	Eine Gruppe in KeyCloak repräsentiert die Entität **Organisation** im Kontext des X4 BiPRO Servers. Eine Organisation wiederum ist eine • Ein einzelner Vermittler • Eine Vermittlergruppe (beispielsweise, wenn mehrere Vermittler/Makler in einem Verbund arbeiten und die selben Fälle bearbeiten)

Das nachfolgende Beispiel veranschaulicht die Definition der Entitäten **Benutzer** und **Gruppen**.

Dabei gilt folgende Konstellation:

- Es existieren die Organisationen A, B und C
- Benutzer X und Benutzer Y gehören der Organisation A an
- Benutzer Y und Benutzer Z gehören der Organisation B an
- Benutzer Z gehört der Organisation C an



Wenn nun Lieferungen bereit gestellt werden, so werden diese immer für eine Organisation bereit gestellt.

Im Beispiel bedeutet das:

- Lieferung 1 und Lieferung 2 werden für die Organisation A bereit gestellt
- Lieferung 3 wird für die Organisation B bereit gestellt
- Für Organisation C wird keine Lieferung bereit gestellt

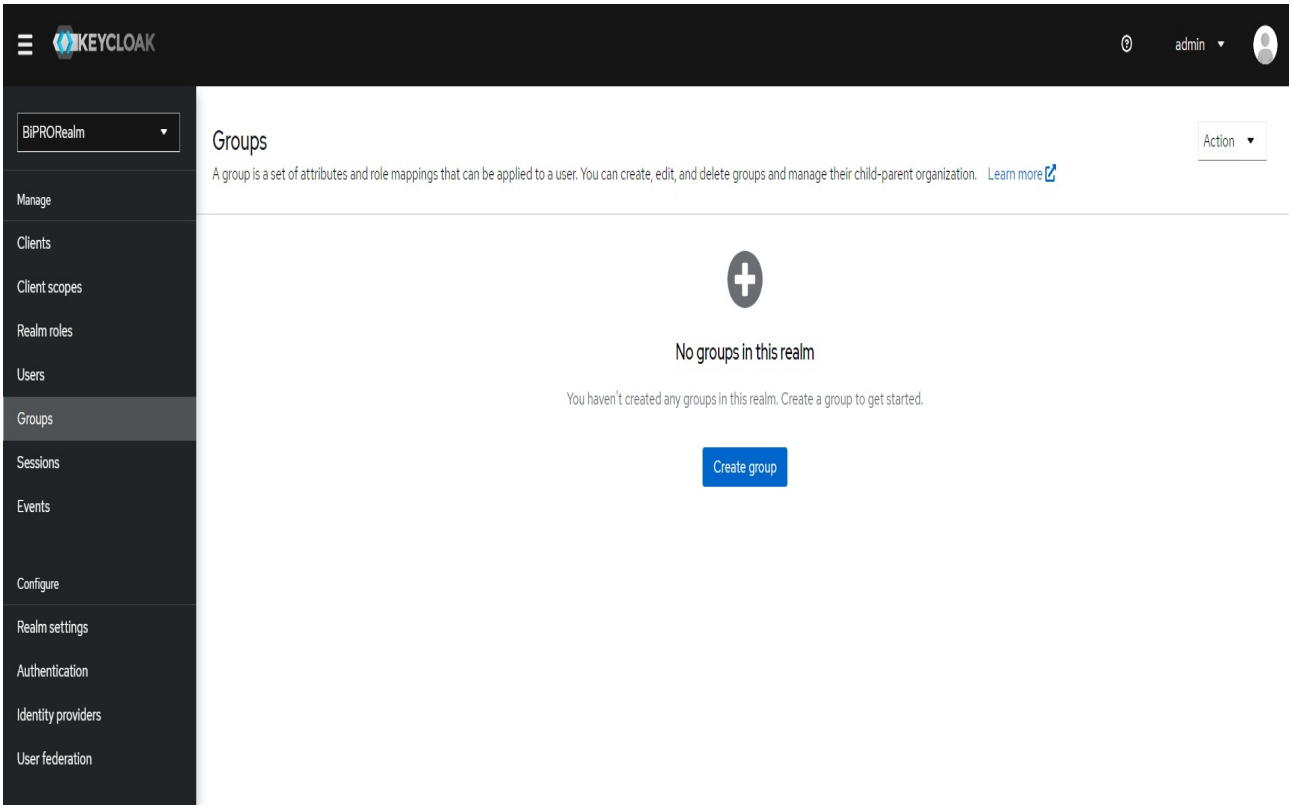
Aufgrund der beschriebenen Organisationszugehörigkeit der Benutzer ergeben sich nun die folgenden Zugriffsmöglichkeiten:

- Mit Zugehörigkeit zu Organisation A haben die Benutzer X und Y Zugriff auf die Lieferungen 1 und 2

- Mit Zugehörigkeit zu Organisation B haben die Benutzer Y und Z Zugriff auf die Lieferung 3

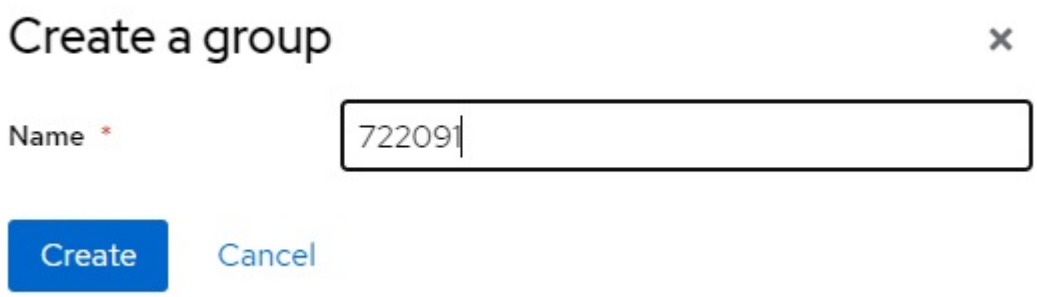
3.2.1.3.1 Gruppen in KeyCloak verwalten

Öffnen Sie KeyCloak und wechseln Sie auf den Realm **BiPRORealm**. Klicken Sie dort im Bereich **Manage** auf den Punkt **Groups.**



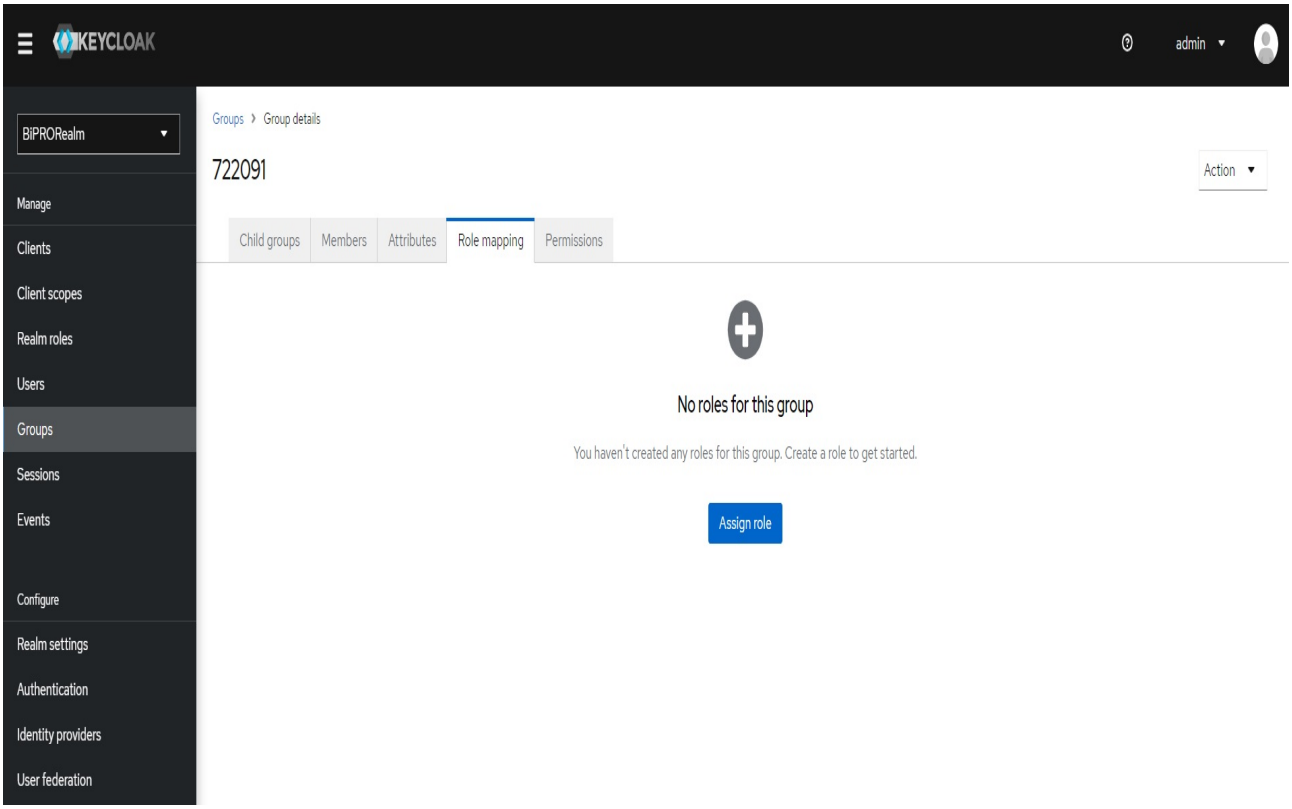
Klicken Sie auf den den Button **Create group,** um eine neue Gruppe (**Organisation** im BiPRO Server Kontext) anzulegen.

Hierbei **muss** der Name eindeutig sein, d.h. der Name der Gruppe muss exakt dem Namen der Organisation entsprechen, für die Lieferungen bereit gestellt werden.



Speichern Sie nach Vergabe eines eindeutigen Namens die neue Gruppe.

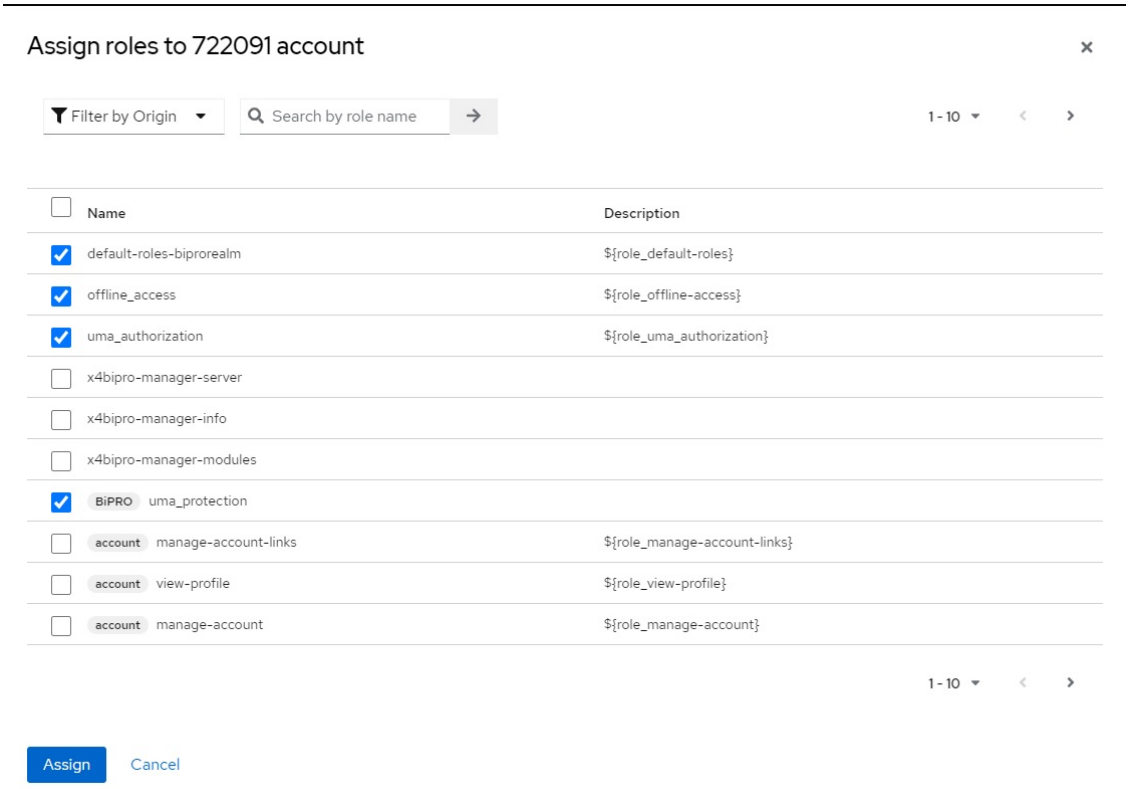
Wechseln Sie nun im Menü der Gruppe auf den Reiter **Role Mappings** und klicken Sie dort auf **Assign role** um die benötigten Berechtigungen zuzuweisen.



Selektieren Sie die Einzelberechtigungen

- default-roles-biprorealm
- offline_access
- uma_authorization
- uma_protection

und speichern Sie anschließend die Anpassungen.



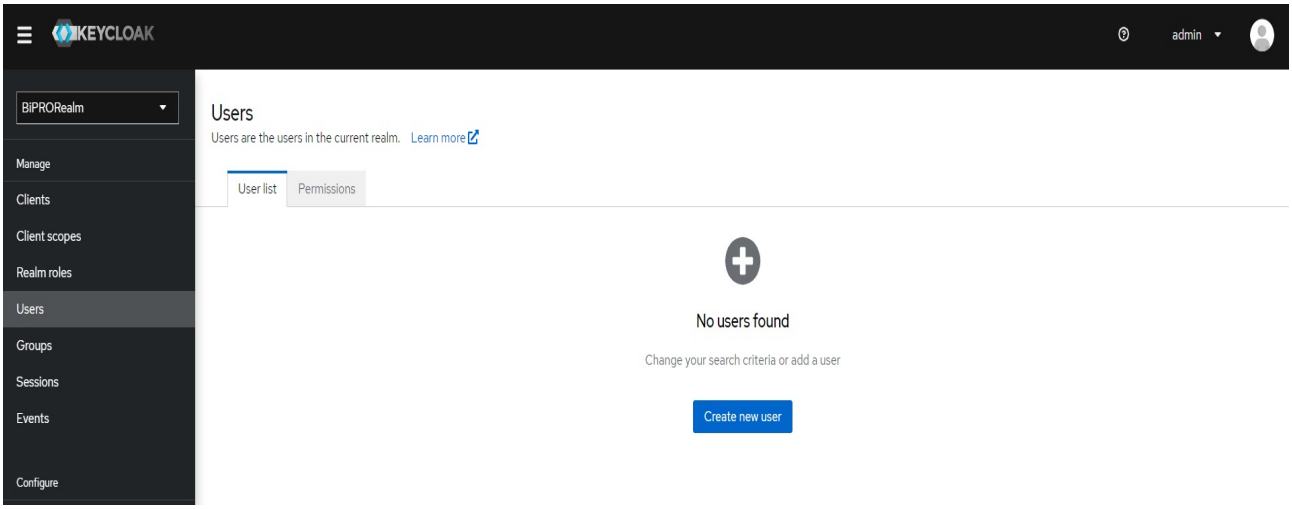
Die Gruppe bzw. die Organisation ist nun konfiguriert.

Im nächsten Schritt wird erläutert, wie die entsprechenden **Benutzer** (**Anmeldungen** im BiPRO Server Kontext) angelegt und entsprechend **Gruppen** (**Organisationen**) zugewiesen werden können.

