ISZR Referenční agent JAVA

Název dokumentu:	ISZR Referenční agent JAVA	**Verze:**	1.02
Autor:	AUTOCONT a.s.	**Datum aktualizace:**	01.02. 2023
Účel:	Popis aplikace Referenčního agenta	**Počet stran:**	9

Obsah

1 Účel dokumentu

Dokument popisuje aplikaci referenčního agenta, která slouží jako návod a demonstrace pro implementaci a technické řešení klienta, který je implementovaný v jazyce Java.

1.1 Slovník Pojmů

Pojem/Zkratka	Význam
IntelliJ IDEA	Vývojové prostředí, které umožňuje vyvíjet aplikace (nejen v jazyce Java).
Java	Programovací jazyk.
Java SDK	Distribuce nástrojů pro vývoj aplikací v programovacím jazyku Java.
jax-ws	Rozhraní v jazyce Java pro volání a vytváření webových služeb.
keytool	Nástroj, který je distribuovaný s Java SDK. Je určený ke správě databáze certifikátů (key store).
Maven2	Apache Maven2 je nástroj pro řízení projektu. Je založený na konceptu „project model object“. Maven se používá k řízení procesu sestavení projektu, testování a reportování.
SOAP	Komunikační protokol pro výměnu zpráv, které jsou ve formátu XML.
STUB	Objekt, který je u klienta webové služby a který tuto webovou službu u klienta zastupuje. Klient volá webovou službu přes tento objekt. Implementace objektu je vygenerovaná na základě wsdl souboru.
wsimport	Nástroj, který je distribuovaný v rámci instalace Java SDK. Je určený k vygenerování STUBu na základě wsdl.
wsdl	Soubor WSDL popisuje webovou službu, tzn.: metody, typy parametrů, návratovou hodnotu, výjimky….
XSD	Technologie, která umožňuje definovat formát XML souboru.
SSL	Socket security layer – komunikační vrstva (služba), která transparentně kóduje síťovou komunikaci.
CA	Certifikační autorita – prvek organizační struktury, který vydává certifikáty a je odpovědný za jejich platnost. Certifikační autority tvoří stromovou strukturu.
Root CA	Certifikační autorita, která stojí nejvýše v hierarchii. Vydává pouze certifikáty pro podřízené certifikační autority.
Sub CA	Certifikační autorita, která je podřízena kořenové CA. Tato CA vydává jednotlivé certifikáty

Certifikát	Datová struktura, která identifikuje zařízení, aplikaci, fyzickou nebo právnickou osobu.
Serverový certifikát	Certifikát, kterým prokazuje server svou identitu klientům.
Klientský certifikát	Certifikát, kterým prokazuje klient svou identitu danému serveru.

2 Copyright a záruka

Zdrojový kód aplikace i přeloženou aplikaci lze použít libovolným způsobem. Tvůrce aplikace a ostatních částí (aplikace, zdrojové kódy, dokumentace, certifikát, ostatní soubory a další) není v žádném ohledu odpovědný za jakýkoliv důsledek přímo nebo nepřímo vzniklý v souvislosti s libovolnou částí díla.

3 Vývojové prostředí

Vývojové prostředí se skládá z:

- Java 17 nebo novější
- Maven2 2.2.1 nebo novější
- IDE