3.2.1.3.2 Benutzer in KeyCloak verwalten

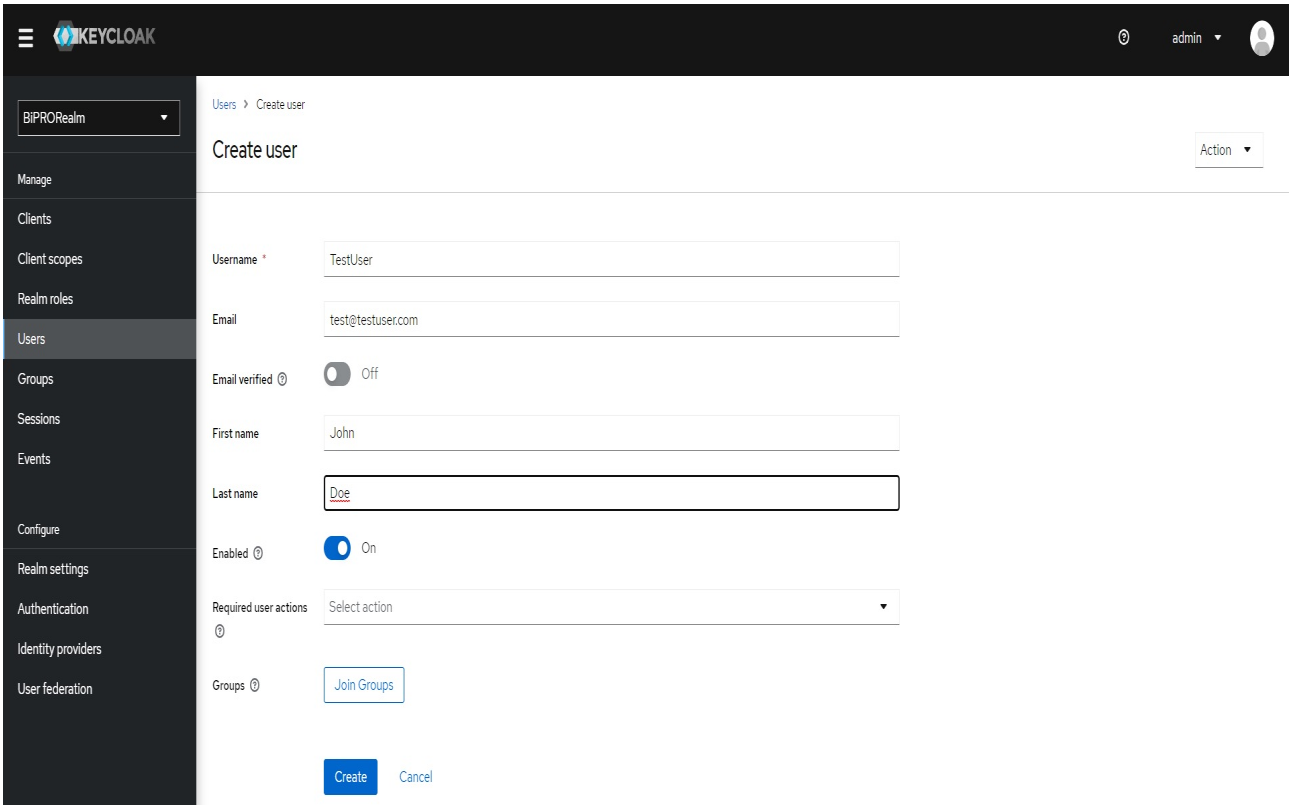
Öffnen Sie KeyCloak und wechseln Sie auf den Realm **BiPRORealm**. Klicken Sie dort im Bereich **Manage** auf den Punkt **Users.**

Klicken Sie auf den den Button **Create new user**, um einen neuen **Benutzer** (**Anmeldung** im BiPRO Server Kontext) anzulegen.

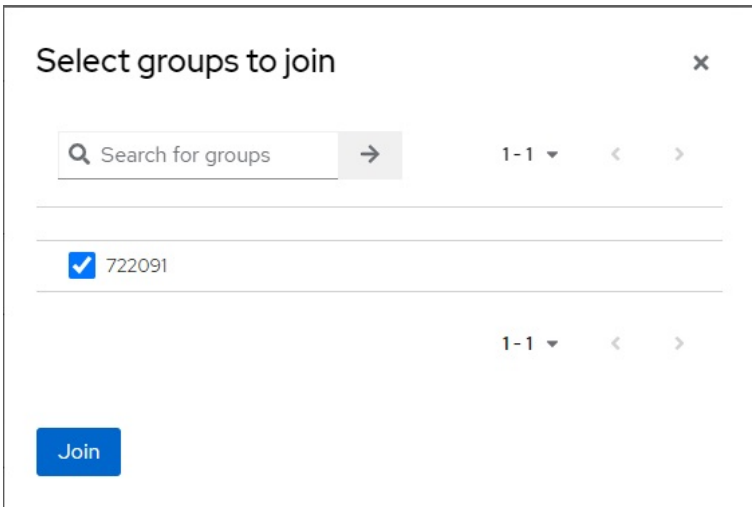


Vergeben Sie einen eindeutigen Benutzernamen, sowie bei Bedarf weitere Informationen.

Wenn die X.509 Zertifikatsauthentifizierung ebenfalls verwendet werden soll, ist die Angabe der **Email-Adresse** Pflicht.



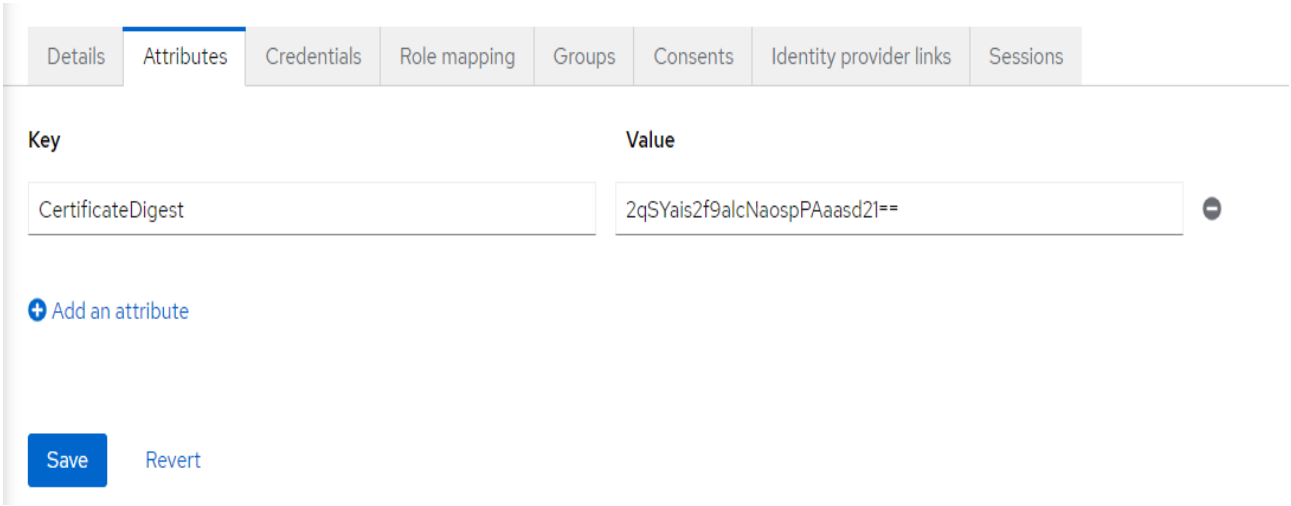
Als letzten Schritt klicken Sie auf den Button **Join Groups** und wählen im neuen Overlay-Fenster die entsprechende Gruppe (Organisation) aus.



Mit Klick auf den Button **Join** ist der Benutzer nun Mitglied dieser Gruppe (Organisation) und entsprechend berechtigt, die zugehörigen Lieferungen zu empfangen.

3.2.1.3.3 X.509 Authentifizierung

Wird auch die X.509 Authentifizierungsmethode verwendet, so muss der Benutz noch ein Attribut erhalten:



Attributname	Wert
CertificateDigest	X.509 Zertifikat als Base64

3.2.1.4 Migration aus vorherigen BiPRO Server Versionen

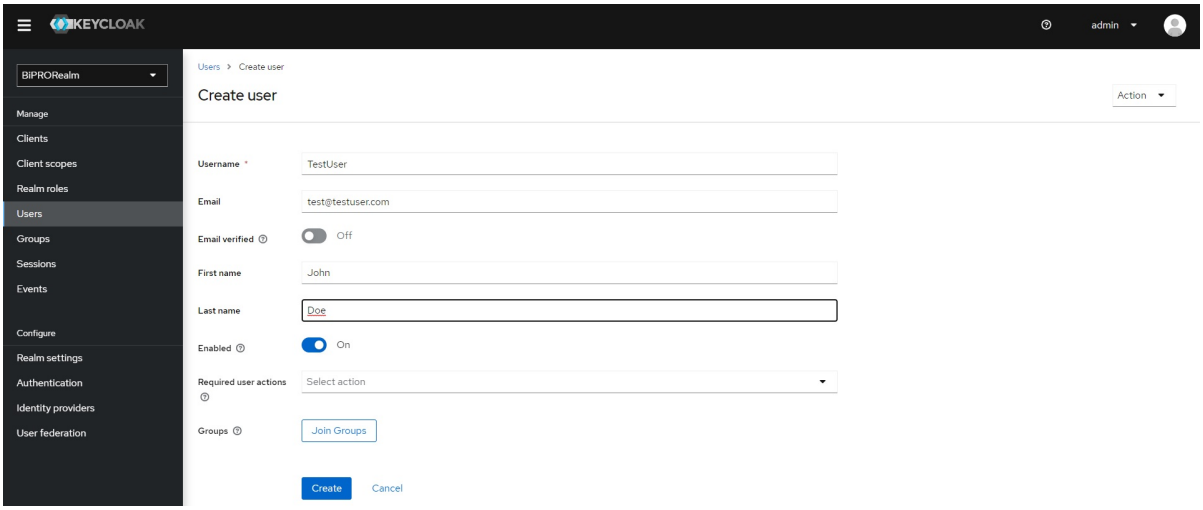
Wenn Sie die Datenbestände eines älteren X4 BiPRO Servers auf die aktuelle Produktlinie migrieren möchten, kontaktieren Sie einen SoftProject-Consultant. Dieser unterstützt Sie gerne bei der Migration.
Für kundenseitig ausgeführte Upgrades kann SoftProject keine Gewähr übernehmen.

3.2.1.5 Authentifizierungsvarianten

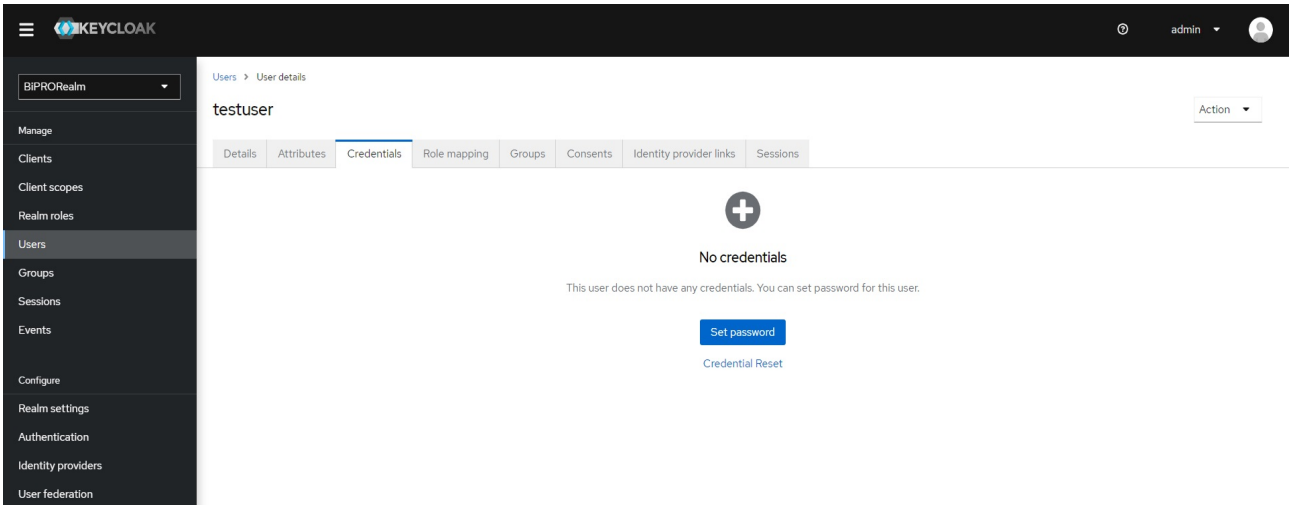
3.2.1.5.1 User/Password

Benutzer werden in KeyCloak im BiPRORealm im Bereich **Users** gepflegt.

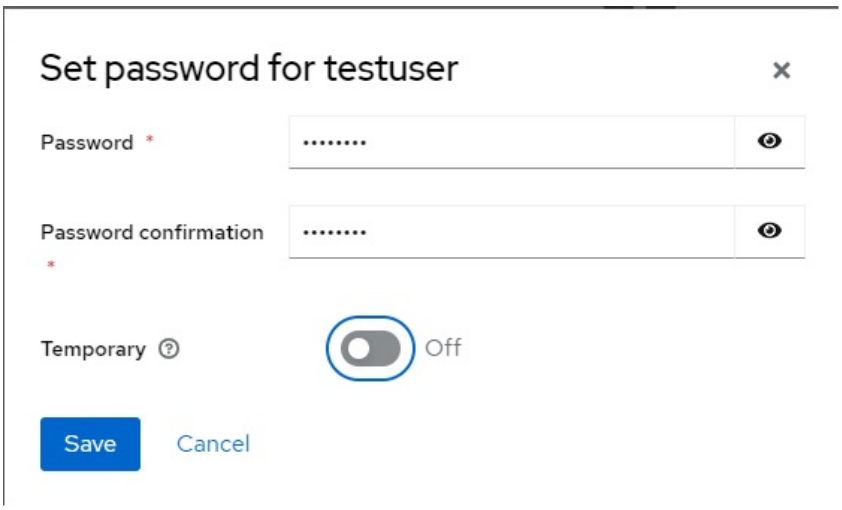
Hier können neue Benutzer über den Button **Create user** angelegt werden.



Ein Passwort wird im Bereich Credentials vergeben. Klicken Sie dazu auf **Set password**.



Wichtig ist nun, nach der Vergabe eines starken Passworts, dass der Schalter bei **Temporary** auf **Off** gestellt wird.



Durch einen Klick auf **Save** wird das Passwort gespeichert und der Benutzer kann sich nun über diese Credentials am Security Token Service anmelden.

3.2.1.5.2 X.509 Authentifizierung

3.2.1.5.2.1 Installation

Für die Bereitstellung der Authentifizierungsvariante mit X.509 Zertifikat bedarf es einer expliziten Custom-Condition für. sowie zusätzlichen Konfigurationsschritten in KeyCloak.

Die Custom-Condition kann auf zwei Wegen installiert werden:

- Über das BiPRO Server Installationsskript

- Durch Kopieren der Datei **Custom-conditional-keycloak.jar** aus dem BiPRO Server Installationspaket in das Verzeichnis **<Serverpfad>/keycloak/providers**

Diese Custom-Condition wird später bei der Konfiguration der Autorisierungsvarianten in KeyCloak benötigt.

3.2.1.5.2.2 Vorbedingung

Um die X.509 Authentifizierungsmethode verwenden zu können, müssen die betroffenen Benutzer in KeyCloak entsprechend konfiguriert sein.

Dazu gehört neben der Vergabe einer Email-Adresse auch die Zuweisung bestimmter Attribute.

Ebenfalls muss ein Management User angelegt werden, der später die User Digests überprüft.

Hierzu kann der BiPRO Service User verwendet werden, er muss jedoch noch zusätzlich über die nachfolgenden Berechtigungen verfügen:

Benutzerrolle	Beschreibung
view-realm	Um KeyCloak Realms sehen zu können
view-users	Um KeyCloak Benutzer sehen zu können

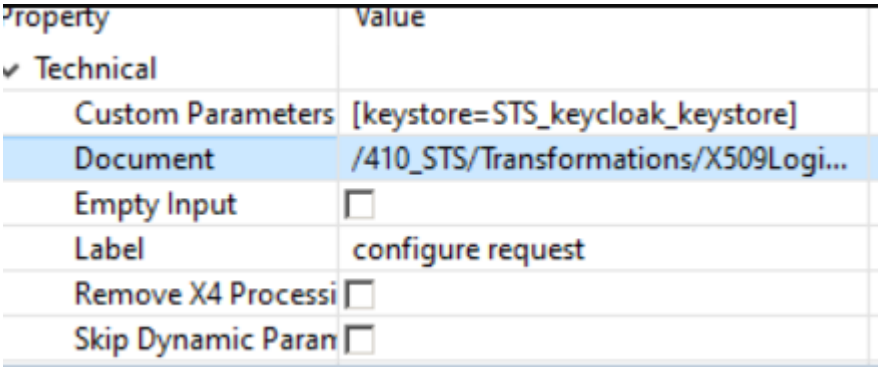
Ein weiterer Schritt ist die Anlage eines Keystores sowie der zugehörigen Verweise über den Designer im BiPRO Server unter **X4BiPRO/Resources/Security.xml**.

```
<NamedKeystore id="STS_keycloak_keystore" type="jks" href="xstore://410_STS/Resources/keystorefred.jks" password="testtest">
    <NamedKey id="managementuser" alias="managementuser" password="testtest"></NamedKey>
    <NamedKey id="fred.flintstone" alias="fred.flintstone" password="testtest"></NamedKey>
</NamedKeystore>
```

Anschließend muss der Keystore im Prozess X509LoginKeycloak.wrf referenziert werden.

Hierbei muss der CustomParameter **keystore** den Key des Keystores als Wert besitzen.

Im Beispiel:



3.2.1.5.2.3 TLS Konfiguration

Im ersten Schritt müssen die Zertifikate und Keystores spezifiziert werden.

Öffnen Sie dazu die Datei **keycloak.conf** unter **<X4Serverpfad>/keycloak/conf** mit einem Texteditor.

Erweitern Sie diese Datei um nachfolgende Struktur.

```
https-client-auth=request
https-enabled=true

https-certificate-file=C:\\X4\\Server\\keycloak\\conf\\localhost.crt
https-certificate-key-file=C:\\X4\\Server\\keycloak\\conf\\localhost.key

https-key-store-file=C:\\X4\\Server\\keycloak\\conf\\keystore.jks
https-key-store-password=test

https-trust-store-file=C:\\X4\\Server\\keycloak\\conf\\truststore.jks
https-trust-store-password=testtest
```

Parametername	Beschreibung
https-client-auth	Typ der Clientauthentifzierung
https-enabled	Aktiviert den HTTPS Listener
https-certificate-file	Dateipfad der Zertifikatsdatei
https-certificate-key-file	Dateifpfad des Private Key
https-key-store-file	Dateipfad des Keystores
https-key-store-password	Passwort des Keystores
https-trust-store-file	Trust Store der Zertifikatsinformationen
https-trust-store-password	Passwort des Trust Stores

Speichern Sie anschließend die Anpassungen.

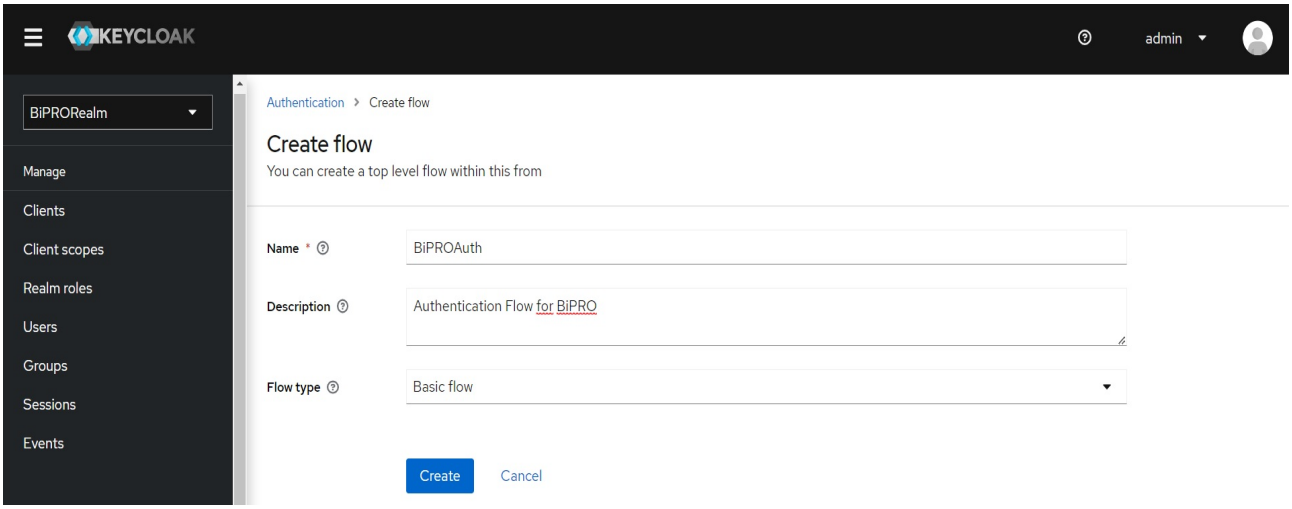
3.2.1.5.2.4 Konfiguration in KeyCloak

Wechseln Sie nun in KeyCloak auf den Realm **BiPRORealm** und klicken Sie dort auf den Bereich Authentication.

Erstellen Sie einen neuen **Authorization Flow** über den Button **Create flow**.

BILD

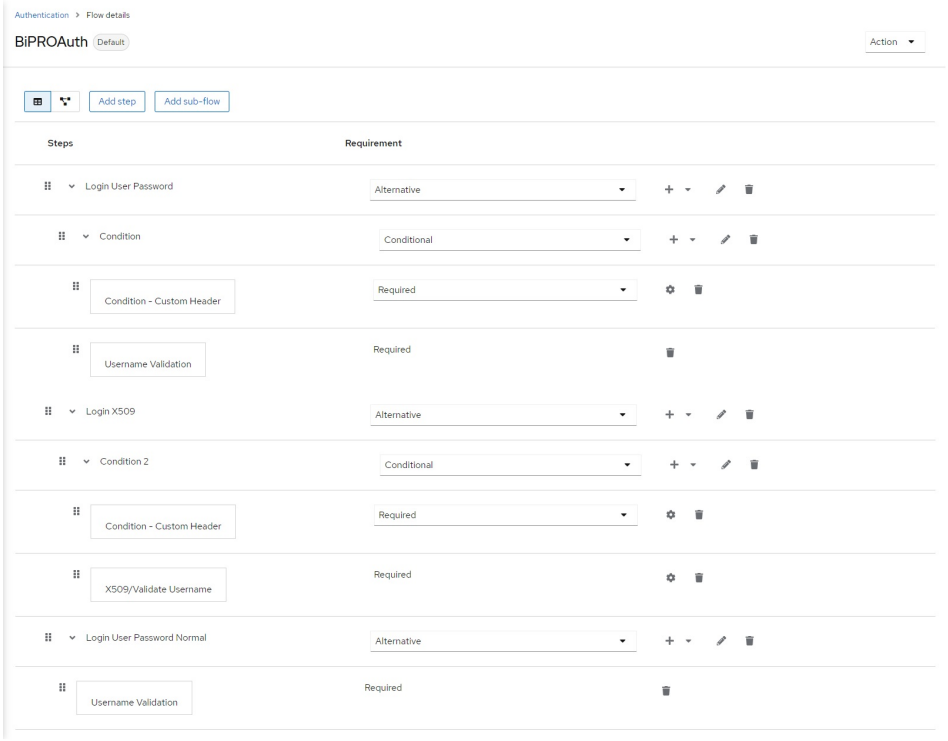
Vergeben Sie einen Namen (im Beispiel **BiPROAuth**), sowie bei Bedarf eine Beschreibung, belassen den Flow type auf Basic flow und klicken Sie auf **Create**.



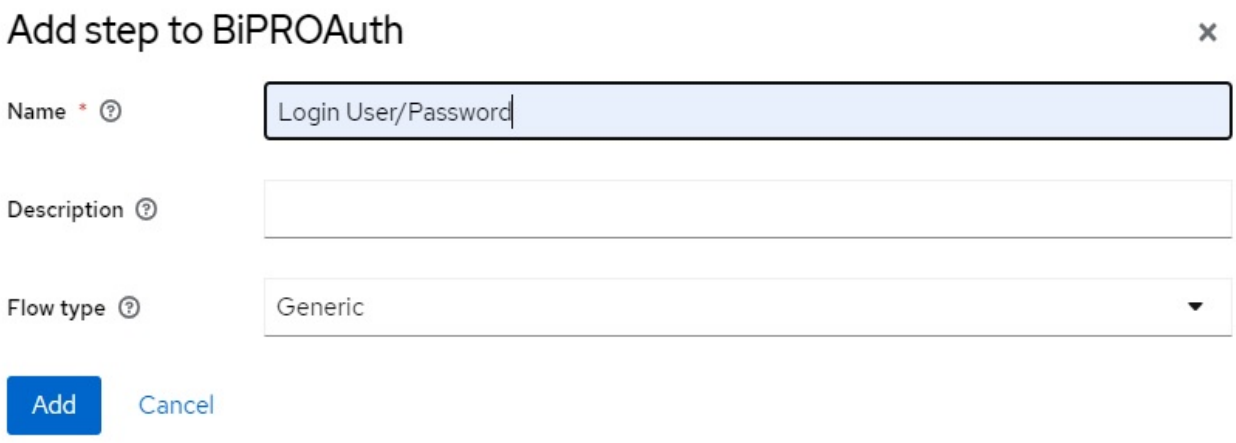
Der neue Flow wird angelegt und kann nun konfiguriert werden.

Anlage des Authentication Flow
Entfernen Sie nun zunächst die zwei existierenden Einträge.

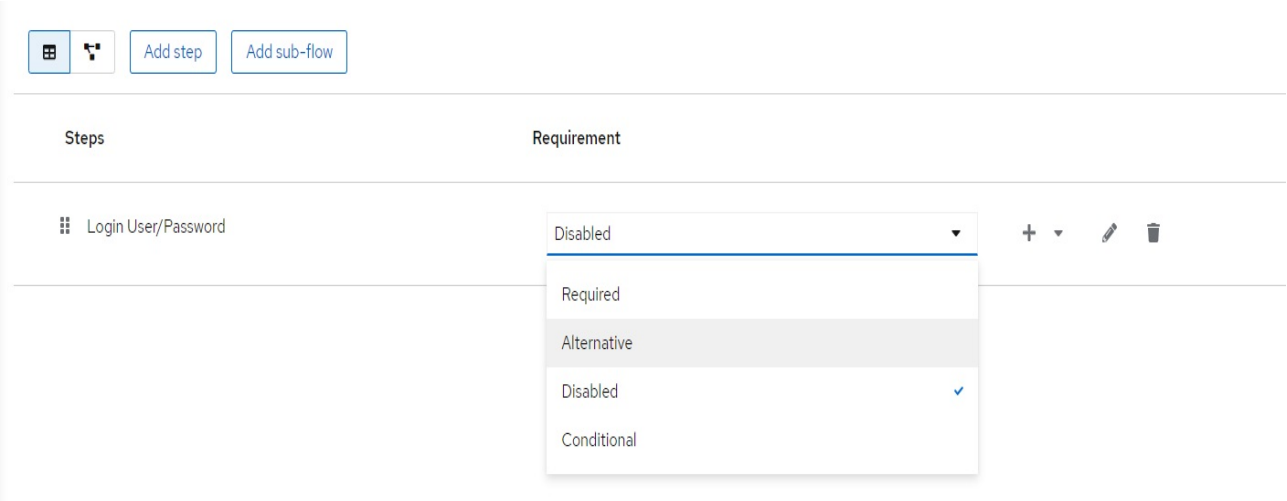
Anschließend legen Sie die nachfolgende Struktur über die Verwendung von **Add step** bzw. **Add sub-flow** an:



User Password Login
Erstellen Sie zunächst einen **sub-flow** und nennen diesen **Login User/Password.**



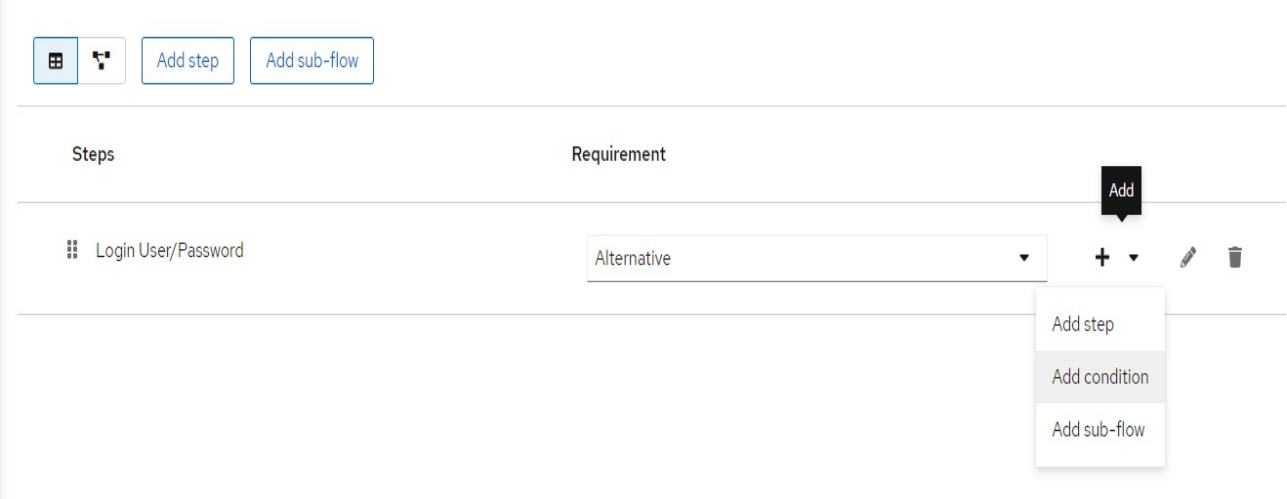
Nachdem der neue sub-flow über **Add** angelegt wurde, klicken Sie nun auf das Dropdown-Menü für **Requirements** und setzen den Wert auf **Alternative**.



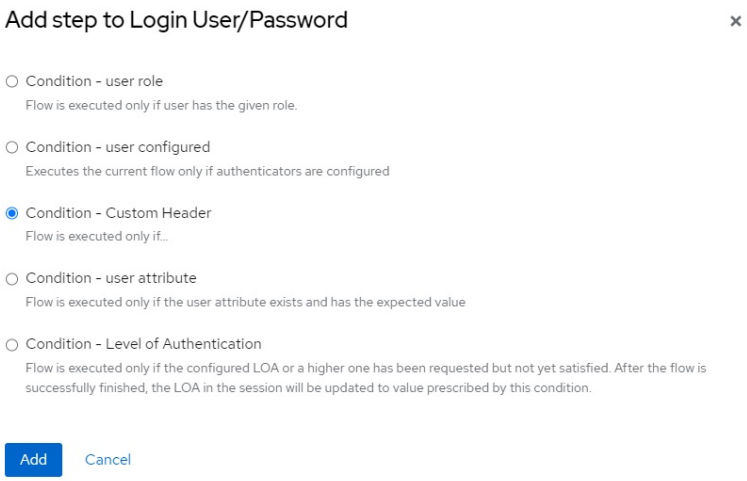
Anschließend erstellen Sie einen neuen sub-flow über den **"+" Button** und nennen diesen **Condition**.

Nach Anlage setzen Sie das zugehörige **Requirement** auf **Conditional**.

Klicken Sie nun auf den **"+" Button** dieses Eintrages und erstellen eine **Condition**.

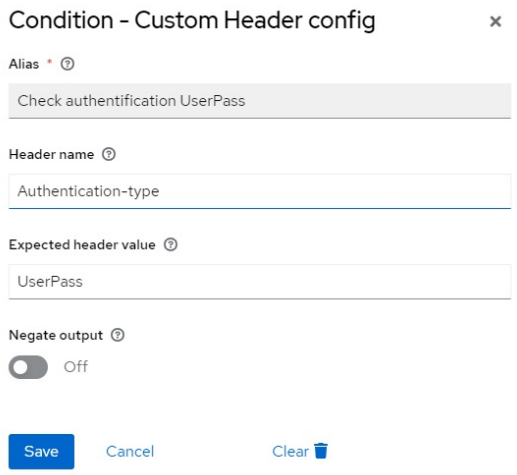


Im auftauchenden Auswahlmenü wählen Sie nun den Eintrag **Condition - Custom Header**, vergeben optional eine Beschreibung und klicken anschließend auf **Add:**



Nach Anlage setzen Sie das zugehörige **Requirement** auf **Required**.

Wechseln Sie dann über den Zahnrad-Button in das Einstellungsmenü und vergeben die nachfolgenden Parameter-Informationen:

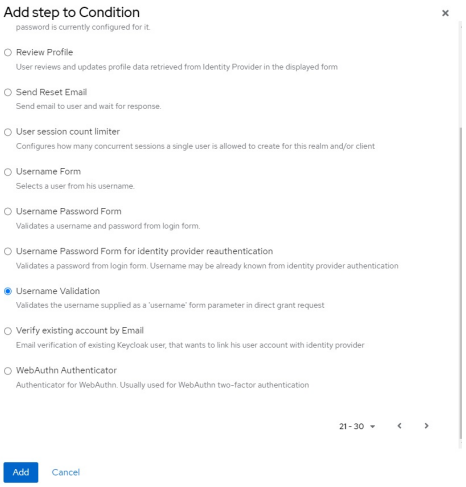


Parametername	**Auswahl**
Header name	Authentication-type
Expected header value	UserPass

Anschließend speichern Sie über **Save**.

Klicken Sie nun wieder auf den **"+" Button** des Condition-Sub-flows und legen hier einen Step an.

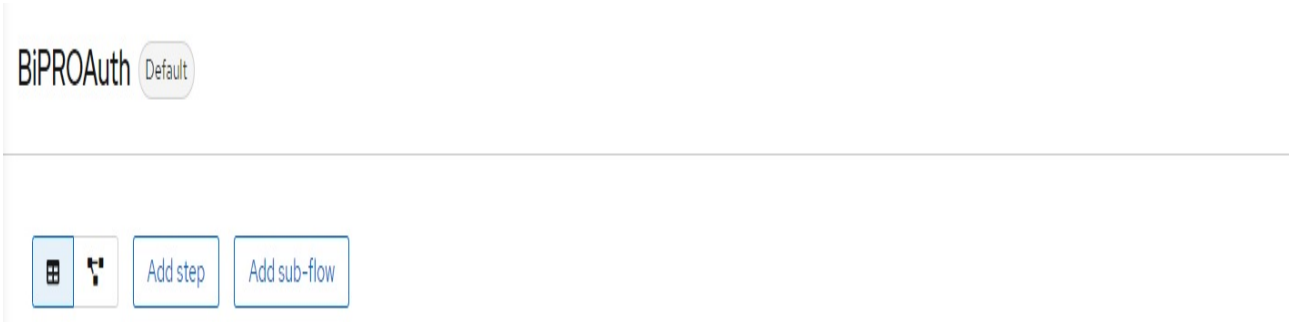
Wählen Sie im auftauchenden Auswahlmenü den Eintrag **Username Validation** und klicken anschließend auf **Add:**



Nach Anlage setzen Sie das zugehörige **Requirement** auf **Required**.

Damit ist der User Password Login angelegt.

X509 Login
Nun erstellen Sie einen grundlegend neuen sub-flow über den Button **Add sub-flow** im Hauptnavigationsberech des Haupt-Flows.



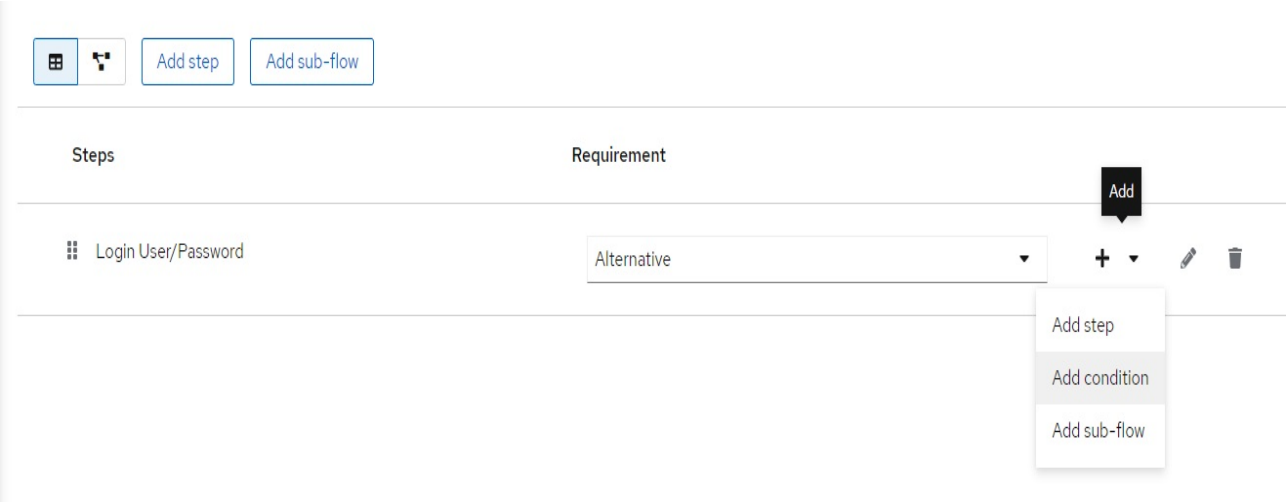
Diesen Sub-Flow nennen Sie **Login X509**.

Nachdem der neue sub-flow über **Add** angelegt wurde, klicken Sie nun auf das Dropdown-Menü für **Requirements** und setzen den Wert auf **Alternative**.

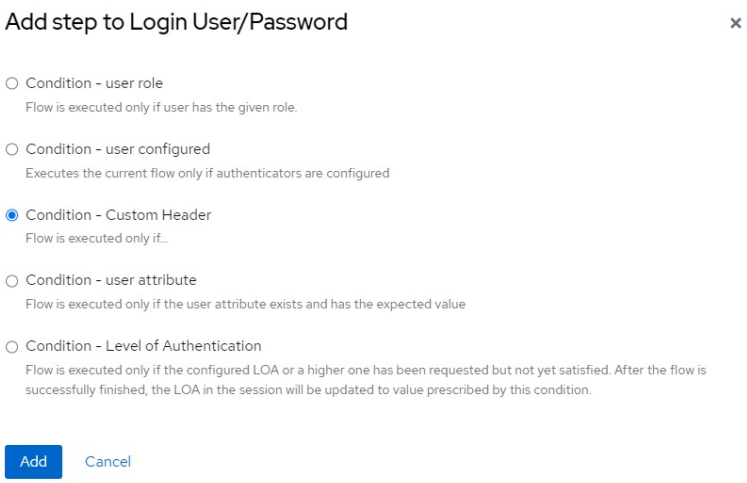
Anschließend erstellen Sie einen neuen sub-flow über den **"+" Button** und nennen diesen **Condition 2**.

Nach Anlage setzen Sie das zugehörige **Requirement** auf **Conditional**.

Klicken Sie nun auf den **"+" Button** dieses Eintrages und erstellen eine **Condition**.

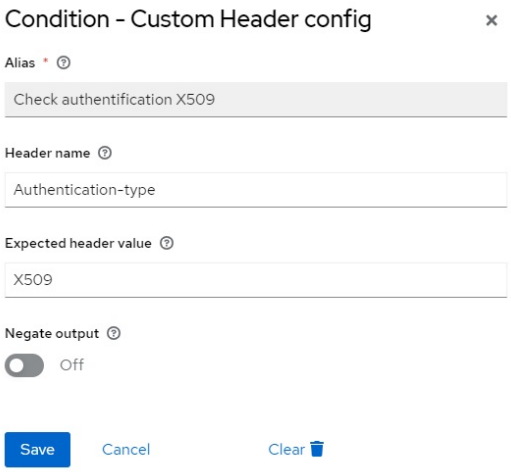


Im auftauchenden Auswahlmenü wählen Sie nun den Eintrag **Condition - Custom Header** und klicken anschließend auf **Add:**



Nach Anlage setzen Sie das zugehörige **Requirement** auf **Required**.

Wechseln Sie dann über den Zahnrad-Button in das Einstellungsmenü und vergeben die nachfolgenden Parameter-Informationen:

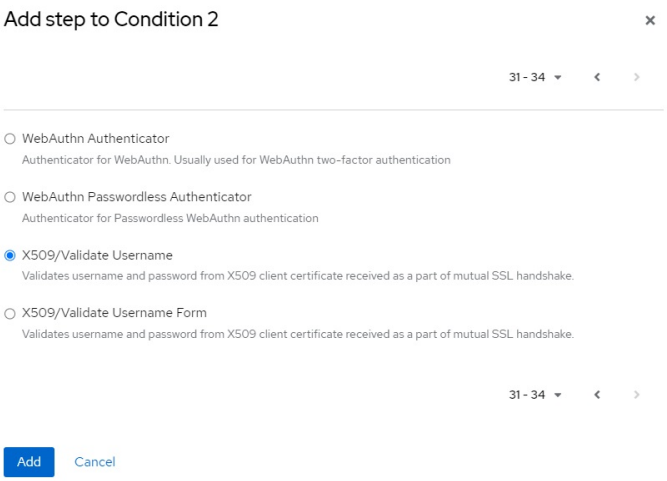


Parametername	Auswahl
Header name	Authentication-type
Expected header value	X509

Anschließend speichern Sie über **Save**.

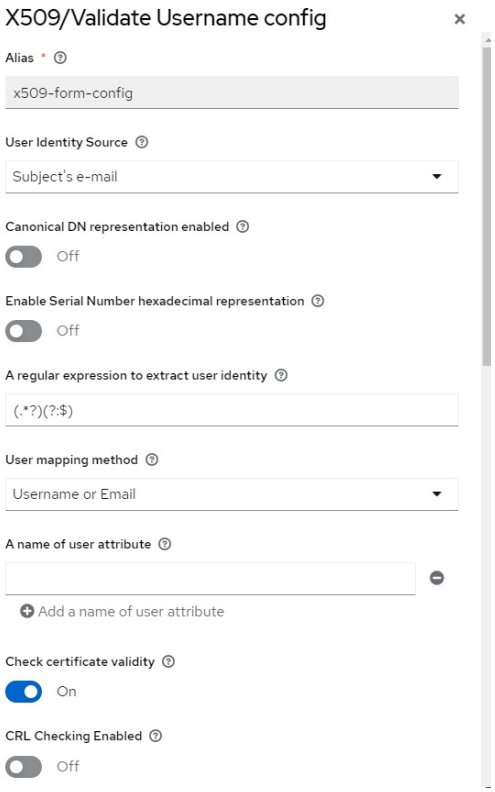
Klicken Sie nun wieder auf den **"+" Button** des Condition-Sub-flows und legen hier einen Step an.

Wählen Sie im auftauchenden Auswahlmenü den Eintrag **X509/Validate Username** und klicken anschließend auf **Add:**



Nach Anlage setzen Sie das zugehörige **Requirement** auf **Required**.

Wechseln Sie dann über den Zahnrad-Button in das Einstellungsmenü und vergeben die nachfolgenden Parameter-Informationen:



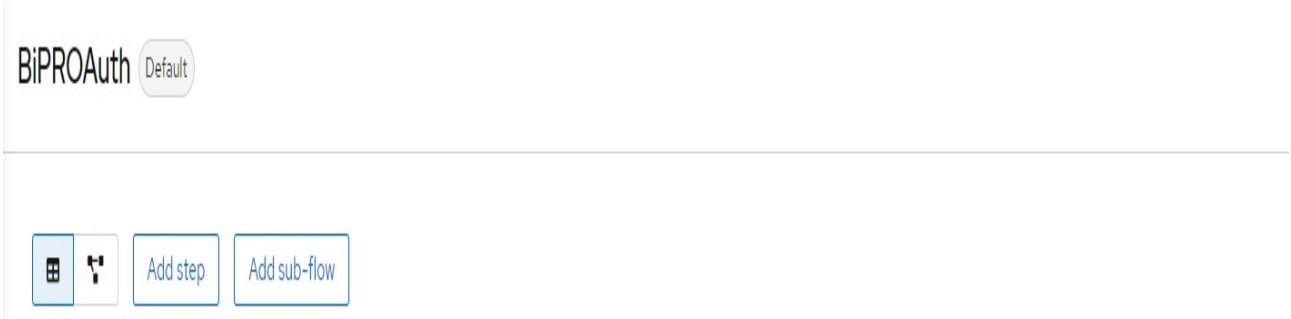
Parametername	Auswahl
User Identity Score	Subject's e-mail
User mapping method	Username or Email

Anschließend speichern Sie über **Save**.

Damit ist der X.509 Login angelegt.

Default Anmeldung
Zum Schluss muss noch eine "Default"-Anmeldevorgabe erstellt werden.

Dazu erstellen Sie wieder einen grundlegend neuen sub-flow über den Button **Add sub-flow** im Hauptnavigationsberech des Haupt-Flows.



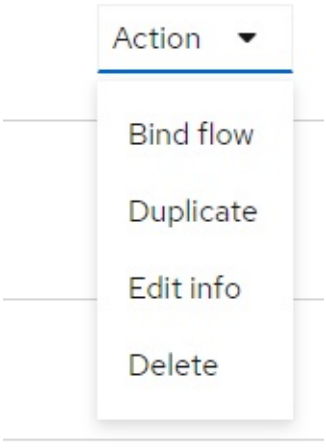
Diesen Sub-Flow nennen Sie **Login User Password Normal**.

Nachdem der neue sub-flow über **Add** angelegt wurde, klicken Sie nun auf das Dropdown-Menü für **Requirements** und setzen den Wert auf **Alternative**.

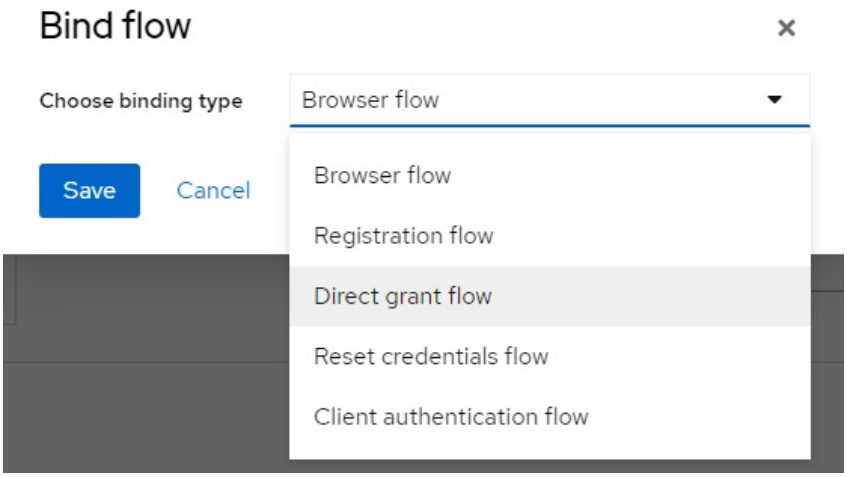
Klicken Sie nun auf den **"+" Button** des Sub-flows und legen hier einen Step an.

Wählen Sie im auftauchenden Auswahlmenü den Eintrag **Username Validation** und klicken anschließend auf **Add.**

Binden des Authentication Flows
Nachdem der Flow angelegt ist, muss der Flow nun noch an den Direct grant flow gebunden werden. Dazu klicken Sie im neu erstellten Flow unter **Action** auf **Bind flow**:



Im auftauchenden Auswahlmenü wählen Sie **Direct grant flow**.



Speichern Sie anschließend über **Save**.

Der neue Anmelde-Flow ist nun verbunden.

3.2.2 Modul BiPRO Norm 410 Security Token Service (Legacy)

Dieses Modul bietet eine "Out-of-the-box"-Lösung eines BiPRO Norm 410 Security Token Service zur Anbindung an eine relationale Datenbank.

Neben der standardmäßigen Service-Methode `RequestSecurityToken` wird auch eine generische **SecurityValidation**-Implementierung zur Verfügung gestellt.

3.2.2.1 Installation

Die Installation des Moduls erfolgt durch die Erstellung eines neuen ESB Projektes, das auf der zugehörigen Projektvorlage **Norm 410: STS Legacy** basiert.

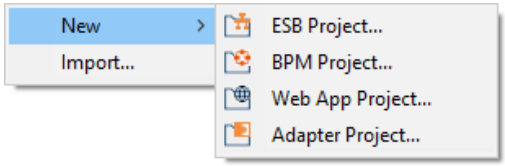
Nachdem das Projekt angelegt ist, erfolgt die Konfiguration des Moduls.

3.2.2.1.1 Vorgehen

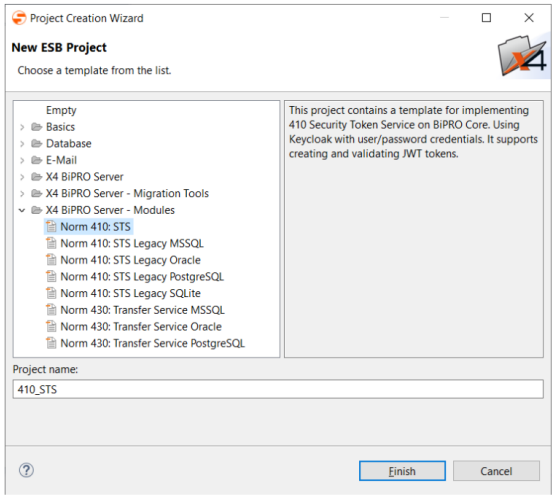
Wenn bereits ein 410 STS mit KeyCloak-Anbindung installiert wurde, so muss dieser vor Einrichtung des 410 STS Legacy entfernt werden, da es sonst zu Konflikten kommen kann.

1. Öffnen Sie den X4 Designer.

2. Klicken Sie im Repository rechts und wählen **New** > **ESB Project**.



3. Wählen Sie die Projektvorlage **Norm 410: STS Legacy** aus dem Ordner *X4 BiPRO Server - Modules* aus.



> Das Modul zur Implementierung und Bereitstellung eines BiPRO Norm 410 Security Token Service wird in verschiedenen Ausführungen angeboten:
>
> | Template Bezeichnung | Beschreibung |
> |---|---|
> | **Norm 410: STS Legacy MSSQL** | Modul zur Implementierung eines 410 Services mit allen Authentifizierungsmöglichkeiten unter Verwendung einer MSSQL Datenbank |
> | **Norm 410: STS Legacy Oracle** | Modul zur Implementierung eines 410 Services mit allen Authentifizierungsmöglichkeiten unter Verwendung einer Oracle Datenbank |
> | **Norm 410: STS Legacy PostgreSQL** | Modul zur Implementierung eines 410 Services mit allen Authentifizierungsmöglichkeiten unter Verwendung einer PostgreSQL Datenbank |
> | **Norm 410: STS Legacy SQLite** | Modul zur Implementierung eines 410 Services mit allen Authentifizierungsmöglichkeiten unter Verwendung einer SQLite Datenbank |

4. Geben Sie den Namen für das neue Projekt ein (Empfehlung: *410_STS*) und bestätigen Sie die Eingabe.

3.2.2.2 Konfiguration

3.2.2.2.1 Datenbank-Konfiguration

Nach Erstellung des Projektes muss die Datenbank und der Scheduler für das Modul konfiguriert werden.

Die Verbindung zwischen dem BiPRO Service und der gewünschten Datenbank wird über die Vergabe eines JNDI-Names und den zugehörigen Parametern erstellt.

Fügen Sie dazu den nachfolgenden **Wildfly Datasource**-Textblock in die **standalone.xml**-Datei (Konfigurationsdatei des Wildfly Servers) im Bereich **datasources** ein. Der Inhalt des Textblock ist abhängig von der verwendeten Datenbank.

Microsoft SQL	**Datasource-Eintrag für MSSQL**
	
	`<datasource jta="true" jndi-name="java:/MSSQLDS" pool-name="MSSQLDS" enabled="true" statistics-enabled="true">`
	`  <connection-url>jdbc:sqlserver://**HOSTNAME**:**MSSQL DATABASE PORT**;DatabaseName=**DATABASE NAME**;trustServerCertificate=true</connection-url>`
	`  <driver>sqlserver</driver>`
	`  <security>`
	`  <user-name>**DB USER**</user-name>`
	`  <password>**DB PASSWORD**</password>`
	`  </security>`
	`  <statement>`
	`  <prepared-statement-cache-size>32</prepared-statement-cache-size>`
	`  </statement>`
	`  <validation>`
	`  <check-valid-connection-sql>select 1</check-valid-connection-sql>`
	`  <validate-on-match>false</validate-on-match>`
	`  <background-validation>true</background-validation>`
	`  <background-validation-millis>1000</background-validation-millis>`
	`  </validation>`
	`  <timeout>`
	`  <allocation-retry>60</allocation-retry>`
	`  <allocation-retry-wait-millis>1000</allocation-retry-wait-millis>`
	`  </timeout>`
	`</datasource>`

Oracle Database	**Datasource-Eintrag für OracleDB**
	
	`<datasource jta="true" jndi-name="java:/OracleDS" pool-name="OracleDS" enabled="true" use-java-context="true">`
	`<connection-url>jdbc:oracle:thin:@localhost/**HOSTNAME**:**ORACLEDB DATABASE PORT**/**DATABASE NAME**</connection-url>`
	`<driver>oracle</driver>`
	`<security>`
	`<user-name>**DB USER**</user-name>`
	`<password>**DB USER PASSWORD**</password>`
	`</security>`
	`<statement>`
	`<prepared-statement-cache-size>32</prepared-statement-cache-size>`
	`</statement>`
	`<validation>`
	`<check-valid-connection-sql>select 1 from dual</check-valid-connection-sql>`
	`<validate-on-match>false</validate-on-match>`
	`<background-validation>true</background-validation>`
	`<background-validation-millis>1000</background-validation-millis>`
	`</validation>`
	`<timeout>`
	`<allocation-retry>60</allocation-retry>`
	`<allocation-retry-wait-millis>1000</allocation-retry-wait-millis>`
	`</timeout>`
	`</datasource>`

PostgreSQL	**Datasource-Eintrag für PostgreSQL**
	
	`<datasource jta="true" jndi-name="java:/PostgreSQL" pool-name="PostgreSQL" enabled="true" use-java-context="true">`
	`<connection-url>jdbc:postgresql://**HOSTNAME**:**POSTGRESQL DATABASE PORT**/**DATABASE NAME**</connection-url>`
	`<driver>postgresql</driver>`
	`<security>`
	`<user-name>**DB USER**</user-name>`
	`<password>**DB USER PASSWORD**</password>`
	`</security>`
	`<statement>`
	`<prepared-statement-cache-size>32</prepared-statement-cache-size>`
	`</statement>`
	`<validation>`
	`<check-valid-connection-sql>select 1</check-valid-connection-sql>`
	`<validate-on-match>false</validate-on-match>`
	`<background-validation>true</background-validation>`
	`<background-validation-millis>1000</background-validation-millis>`
	`</validation>`
	`<timeout>`
	`<allocation-retry>60</allocation-retry>`
	`<allocation-retry-wait-millis>1000</allocation-retry-wait-millis>`
	`</timeout>`
	`</datasource>`

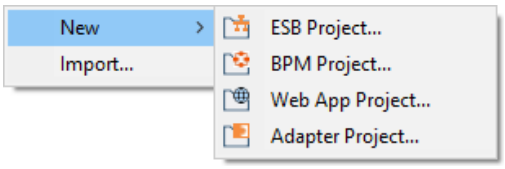
Passen Sie nun die folgenden Einträge an:

Parametername	Beschreibung
HOSTNAME	Hostname oder IP-Adresse des Datenbankservers
DATABASE PORT	Portnummer des Datenbankservers
DATABASE NAME	Eindeutiger Name der verwendeten Datenbank
DB USER	Datenbankbenutzer
DB USER PASSWORD	Passwort des Datenbankbenutzers

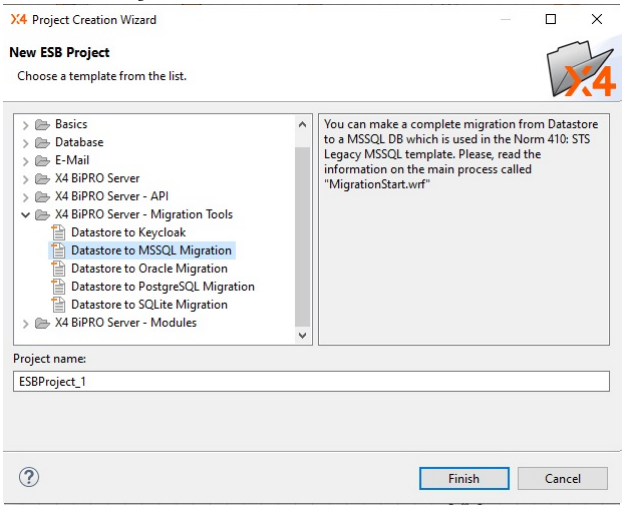
Starten Sie anschließend den X4 Server neu.

Öffnen Sie den X4 Designer.

Klicken Sie im Repository rechts und wählen **New** > **ESB Project**.



Wählen Sie die Projektvorlage **Datastore to (Datenbankname) Migration** aus dem Ordner *X4 BiPRO Server - Migration* aus.



Geben Sie den Namen für das neue Projekt ein und bestätigen Sie die Eingabe.

Starten Sie nun den Prozess **InitDatabase** unter **Processes/(Datenbankname)Queries.**

Die benötigte Datenbank samt zugehöriger Tabellen ist nun angelegt und kann verwendet werden.

Möchten Sie Bestände aus einer älteren BiPRO Server Version migrieren, so finden Sie die zugehörige Anleitung unter Migration.

3.2.2.3 Migration aus vorherigen BiPRO Server Versionen

Wenn Sie die Datenbestände eines älteren X4 BiPRO Servers auf die aktuelle Produktlinie migrieren möchten, kontaktieren Sie einen SoftProject-Consultant. Dieser unterstützt Sie gerne bei der Migration.
Für kundenseitig ausgeführte Upgrades kann SoftProject keine Gewähr übernehmen.

3.2.2.4 Authentifizierungsvarianten

3.2.2.4.1 User/Password

3.2.2.4.2 X.509 Authentifizierung

3.2.2.4.3 VDG Ticket Login

3.2.2.5 X4 BiPRO Token API

Die STS-Programmierschnittstelle unterstützt den Prozessentwickler dabei, die Token-Erstellung und -verwaltung des internen Security Token Servers (STS) von X4 BiPRO Server zu nutzen, wenn ein eigener Entity Storages (z. B. LDAP) angebunden werden soll.

3.2.2.5.1 Token-Erstellung (CreateToken)

Dieser Aufruf erstellt ein neues Token, mit dem anschließend eine Legitimierung möglich ist.

Elementname	Beschreibung	Mögliche Werte
`<LoginName>` (optional)	Der Name des Token-Ausstellers	Strings mit einer Länge von maximal 50 Zeichen
`<Organisation>` (optional)	Organisation, der der Token-Aussteller zugehörig ist (mehrere "Organisation"-Elemente unter dem Knoten "Organisationen" sind möglich)	Strings mit einer Länge von maximal 50 Zeichen
`<TokenType>`	Definiert, ob es sich um ein reguläres SecurityContext Token oder ein SAML Token handelt	• Default • SAML
`<AuthContext>`	Authenfizierungskontext des Services	• UserPassword • X.509 • VDG
`<Data>` (optional)	Unter diesem Knoten kann eine beliebige XML-Struktur mit dem Token persistiert werden	beliebige XML-Struktur

3.2.2.5.1.1 Beispiele

Folgende Beispiele sind im Testprozess `CreateTokenDemo` unter `/X4BiPROTokenAPI/Processes/Demo/Resources/Examples` enthalten:

Beispiel CreateToken.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<CreateToken
    xmlns="http://www.softproject.de/schemas/2018/sts"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:sts="http://www.softproject.de/schemas/2018/sts">

    <LoginName>TestUser</LoginName>
    <Organisations>
        <Organisation>10</Organisation>
        <Organisation>20</Organisation>
    </Organisations>
    <TokenType>Default</TokenType>
    <AuthContext>X.509</AuthContext>
    <Data>
        <!-- Arbitrary XML structure that'll be stored in the Token -->
        <Example>
            <Example>TestContent</Example>
            <ExampleElementWithChildren>
                <ExampleChildren>12</ExampleChildren>
                <ExampleChildren>23</ExampleChildren>
            </ExampleElementWithChildren>
        </Example>
    </Data>
</CreateToken>
```

Beispiel CreateTokenSaml.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<CreateToken
    xmlns="http://www.softproject.de/schemas/2018/sts"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:sts="http://www.softproject.de/schemas/2018/sts">

    <LoginName>TestUser</LoginName>
    <TokenType>SAML</TokenType>
    <AuthContext>X.509</AuthContext>
</CreateToken>
```

3.2.2.5.2 VDG-Ticket auslesen (DecodeVdgTicket)

Dieser Aufruf entschlüsselt ein VDG-Ticket aus der VDG-Anmeldung und gibt die ausgelesenen Informationen zurück. Dabei wird kein Token erzeugt. Mit diesem Aufruf können z. B. Informationen

gegen ein Benutzersystem wie LDAP geprüft werden. Je nach Ergebnis der Prüfung kann ein Aufruf von CreateToken erfolgen.

3.2.2.5.2.1 Beispiel

Folgende Beispiele für die Token-Erstellung sind unter `/X4BiPROTokenAPI/Resources/Examples` enthalten:

Beispiel DecodeVdgTicket.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<DecodeVdgTicket xmlns="http://www.softproject.de/schemas/2018/sts">
    <Ticket>
        <!-- Insert your Base64 ticket string here! -->
    </Ticket>
</DecodeVdgTicket>
```

3.2.2.5.3 Token-Aufhebung (RevokeToken)

Dieser Aufruf löscht ein ausgestelltes Token, so dass damit keine Legitimierung mehr möglich ist.

3.2.2.5.3.1 Beispiel

Folgendes Beispiel für die Token-Aufhebung ist unter `/X4BiPROTokenAPI/Resources/Examples` enthalten:

Beispiel RevokeToken.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<RevokeToken xmlns="http://www.softproject.de/schemas/2018/sts">
    <Token>
        <wsc:SecurityContextToken xmlns:wsc="http://schemas.xmlsoap.org/ws/2005/02/
sc">
            <wsc:Identifier>bipro:x509.Gz5QZcnDT9jFNnMEYPzPqtRsBkgyQ8</wsc:Identifier
>
        </wsc:SecurityContextToken>
    </Token>
</RevokeToken>
```

3.2.2.5.4 Token-Validierung (ValidateToken)

Mit diesem Aufruf kann ein Token validiert werden. In der Antwort werden Informationen zum Token bereitgestellt.

3.2.2.5.4.1 Beispiel

Folgende Beispiele für die Token-Validierung sind unter `/X4BiPROTokenAPI/Resources/Examples` enthalten:

Beispiel ValidateToken.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ValidateToken xmlns="http://www.softproject.de/schemas/2018/sts">
    <Token>
        <wsc:SecurityContextToken xmlns:wsc="http://schemas.xmlsoap.org/ws/2005/02/
sc">
            <wsc:Identifier>bipro:x509.DBeSpMCmxw6pQzeaQwj2tRczNfQp6R</wsc:Identifier
>
        </wsc:SecurityContextToken>
    </Token>
</ValidateToken>
```

Beispiel ValidateTokenSaml.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ValidateToken xmlns="http://www.softproject.de/schemas/2018/sts">
    <Token>
        <saml:EncryptedAssertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
            <xenc:EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
                <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#aes256-cbc" />
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <xenc:EncryptedKey
                        Recipient="CN=X4 BiPRO Server, O=SoftProject GmbH,
L=Ettlingen, ST=Baden-Württemberg, C=DE">
                        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/
xmlenc#rsa-1_5" />
                        <ds:KeyInfo>
                            <ds:X509Data>
                                <ds:X509Certificate>MIIDYjCCAkqgAwIBAgIEW51sCzANBgkqh
kiG9w0BAQsFADBzMQswCQYDVQQGEwJERTEbMBkGA1UECAwSQmFkZW4tV8O8cnR0ZW1iZXJnMRIwEAYDVQQHDA
lFdHRsaW5nZW4xGTAXBgNVBAoMEFNvZnRQcm9qZWN0IEdtYkgxGDAWBgNVBAMMD1g0IEJpUFJPIFNlcnZlcjA
eFw0xODA5MDEwMDAwMDBaFw0yMzA5MDEwMDAwMDBaMHMxCzAJBgNVBAYTAkRFMRswGQYDVQQIDBJCYWRlbi1X
w7xydHRlbWJlcmcxEjAQBgNVBAcMCUV0dGxpbmdlbjEZMBcGA1UECgwQU29mdFByb2plY3QgR21iSDEYMBYGA
1UEAwwPWDQgQmlQUk8gU2VydmVyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA142aCcw8nSjU7k
qNplNxeQ1jCCxTHAPpXcsGCRE4CUrHRDhTfsQO8LRdyLgsvHUg9jZvTxNQBOn3laN1xPyIo7wLu1FWMqbH+So
cfbTtHqMC5arEuALw1c/
oFCM+rBmKJYrBKcFvrgbkr+YhzoK0bC0Xli0hsYxzOr3rIrUIbk9uP41yI4jzA48cuz67ofPCZPd9SzQJIMA8
GNDJmlpY+UbRtfNEpfJmLY4GzghEl1QU57lSEv8UoL2Otcx4fVyqgXncgPDA14FOJFVn/
vxxZ+DsTu8UrfGPRKlq/
89GDzhadn78s8mT0p6No9ytgE2VVyE3yI+lqRWmtinuBM85MwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQC8Dc
iGEWA2zBWNC2xwpE9E57q0HmVsEJQVpUJ/
+DY7a4UeRpAmkZv1pL+wwPbqsg9bo6Lok2tPB+AEMnpyGERdOMLqyRQPI8gEOtYjmjDvrsnIZKu48O87bJhrj
/U/
wUTXRQJizl+yN+a6RjY8YOtopofqGVzB5PYaL5MWVRKq+QgTzOxlAObLQhpuvngSPQn81QLDHl6FjbeFDS/
d6FUjy/
CLUyNtU5uVs4OBTlQ+C4w89ZttD8MyjPXiJ33hR7k78rwkm4TBlZaCvtZj9mLy1i05kPalMY8XvAc3rWE7mRl
+X2PFfDsuJq3HoPhr8lR8k2YGUTt82pzrnGEWJt79
                                </ds:X509Certificate>
                            </ds:X509Data>
                        </ds:KeyInfo>
                        <xenc:CipherData>
                            <xenc:CipherValue>J3rp+OAlg5qGdYsVWOIra6ZmxMFvY2RFWOR4WAr
D50UOYqDDxLCJsvG/
3RkNvU5RX9J0YXNiWFrIx2n827ifT6BumAjgL8AwagH6dRpbxwnK2XstalT4BByKB8HQT7+QMhm9eNx19Zb4v
XAA9/pZjzpvikpRSNr5G+FtqQniJosSJgarDm2Fi+F3wr/nhbhk7/355E48eV1QQa+8XGXGu3fz8/
roUnYiUmvKCFrSnEOdF9j3rVAgElrIUf41ZFfGwGOgpRPEWK3uQUOcTADG6HGRFgsb61WLWVu56B1f2CRCbgH
KU8o51plt2mvHDCZr2iWB6H9QleP8y/KUGUY/Uw==
                            </xenc:CipherValue>
                        </xenc:CipherData>
                    </xenc:EncryptedKey>
                </ds:KeyInfo>
                <xenc:CipherData>
```

```
                    <xenc:CipherValue>ZzS9ksL9S8bSIhU/
cnQIFtvFMASE3rBw9KKT0WBH34yNmXP36Gq3H45rH3KIo2AODxyGazEUBkOzse3or2CY9L08pE6h6lp0SpTK1
BxUvJe9sdIK9Mfk4APqAp9WPYpwnro5QIs0IZ+3q+0KKyKyD8W14cRz9THy0YwjhP/
F2aeT4HkHmBj140xaR+Qqy00A14UXHwUYQqRT6WmCsZD61VdrlPCMCvzWrOsWi6qn8gcqYuVchZc4drZAKOwZ
6j2ub+8vamofiEMD6FQaOfgO1KB5QLOMFag51C8i6udbp0r07igXXhUDVEszhJK9oQcWprFJxF3AbSArSI2B1
MtPQl3R0bCHqkWSpQY5b9F+7v+EFM0b6we9FG0baO/
Ux1Jh5D8fbm4kXhknn7XzUOtCXBN1ygD7tqx0rTZ8l2Z8kR5NB5AOxJgtPwQWGpw21dNb3+vFY6IfJlEAh5MJ
hK81vbhbU2E2QS99EalhXgJ2GQABHX+2EdzQXnhyRd4GyGpdY3hHllJQzNsBsqo+A68ZQczxt6iGQ6vq/
EPS4qQSlhIrygd7VUKYilkB9Hx1IzHXFTFa6YV5qheKSrRwPD9kji4d23xnaGo/
Tf9hEVhjC6RLNCdVUfWwQGRIo5VKOTRNasQ5P1V7m82xOYpVvhg+vkuoFz3SbjD87E/
3a1WT5oIvoQxTKcHJJz0dY4WAPKRPWKrHDC//
b2CBOXxvMypboYc2kYdu6baMtyC08ihl+O5NyjoLRO5dO5iZCmMfsuO2umS6VhFhJMrTsx3ia9izRsudEJClL
BAM8UTClLEcnQ4tbCXws4CVtPnQREZFcUfGm4Ivd6RfSAZHzuMzu1YW6o9SWTb7cDaByBs66fddAXa7e26KUM
xZsu+C9HbrSDftDkcGX48SA6Ei2fTkMZ4iQ7KomlIWZD9LnVH2v7AkEbY7TbGh/
aq8SrZUa+dbsGV+ImLuuYnq/hKERop/vh+tZJCSJjJKFdJeFXkkg/
d6iAES2BoSkVSyEt3Ou8vyW0hR0UwK4anmcjgiOvo4LfXYU0NCgE6BVIQDJcqGM/
FP2sP9i7XtIKR6u2C60aVeaxNpmNr86gMd6/
oYT2ZqdpQGEmhxYqkcg5w50yH4hT9n7fXwZcRA+1RVQjs6roMCsGQQ0kTw2+H4yYQFU911O4ODK+XxGhqr7cI
SAK94xcEu5O+M1OoZBU0yhVgJ7ZoB6lh8AUnSPxEnBzn95r1IIvzH09ss5WaV4heHl8euEP8FetRvgb4s8sz/
iNVaOV7WzTMUIU1s6IFFqSjf/U10pZOnnJfPmKWCc4AEsF+pZ/gIju7dIBVrKZf/
aJuszXFXAj3GCSX8TH7XpHBNN6y+2Rgg6qj21V8mm+JaWfJWK/CXQV0JwJem/
2nagUWK1m411BpMQu0NUgR5iMCzHRKLMtliAPYJyfyTeGFZh7dZjx6bEBrE7GdQjlcUl+GvPJLQB/
AKoi9kqzzu7fZQcd9oegxLNI+OX8GR4TRwnBNJRI6FB+JYkw7k+5p3/8uDW1kb9jckx3JNXSYG6HWsJLrPgbk
y8srAMcP4LkQ+Bmub4ux88+mIQR0Hn8WMwq6hUgxtYccz5yOiddFMh5Z1ZiCJY9iH5mCdXNdFjk65I7vC1BAe
+NDVeSQgPYW7eAMpABUXh3pMAfJolqq97kv7NAhouOoVXN+v5wYUUmZEwzWS/
ozOHUu2gs14JvFilrlHhM6pUyeJof5alIZpuZz3Dtj+4BywbDW2PJC34LVcS7opBcc2MgFHiSv1NItD9uj1pt
115TQC8vIAO6k9BLv/
VGO+l7DKve5kgq122g1BPMOmTIYWHNDTtsVJKg4OOLszIVA3rHO+Guqxc1WzxWoMFzZ9iSUrPdaSjMaKS76rJ
KSEoVK0CWCBFyMQYZvmKeZC0HXnDQqphuo6hrK6fWROkBT5Uw3llIEYL3CpcivpewgZ9PIpEBtSzpU7Fo0MIe
vPkz81+82q+xQVvTP5ql8eHY9vLRTjgrUOQh53hgmWteXpzclyp9uc/
NbGZCFEFhOLCYyIGdFoaBflFetTtd1OmmVgsmxqR8RwSo2TIIsLYXwzlzlJYQx74t1u9c1OE23rENHWyMuzSR
RHnYDZ04ljD1Oz6dAzI+p6vD5yfWoBRoIeJGaso9tro5YEOUEZ5pcGrzNRT9f/
3OYiBK2kt7l2X9Sv4XdHpgGcvxbvdAN3cI40JLUPt/
PzQDykGaW7ZrUWtcHngTpXVws4UqXow5zvJYIDqQoaLRpl+I/
vVOIFkbY+M1ZbvKHgww4yMk+VHhX9PZmXmhyPScjffnqfQlHlkQQ9giYW4U1uG6RR2wRtKaC9F2qWpSomoy8C
UTKCVhQQOdSv7fWqAhsMzDrRiCUUKteHDOmGxsHxGe542Vq2KMDD8/6y5pOEaAaRZ01XCcyq4MOTcj7Kd8Nf7
5atLWFyy6WKK5X68noEtTfT9W8Zm2PArI4zYEJ5xy5mw5lg4CRMoExAHKKikYFNFFDgeCpH1Th9IHHul3Yey2
zKeTjBk0/trG8WhuaDG7VnfCJgwxKde98reAxiLm0Mwp+
+eAcQZpujBYCA0I6cmgXQIh4VouutLMWy+d99T02w9w4BiLBDQ1z0ns1urC4uxHVjWAL4+m0Bm4BUi00ynMKS
O3acLPamzXAjRtsuN/N4v5t9Rhr48Au1aAT1dywTi/CHEOSl7UrnadSf7nLtRpoOF0ASBNUbOC/
sAA4csUXa5mb8RJdel/
FVSOG04sH9xKEyQHaFTfKurXV2W9dPDxbC9tmySfkYcPqWMeT3HIWisAUP6XaFpDp7YZwZN8mfrdqI6kYpTLn
ZLtnvXKEP31gpvbmPoMRZqFMI9uJwjInyRJz2YpH+od/
BbYGv0CvIvFSt7MANtZCBP9zLQUQbBuOmKIkb5XFRmAXs9OC7oYGG0qRO83PlHu8ZF1E+xvqt5mXVwlEqLCvZ
EaynSk1khyaEtd271uyl6aojjPPAjebMuH0hjsUvW0+3L9UiHKDUGPwtj/loymW/uPWCXdqDkQW31OHtU/
EMeYlCRwX1tftUzGK8c7y4/RBJXpEAey0n8GOs4LuvHGlcCRWnFlmEer2uBTfJRjt4KqJPJV/
kzuKDPLcsA6XExUmNyMRsP8B9BXRC9WTyyfgNLM7OoEWrVQnmAMr1Js7DciShMNtk+RrjU/
dq56azoWa8jaoxS0ILkOVNJQOwRZ0FH4Q1wuU2cbQq7RvRFJAO+hwLBWeOFMfudZzArwXiH7LeNYrVDDv1171
8yhP2KKJbFwWtjELUS1eKQ1V2eo8RSXx1oGDqw7Hya6gv+RZuEMpgIH1LnhPF0yv9r1jRyqzlH4Uw0EO1Cp6p
Rx8YuT2cQfCy8o0/QlTINLF2nGIes/
W5dl7vfrgPcIu7GeHlludlw0jETvC89dw8ufEkbIPJbQGXRCC3KfmumyAWi15LwATX1NfFc17zoW7M6wLp5/
iZ19qQXcD+TnmYbeQntBOQDGBDGL2XPCD2tW/
tARcX8GMHlElvE3h3sHiuwFCnoztBvVHkFxhAlYu2tSq0RNfM/SqD8d1C/
vjUf9CmbM6TGaRt9GhrOSX3OwVUp8rZC/
L4nj9wWtmdtH6W1PH1z12XPmcau2UK4pAFJzl5vdz87uNHgLCknQieWB9Z2oUUYZasVXmgdtVNgtijln0vVXv
```

```
S8kpVhwOgZDRHkwimkmg0/CfmJR8uP+wfTaZ3TcG5bvtA0maan9a3cTgBCkf8AwuFTYVN/
OSNzyYgjGi+yl5XEe/
Fg9+QGAsevuLMqT+uvxTBZEOSYzkmcZ6326Q0ydKCujbDAwrN89ac+j3ar6a1ypJorUR/
zaczdoGtjHXobndUm/
S3ow0Whire+0PfVywRNI205JJ1LNgPmz2DiZytsP4Q2tL9VcRiHVEPsan7B2RmTh7L0QL3sNi08viHE+6iMM1
a6ZsKTHEkHG7Ifl8bgeFtrsNygwKa22pJUGE0t+MZLOY4z28e3afusbEK7ST0BmINMPZXcZnqT/
TMT74p4S18KQYgR2w6OzGkusucclHptp7FSgjhxB2Mjw5ZgkQtdI9WjdXK2lKNWjWz0Vj8C3FgIEE/
lWeZI2m0/I9/EyXW8Q8+ej0AXN3t1gBywjKhsfAxOUSBlMYk/
tVD14uagVjxdzTpPs22dA3awTXIz0ERNdf6kfQAoAZO3XeIw/RT8TzRRxEBZmxuHsaZJmU2CCpXltHCrI/
Ay7hJgb1rHZYSLJMQSrRCRR1HAlih+fjAs5AlCVKhLil1ete/
QJBjzMGavRyOZVi24Rwe8kPtEojZjNpuRm8wLMAuQDOWX+tt0fefIdusgMSA01p3ZWZfd9HrOnqHMwwxhQbvZ
9GT/KJwe52OS0Jw7HIHtJzhtX17CvdI0pMkVbhq1yxQq0K8n/
MdDju9Xs5q5BbOXu1h9YPBiPJhw3oDRngzqMJgBS3ftqQ7G4AXG0EX7SY+Bx2PTdqLlzqy082LzGhB+7BH2Ur
HtumTF+u1ecAAVo3G/cjnObGIxSscfzoAUr/
KQFi8nBIgUAHd33aLEuq+5zFYLpvLHrgpwCvOUOncJPlk6NKL3VkVFucZV7Q99ugMM2zjN9sP61wenovxs18O
Yw0ff7AmBs6QHlEZu/AHh6u7powggddPP1vekWVeC86DwL+91/
d6CJLLHlg3bUnvmVNLNSsVp3FB25DA7UykYNlho9KGSbCd89n7N7rz27/690vFTgzMkvBOkU9rAffkUNQKXOL
hvTKx51TIKCE+AfNV3wu9xALa/AjKpwJs5n/
TmHOcNF9e6nvNx0nSBqmZYzWc9R5onKuYpGvXXbGN6Hy4CZD1OVHNjrI7AIgJ95I1KmaQtT98CuFGP+BRyMgO
SsnBp9TH7piPQdFUeiT92R4lhVAYU6sFmkzz9MRpg4HCqWXvVmVys5Ln0jxQ4o0IGM1RYrwwrh5iqg==
                    </xenc:CipherValue>
                </xenc:CipherData>
            </xenc:EncryptedData>
        </saml:EncryptedAssertion>
    </Token>
</ValidateToken>
```

3.2.3 Modul BiPRO Norm 430 Transfer Service

Dieses Modul bietet eine vollständige "Out-of-the-box"-Lösung eines BiPRO Norm 430 Transfer Service.

Das Modul enthält alle Komponenten, die zur Erstellung und Bereitstellung eines vollwertigen BiPRO Norm 430 TransferService nötig sind.

Neben den standardmäßigen Service-Methoden `listShipments`, `getShipment` und `acknowledgeShipment` wird auch eine Importschnittstelle in Form eines X4 Adapters angeboten, mit der Lieferungen für den Consumer bereitgestellt werden können.

3.2.3.1 Installation

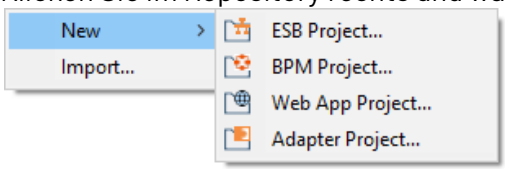
Die Installation des Moduls erfolgt durch die Erstellung eines neuen ESB Projektes, das auf der zugehörigen Projektvorlage **Norm 430: TransferService** basiert.

Nachdem das Projekt angelegt ist, erfolgt die Konfiguration des Moduls.

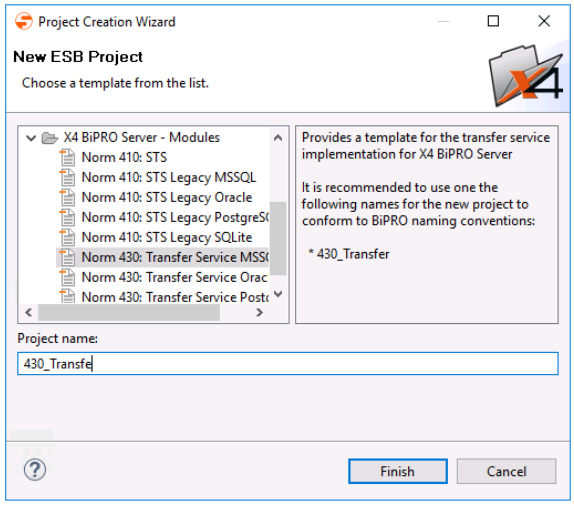
3.2.3.1.1 Vorgehen

1. Öffnen Sie den X4 Designer.

2. Klicken Sie im Repository rechts und wählen Sie **New** > **ESB Project**



3. Wählen Sie eine Projektvorlage **Norm 430: Transfer Service** aus dem Ordner *X4 BiPRO Server - Modules* aus.



> Es werden drei verschiedene Variationen des Norm 430: Transfer Service Template angeboten:
>
> | Template Bezeichnung | Beschreibung |
> |---|---|
> | **Norm 430: Transfer Service MSSQL** | Modul zur Implementierung eines 430 Services unter Verwendung einer MSSQL Datenbank |
> | **Norm 430: Transfer Service Oracle** | Modul zur Implementierung eines 430 Services unter Verwendung einer Oracle Datenbank |
> | **Norm 430: Transfer Service PostgreSQL** | Modul zur Implementierung eines 430 Services unter Verwendung einer PostgreSQL Datenbank |
>
> Die Vorlagen unterscheiden sich nur in den Anbindungsparametern für die Datenbank, sowie in der Syntax der verschiedenen SQL Statements.
> Die Prozesslogik ist bei allen Norm 430: Transfer Service Vorlagen identisch.

4. Vergeben Sie einen Namen für das neue Projekt (Empfehlung: *430_Transfer*) und klicken Sie auf **Finish**.

3.2.3.1.2 Service registrieren

Bevor der BiPRO Service genutzt werden kann, muss er zunächst über die zentrale Servicekonfiguration registriert werden.

Öffnen Sie dazu die Readme-Datei (**Readme.txt** unter Resources) des Moduls.

Hier befindet sich im Bereich "Service registration" ein **XML**-Textblock,

Textblock der Servicekonfiguration

```
    <Component href="xstore://430_Transfer/Resources/Service.xml" />
    <Group id="430_Transfer">

        <!-- Activates MTOM -->
        <Optimization behavior="default">
            <BinaryContainer local-name="Daten" namespace-uri="http://www.bipro.net/
namespace/allgemein" />
        </Optimization>

        <Endpoint id="Service" version="2.6.1" href="#430_Transfer" />
        <Endpoint id="Service" version="2.7.1" href="#430_Transfer" />
        <Endpoint id="Service" version="2.8.0" href="#430_Transfer" />
    </Group>
```

Kopieren Sie diesen in die Zwischenablage und fügen Sie ihn anschließend in die **Service.xml**-Datei (zentrale Service-Konfigurationsdatei unter *X4BiPRO/Resources*) unter das Root-Element <Component> ein und speichern Sie.

Angepasste Service.xml nach Einfügen der 430 Konfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<!--

    SERVICE.XML
    ===========

    This file contains the extension point for the main BiPRO service. It is the
location
    where BiPRO services are defined and configured.

    The service is structured in three layers: Groups, Endpoints and Implementations.
    This is due to the URL pattern all BiPRO services must implement:

        http://<hostname:port>/<path>/<group id>/<endpoint id>_<endpoint version>

    The implementation layer for SOAP services (BiPRO 2.x) is the service method.
    For example, you can declare a very simple (and incomplete) BiPRO 421 service
like this:

        <Group id="421_TAASUH">
            <Endpoint id="Service" version="2.6.1">
                <Implementation operation="getQuote">
                    ...
                </Implementation>
            </Endpoint>
        </Group>

    The version number can be one (X) to five (X.X.X.X.X) digit groups. The more
digit groups,
    the more specific the binding to a BiPRO version is. In the above example, the
service
    responds to the BiPRO version range "2.6.1.*.*"

 -->
<Component
    xmlns="http://www.softproject.de/schemas/2018/x4bipro"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.softproject.de/schemas/2018/x4bipro xstore://
X4BiPRO/Resources/Schemas/System/Service/Service.xsd">

    <Component href="xstore://430_Transfer/Resources/Service.xml" />
    <Group id="430_Transfer">

        <!-- Activates MTOM -->
        <Optimization behavior="default">
            <BinaryContainer local-name="Daten" namespace-uri="http://www.bipro.net/
namespace/allgemein" />
        </Optimization>

        <Endpoint id="Service" version="2.6.1" href="#430_Transfer" />
```

```
        <Endpoint id="Service" version="2.7.1" href="#430_Transfer" />
        <Endpoint id="Service" version="2.8.0" href="#430_Transfer" />
    </Group>


    <!-- TODO: Place your own service definitions here -->


</Component>
```

3.2.3.1.3 Datenbank-Konfiguration

Nach Erstellung des Projektes muss die Datenbank und der Scheduler für das Modul konfiguriert werden.

Die Verbindung zwischen dem BiPRO Service und der gewünschten Datenbank wird über die Vergabe eines JNDI-Names und den zugehörigen Parametern erstellt.

Öffnen Sie dazu die Readme-Datei (**Readme.txt** unter Resources) des Moduls.

Hier befindet sich im Bereich "Initial Database Setup" ein **Wildfly Datasource**-Textblock, welcher sich je nach verwendeter Datenbank syntaktisch unterscheidet.

Microsoft SQL	**Datasource-Eintrag für MSSQL**
	
	`<datasource jta="true" jndi-name="java:/BiPRO" pool-name="X4BIPRO" enabled="true" statistics-enabled="true">`
	`<connection-url>jdbc:sqlserver://**HOSTNAME**:**MSSQL DATABASE PORT**;DatabaseName=**DATABASE NAME**;trustServerCertificate=true</connection-url>`
	`<driver>sqlserver</driver>`
	`<transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>`
	`<pool>`
	`<min-pool-size>5</min-pool-size>`
	`<max-pool-size>20</max-pool-size>`
	`</pool>`
	`<security>`
	`<user-name>**DB USER**</user-name>`
	`<password>**DB PASSWORD**</password>`
	`</security>`
	`<validation>`
	`<valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>`
	`<check-valid-connection-sql>select 1</check-valid-connection-sql>`
	`<validate-on-match>true</validate-on-match>`
	`</validation>`
	`<statement>`
	`<prepared-statement-cache-size>20</prepared-statement-cache-size>`
	`<share-prepared-statements>true</share-prepared-statements>`
	`</statement>`
	`</datasource>`

Oracle Database	**Datasource-Eintrag für OracleDB**
	
	`<datasource jta="true" jndi-name="java:/BiPRO" pool-name="X4BIPRO" enabled="true" use-java-context="true">`
	`<connection-url>jdbc:oracle:thin:@localhost/**HOSTNAME**:**ORACLEDB DATABASE PORT**/**DATABASE NAME**</connection-url>`
	`<driver>oracle</driver>`
	`<security>`
	`<user-name>**DB USER**</user-name>`
	`<password>**DB USER PASSWORD**</password>`
	`</security>`
	`<statement>`
	`<prepared-statement-cache-size>32</prepared-statement-cache-size>`
	`</statement>`
	`<validation>`
	`<check-valid-connection-sql>select 1 from dual</check-valid-connection-sql>`
	`<validate-on-match>false</validate-on-match>`
	`<background-validation>true</background-validation>`
	`<background-validation-millis>1000</background-validation-millis>`
	`</validation>`
	`<timeout>`
	`<allocation-retry>60</allocation-retry>`
	`<allocation-retry-wait-millis>1000</allocation-retry-wait-millis>`
	`</timeout>`
	`</datasource>`

PostgreSQL	**Datasource-Eintrag für PostgreSQL**
	
	`<datasource jta="true" jndi-name="java:/BiPRO" pool-name="X4BIPRO" enabled="true" use-java-context="true">`
	`<connection-url>jdbc:postgresql://**HOSTNAME**:**POSTGRESQL DATABASE PORT**/**DATABASE NAME**</connection-url>`
	`<driver>postgresql</driver>`
	`<new-connection-sql>SET search_path TO x4bipro;</new-connection-sql>`
	`<pool>`
	`<min-pool-size>5</min-pool-size>`
	`<max-pool-size>20</max-pool-size>`
	`</pool>`
	`<security>`
	`<user-name>**DB USER**</user-name>`
	`<password>**DB USER PASSWORD**</password>`
	`</security>`
	`<statement>`
	`<prepared-statement-cache-size>20</prepared-statement-cache-size>`
	`<share-prepared-statements>true</share-prepared-statements>`
	`</statement>`
	`<validation>`
	`<check-valid-connection-sql>select 1</check-valid-connection-sql>`
	`<validate-on-match>false</validate-on-match>`
	`<background-validation>true</background-validation>`
	`<background-validation-millis>1000</background-validation-millis>`
	`</validation>`
	`</datasource>`

Passen Sie nun die folgenden Einträge an:

Parametername	Beschreibung
HOSTNAME	Hostname oder IP-Adresse des Datenbankservers
DATABASE PORT	Portnummer des Datenbankservers
DATABASE NAME	Eindeutiger Name der verwendeten Datenbank
DB USER	Datenbankbenutzer
DB USER PASSWORD	Passwort des Datenbankbenutzers

Kopieren Sie nun den angepassten datasources-Textblock in die Zwischenablage und fügen Sie ihn anschließend in die **standalone.xml**-Datei (Konfigurationsdatei des Wildfly Servers) im Bereich **datasources** ein und speichern Sie die Änderungen.

Starten Sie anschließend den X4 Server neu. Nach dem Neustart sollte die Verbindung zur Datenbank hergestellt sein.

Überprüfen Sie die Verbindung, in dem Sie die WSDL des Norm 430 TransferServices im Browser aufrufen (ersetzen Sie in nachfolgendem Link den Hostname und den Port mit der verwendeten Konfiguration):

http://localhost:8080/X4/httpstarter/ReST/BiPRO/430_Transfer/Service_2.6.1.1.0?wsdl

Wird die WSDL angezeigt, ist der TransferService nun verfügbar.

3.2.3.2 Scheduler

In der Modul-Scheduler-Datei werden bei der initialen Erstellung folgende Schedule-Prozesse definiert:

Name	Turnus	Beschreibung
InitialTransferServiceModule Setup	bei Serverstart	Initialisiert bei jedem Serverstart die benötigten Custom Placeholder für den TransferService
DropExpiredObjects	täglich	Entfernt alle abgelaufenen Lieferungen aus der Datenbank
EscalateShipments	alle 5 Minuten	Prüft, ob sich eine Lieferung im Eskalationszustand befindet

3.2.3.3 Konfiguration des Transfer Service

Die Ausgabe der importierten Daten kann in der Datei `/X4BiPRO/Resources/Transfers.xml` definiert werden, indem optionale Unternormen aktiviert, die Struktur der BiPRO-Lieferungen erweitert und die Anzahl der Geschäftsvorfälle limitiert werden.

Strukturübersicht Transfers.xml

```
<Component xmlns="http://www.softproject.de/schemas/2018/x4bipro">

    <Service xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <SchemaExtension>
            <!-- Import von Schemadateien zur Aktivierung optionaler Unternormen oder
providerspezifischer Schema-Erweiterungen -->
         </SchemaExtension>
    </Service>

    <Import>
        <ShipmentTemplate version="...">
            <!-- (Unvollständige) BiPRO-Lieferung-Struktur, die als Vorlage für
Lieferungen im Service verwendet wird.
        </ShipmentTemplate>
    </Import>
</Component>
```

3.2.3.3.1 Aktivierung optionaler Unternormen und Schema-Erweiterungen

Im Bereich `SchemaExtension` können Schema-Erweiterungen im Namespace `http://www.bipro.net/namespace/transfer` vorgenommen werden.

Hierzu steht die Syntax der XSD-Schemasprache zur Verfügung. Es ist sinnvoll, eigene Schemadateien zu importieren, um auch andere Namespaces abbilden zu können:

Strukturübersicht Transfers.xml

```
<Component xmlns="http://www.softproject.de/schemas/2018/x4bipro">
    <Service xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <SchemaExtension>
            <xsd:import namespace="http://www.bipro.net/namespace/transfer/provision"
 schemaLocation="${context.baseUrl}/${context.schemaPath}/${context.release}/bipro-
transfer-provision-${context.release.base}.xsd" />
         </SchemaExtension>
    </Service>
</Component>
```

Da diese Schemas über eine Webservice-Beschreibung (WSDL) bereitgestellt werden, müssen alle URL-Verweise extern erreichbar sein. Hierzu stehen die Platzhalter zur Verfügung, die auch in der Dienstkonfiguration verwendet werden können.

3.2.3.3.2 Erweiterung der Struktur für BiPRO-Lieferungen

Bei der Bereitstellung von BiPRO-Lieferungen werden vom System automatisch BiPRO-konforme XML-Strukturen aufgebaut. Diese können mit Hilfe von Lieferungsvorlagen erweitert werden:

Strukturübersicht Transfers.xml

```
<Component xmlns="http://www.softproject.de/schemas/2018/x4bipro">
    <Import>
        <ShipmentTemplate version="2.6.1">
            <tran:Lieferung xmlns:tran="http://www.bipro.net/namespace/transfer">
                <tran:Organisation xmlns:allg="http://www.bipro.net/namespace/
allgemein">
                    <allg:Name>Beispielorganisation</allg:Name>
                </tran:Organisation>
            </tran:Lieferung>
        </ShipmentTemplate>
    </Import>
</Component>
```

Im obigen Beispiel wird der Service mit der Schemaversion 2.6.1 um eine Struktur erweitert, die für jede Lieferung eine Organisation mit dem Namen `Beispielorganisation` ergänzt.

3.2.4 Modul X4 BiPRO Import (Demo Projekt)

Dieses Modul enthält ein Demo-Projekt zur Nutzung des neuen ImportProcess Adapters um Lieferungen für Vermittler im Norm 430 TransferService Kontext zur Verfügung zu stellen.

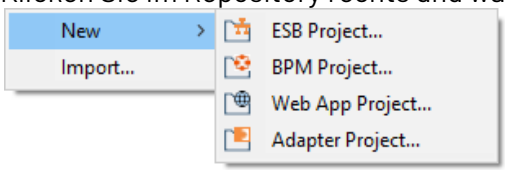
3.2.4.1 Installation

Die Installation des Moduls erfolgt durch die Erstellung eines neuen ESB Projektes, das auf der zugehörigen Projektvorlage **X4BiPROImportProcess** basiert.

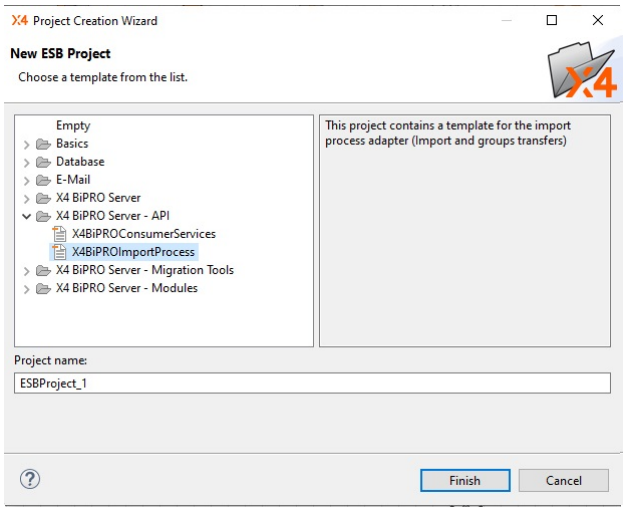
Nachdem das Projekt angelegt ist, erfolgt die Konfiguration des Moduls.

3.2.4.1.1 Vorgehen

1. Öffnen Sie den X4 Designer.
2. Klicken Sie im Repository rechts und wählen Sie **New** > **ESB Project**



3. Wählen Sie eine Projektvorlage **X4BiPROImportProcess** aus dem Ordner *X4 BiPRO Server - API* aus.



4. Vergeben Sie einen Namen für das neue Projekt und klicken Sie auf **Finish**.

3.2.4.2 Import

Der Import-Prozess des X4 BiPRO Server besteht aus zwei Teilen:

1. Importieren und Persistieren der Geschäftsvorfälle in die Datenbank als einzelne BiPRO-konforme Transfers
2. Gruppieren der Transfers zu Lieferungen

Um möglichst viele Architekturen abbilden zu können, können die Vorgänge unabhängig voneinander oder im Paket abgerufen werden.

`Import.wrf`	Nur Import und Persistierung der Geschäftsvorfälle als BiPRO-konforme Transfers
`ImportAndGroup.wrf`	Import, Persistierung der Geschäftsvorfälle und anschließende Gruppierung der betroffenen Datensätze
`Group.wrf`	Gruppierung der benannten Datensätze

3.2.4.2.1 Limitierung der Anzahl Geschäftsvorfälle (Transfers) pro BiPRO-Lieferung

Mithilfe des Attributes `maxShipmentSize`kann im Eingabe-XML für den API-Prozess festgelegt werden, wie viele Transfers maximal beim Import-Vorgang zu einer Lieferung zusammengefasst werden.

Beispiel für ein Eingabe-XML

```
<Batch maxShipmentSize="3">
    ...
</Batch>
```

3.2.5 Modul X4 BiPRO Consumer Services

In diesem Abschnitt werden die verschiedenen Consumer Services erläutert. Diese ermöglichen die Anbindung externer Webservices und Authentifizierungsverfahren, wie beispielsweise TGIC.

Das Modul enthält alle Komponenten, die zur Erstellung und Bereitstellung von BiPRO Consumer Services benötigt werden.

Diese ermöglichen es, externe BiPRO Server und die zugehörigen Services anzubinden.

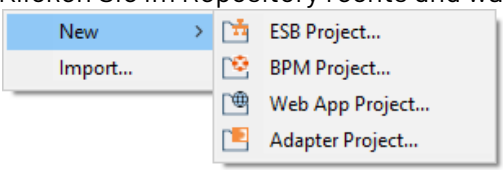
3.2.5.1 Installation

Die Installation des Moduls erfolgt durch die Erstellung eines neuen ESB Projektes, das auf der zugehörigen Projektvorlage **X4BiPROConsumerServices** basiert.

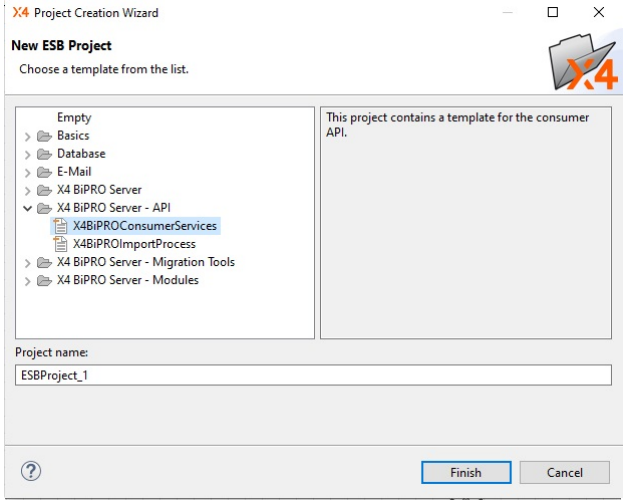
Nachdem das Projekt angelegt ist, erfolgt die Konfiguration des Moduls.

3.2.5.1.1 Vorgehen

1. Öffnen Sie den X4 Designer.
2. Klicken Sie im Repository rechts und wählen Sie **New** > **ESB Project**



3. Wählen Sie eine Projektvorlage **X4BiPROConsumerServices** aus dem Ordner *X4 BiPRO Server - API* aus.



4. Vergeben Sie einen Namen für das neue Projekt und klicken Sie auf **Finish**.

3.2.5.2 Call Service

Mit diesem Aufruf kann auf externe Webservices zugegriffen werden. Beim Zugriff werden die zugehörigen Authentifizierungsregeln und Parameter berücksichtigt.

Elementname	Beschreibung	Mögliche Werte
`<Url>`	Die URL des anzusprechenden Endpunktes	URL
`<Version>`	Die BiPRO-Version des anzusprechenden Endpunktes	BiPRO-Versionsnummer
`<Headers>` innerhalb von `<Endpoint>`	HTTP Header des anzusprechenden Endpunktes	Beliebiger String-Wert
`<Method>`	Name der auszulösenden Service-Methode	String-Wert (Name einer Service-Methode)
`<ConsumerId>`	Über dieses Feld kann dem extern aufgerufenen BiPRO-Service eine Consumer-ID übergeben werden	String-Wert
`<SkipOptimization>`	*Optional*. Mit diesem Element kann die Namespace-Optimierung de-/aktiviert werden.	• true • false
`<Token>`	In diesem Feld wird ein für die Nutzung des jeweiligen angesprochenen externen BiPRO Services zugehörige BiPRO Token eingefügt. Der Inhalt ist die zugehörige jeweilige WSC-konforme XML-Struktur	• SCT Token • SAML Token

Elementname	Beschreibung	Mögliche Werte
`<Headers>`	Die Security Header, die für den angesprochenen BiPRO-Service verlangt werden.	XML-Struktur des jeweiligen Headers
`<Body>`	In diesem Feld wird die aufzurufende Service-Methode des angesprochenen BiPRO-Services eingefügt.	BiPRO-konforme XML-Struktur der jeweiligen Service-Methode

3.2.5.2.1 Beispiel

Folgendes Beispiel ist im Testprozess `CallServiceDemo` unter `/Processes/Demo/Resources/Examples` enthalten:

Beispiel ServiceDemo.xml

```
<Request xmlns="http://www.softproject.de/schemas/2018/consumer">

    <Endpoint>
        <Url>http://localhost:8080/X4/httpstarter/ReST/BiPRO/430_Transfer/
Service_2.6.1.1.0</Url>
        <Version>2.6.1.1.0</Version>

        <Headers>
            <X-Custom-Header>Custom header value</X-Custom-Header>
            <X-Another-Custom-Header>Yet another custom header value</X-Another-
Custom-Header>
        </Headers>
    </Endpoint>
    <Method xmlns:tran="http://www.bipro.net/namespace/transfer">tran:listShipments</
Method>

    <Settings>
        <ConsumerId>X4 BiPRO Consumer Services</ConsumerId>
        <!-- OPTIONAL: Include this setting to skip XML namespace optimization. -->
        <!-- <SkipOptimization>true</SkipOptimization> -->
    </Settings>

    <Token>
        <!-- SCT or SAML-Token -->
        <wsc:SecurityContextToken xmlns:wsc="http://schemas.xmlsoap.org/ws/2005/02/
sc">
            <wsc:Identifier>bipro:up.adhajy9JwweFTTB58H9Q7KyftHrtR7</wsc:Identifier>
        </wsc:SecurityContextToken>
    </Token>

    <Headers>
        <example:Test xmlns:example="urn:some.example.namespace">
            <example:Value>Test</example:Value>
        </example:Test>
    </Headers>

    <Body>
        <tran:listShipments xmlns:tran="http://www.bipro.net/namespace/transfer">
            <tran:Request />
        </tran:listShipments>
    </Body>

</Request>
```

3.2.5.3 Get Token from Certificate

Mit diesem Aufruf kann auf einen externen SecurityToken-Webservice zugegriffen werden, um ein BiPRO-Token mithilfe eines Zertifikats (X.509 bzw. VDG) auszustellen.

Elementname	Beschreibung	Mögliche Werte
`<Url>`	Die URL des anzusprechenden Endpunktes	URL
`<Version>`	Die BiPRO-Version des anzusprechenden Endpunktes	BiPRO-Versionsnummer
`<Headers>`	HTTP Header des anzusprechenden Endpunktes	Beliebiger String-Wert
`<KeystoreId>`	ID des Keystores/Schlüsselspeichers, in dem das zu nutzende X.509-Zertifikat als Key/Schlüsselpaar gespeichert ist. ⓘ Der Keystore mit der dazugehörigen Keystore-ID wird in der Datei `Security.xml` im Verzeichnis `X4BiPRO` definiert.	In der Datei `Security.xml` vergebene ID (Element `<NamedKeystore>`, Attribut `id`)
`<KeyId>`	ID des Keys/Schlüsselpaars, in dem das genutzte X.509-Zertifikat gespeichert ist ⓘ Die Key-ID wird im dazugehörigen Keystore in der Datei `Security.xml` im Verzeichnis `X4BiPRO` definiert.	In der Datei `Security.xml` vergebene ID (Element `<NamedKey>`, Attribut `id`)
`<TokenType>`	Definiert, ob es sich um ein reguläres SecurityContext-Token oder ein SAML-Token handelt	• Default • SAML
`<DigestMethod>`	Definiert die Digest-Methode für den Request	• SHA1 (bei veralteten Services) • SHA256 (wird von TGIC genutzt) • SHA512
`<SignatureMethod>`	Definiert den Signaturalgorithmus für den Request	• SHA1 (bei veralteten Services) • SHA256 (wird von TGIC genutzt) • SHA512

Elementname	Beschreibung	Mögliche Werte
`<CanonicalizationMethod>`	Definiert den Kanonisierungsalgorithmus, der auf die SignedInfo angewendet wird, um die Daten zu signieren	• http://www.w3.org/TR/2001/REC-xml-c14n-20010315 • http://www.w3.org/2001/10/xml-exc-c14n# (wenn TGIC verwendet wird)
`<MustUnderstand>`	*Optional*. Mit diesem Feld kann gesteuert werden, ob die Verarbeitung der Header Pflicht ist.	• true • false
`<GenerateMessageId>`	*Optional*. Mit diesem Feld kann gesteuert werden, ob eine zufällige Message-ID erzeugt werden soll.	• true • false
`<Context>`	*Optional*. Mit diesem Feld kann der Kontext des jeweiligen Requests explizit bestimmt werden	URI
`<AppliesTo>`	*Optional*. Mit diesem Feld kann explizit der Bereich, zu dem das Token gehört, explizit festgelegt werden.	URI
`<SkipOptimization>`	*Optional*. Mit diesem Element kann die Namespace-Optimierung de-/aktiviert werden.	• true • false

3.2.5.3.1 Beispiel

Folgendes Beispiel ist im Testprozess `GetTokenFromCertificateDemo` unter `/Processes/Demo/Resources/Examples` enthalten:

Beispiel CertificateLoginDemo.xml

```
<GetTokenByCertificate xmlns="http://www.softproject.de/schemas/2018/consumer">

    <Endpoint>
        <Url>http://localhost:8080/X4/httpstarter/ReST/BiPRO/410_STS/
X509Login_2.6.1.1.0</Url>
        <Version>2.6.1.1.0</Version>

        <Headers>
            <X-Custom-Header>Custom header value</X-Custom-Header>
            <X-Another-Custom-Header>Yet another custom header value</X-Another-
Custom-Header>
        </Headers>
    </Endpoint>

    <Credentials>
        <!-- These identifiers point to this file: xstore://X4BiPRO/Resources/
Security.xml -->
        <KeystoreId>consumer:x509:demo</KeystoreId>
        <KeyId>consumer:x509:demo:login</KeyId>
    </Credentials>

    <!-- Other alternative: SAML -->
    <TokenType>Default</TokenType>

    <!-- It is recommended to use SHA-256 for best compatibility. Use SHA-1 for older
services. -->
    <DigestMethod>SHA256</DigestMethod>

    <!-- It is recommended to use SHA-256 for best compatibility. Use SHA-1 for older
services. TGIC requires SHA256. -->
    <SignatureMethod>SHA256</SignatureMethod>

    <!-- W3C canonicalization method - keep the default value wherever possible: -->
    <CanonicalizationMethod>http://www.w3.org/TR/2001/REC-xml-c14n-20010315</
CanonicalizationMethod>

    <!-- ... or use this for TGIC services:
    <CanonicalizationMethod>http://www.w3.org/2001/10/xml-exc-c14n#</
CanonicalizationMethod> -->

    <!-- OPTIONAL: This flag indicates whether processing of the header is optional
('false') or mandatory ('true') -->
    <!-- <MustUnderstand>true</MustUnderstand> -->

    <!-- OPTIONAL: This flag indicates whether a random unique message ID should be
generated ('true') or not ('false') -->
    <!-- <GenerateMessageId>true</GenerateMessageId> -->

    <!-- OPTIONAL: This optional URI specifies an identifier/context for this
request. -->
    <!-- <Context>http://www.softproject.de/BiPRO</Context> -->
```

```
    <!-- OPTIONAL: This optional URI specifies the scope for which this security
token is desired. -->
    <!-- <AppliesTo>http://www.softproject.de/BiPRO/ConsumerServices/</AppliesTo> -->

    <!-- OPTIONAL: Include this setting to skip XML namespace optimization. -->
    <!-- <SkipOptimization>true</SkipOptimization> -->
</GetTokenByCertificate>
```

3.2.5.4 Get Token from UserPassword

Mit diesem Aufruf kann auf einen externen SecurityToken-Webservice zugegriffen werden, um ein BiPRO-Token mithilfe eines Benutzernamens und dem dazugehörigen Passwort auszustellen (Username/Password Authenticiation).

Elementname	Beschreibung	Mögliche Werte
`<Url>`	URL des anzusprechenden Endpunktes	URL
`<Version>`	BiPRO-Version des anzusprechenden Endpunktes	BiPRO-Versionsnummer
`<Headers>`	HTTP Header des anzusprechenden Endpunktes	Beliebiger String-Wert
`<Credentials>`	Anmeldedaten des Benutzers, der ein Token erhalten soll	Strings mit einer Länge von maximal 50 Zeichen
`<TokenType>`	Definiert, ob es sich um ein reguläres SecurityContext-Token oder ein SAML-Token handelt	• Default • SAML
`<MustUnderstand>`	*Optional*. Mit diesem Feld kann gesteuert werden, ob die Verarbeitung der Header Pflicht ist.	• true • false
`<GenerateMessageId>`	*Optional*. Mit diesem Feld kann gesteuert werden, ob eine zufällige Message-ID erzeugt werden soll.	• true • false
`<Context>`	*Optional*. Mit diesem Feld kann der Kontext des jeweiligen Requests explizit bestimmt werden	URI
`<AppliesTo>`	*Optional*. Mit diesem Feld kann explizit der Bereich, zu dem das Token gehört, explizit festgelegt werden.	URI
`<SkipOptimization>`	*Optional*. Mit diesem Element kann die Namespace-Optimierung de-/aktiviert werden.	• true • false

3.2.5.4.1 Beispiel

Folgendes Beispiel ist im Testprozess `GetTokenFromUserPasswordDemo` unter `/Processes/Demo/Resources/Examples` enthalten:

Beispiel UserPasswordLoginDemo.xml

```
<GetTokenByUsernamePassword xmlns="http://www.softproject.de/schemas/2018/consumer">

    <Endpoint>
        <Url>http://localhost:8080/X4/httpstarter/ReST/BiPRO/410_STS/
UserPasswordLogin_2.6.1.1.0</Url>
        <Version>2.6.1.1.0</Version>

        <Headers>
            <X-Custom-Header>Custom header value</X-Custom-Header>
            <X-Another-Custom-Header>Yet another custom header value</X-Another-
Custom-Header>
        </Headers>
    </Endpoint>

    <Credentials>
        <Username>demo</Username>
        <Password>test1234</Password>
    </Credentials>

    <!-- Other alternative: SAML -->
    <TokenType>Default</TokenType>

    <!-- OPTIONAL: This flag indicates whether processing of the header is optional
('false') or mandatory ('true') -->
    <!-- <MustUnderstand>true</MustUnderstand> -->

    <!-- OPTIONAL: This flag indicates whether a random unique message ID should be
generated ('true') or not ('false') -->
    <!-- <GenerateMessageId>true</GenerateMessageId> -->

    <!-- OPTIONAL: This optional URI specifies an identifier/context for this
request. -->
    <!-- <Context>http://www.softproject.de/BiPRO</Context> -->

    <!-- OPTIONAL: This optional URI specifies the scope for which this security
token is desired. -->
    <!-- <AppliesTo>http://www.softproject.de/BiPRO/ConsumerServices/</AppliesTo> -->

    <!-- OPTIONAL: Include this setting to skip XML namespace optimization. -->
    <!-- <SkipOptimization>true</SkipOptimization> -->

</GetTokenByUsernamePassword>
```

3.2.5.5 Sign Request

Dieser Aufruf ermöglicht das Signieren von Requests und X.509 Zertifikaten von externen SecurityToken Webservices.

Elementname	Beschreibung	Mögliche Werte
`<KeystoreId>`	ID des Keystores / Schlüsselspeichers, in dem das zu nutzende X.509 Zertifikat als Key / Schlüsselpaar gespeichert ist. Der Keystore mit zugehöriger KeystoreId wird in der Datei Security.xml im Verzeichnis X4BiPRO definiert	In der Datei Security.xml vergebene ID (Element "NamedKeystore", Attribut "id")
`<KeyId>`	ID des Keys / Schlüsselpaares, in dem das genutzte X.509 Zertifikat gespeichert ist Die KeyId wird im zugehörigen Keystore in der Datei Security.xml im Verzeichnis X4BiPRO definiert	In der Datei Security.xml vergebene ID (Element "NamedKey", Attribut "id")
`<DigestMethod>`	Die Digest Methode für den Request	• SHA1 (bei veralteten Services) • SHA256 (wird von TGIC genutzt) • SHA512
`<SignatureMethod>`	Der Signaturalgorithmus für den Request	• SHA1 (bei veralteten Services) • SHA256 (wird von TGIC genutzt) • SHA512
`<CanonicalizationMethod>`	Der Kanonisierungsalgorithmus, der auf die SignedInfo angewendet (zur Signierung der Daten)	• http://www.w3.org/TR/2001/REC-xml-c14n-20010315 • http://www.w3.org/2001/10/xml-exc-c14n# (bei der Verwendung von TGIC)
`<ExpiresInMinutes>`	Mit diesem Feld kann gesteuert werden, wie lange die Signierung gültig ist	Integer-Wert
`<Envelope>`	In diesem Bereich wird der zu signierende SOAP-Envelope eingefügt	Ein SOAP-Envelope

3.2.5.5.1 Beispiel

Folgendes Beispiel ist im Testprozess `SignRequest` unter `/Demo/Resources/Examples` enthalten:

Beispiel SignDemo.xml

```
<SignRequest xmlns="http://www.softproject.de/schemas/2018/consumer">

    <Credentials>
        <!-- These identifiers point to this file: xstore://X4BiPRO/Resources/
Security.xml -->
        <KeystoreId>consumer:x509:demo</KeystoreId>
        <KeyId>consumer:x509:demo:login</KeyId>
    </Credentials>

    <!-- It is recommended to use SHA-256 for best compatibility. Use SHA-1 for older
services. -->
    <DigestMethod>SHA256</DigestMethod>

    <!-- It is recommended to use SHA-256 for best compatibility. Use SHA-1 for older
services. TGIC requires SHA256. -->
    <SignatureMethod>SHA256</SignatureMethod>

    <!-- W3C canonicalization method - keep the default value wherever possible: -->
    <CanonicalizationMethod>http://www.w3.org/TR/2001/REC-xml-c14n-20010315</
CanonicalizationMethod>

    <!-- ... or use this for TGIC services:
    <CanonicalizationMethod>http://www.w3.org/2001/10/xml-exc-c14n#</
CanonicalizationMethod> -->

    <!-- Optional, default is 10 Minutes. Must contain an integer number -->
    <ExpiresInMinutes>10</ExpiresInMinutes>

    <!-- Must contain exactly one SOAP envelope to sign -->
    <Envelope>
        <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" >
            <soap:Header>
                <!-- Signature will be placed here -->
            </soap:Header>
            <soap:Body wsu:Id="body">
                <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/
2005/02/trust">
                    <wst:TokenType>http://schemas.xmlsoap.org/ws/2005/02/sc/sct</wst:
TokenType>
                    <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/
Issue</wst:RequestType>
                </wst:RequestSecurityToken>
            </soap:Body>
        </soap:Envelope>
    </Envelope>

</SignRequest>
```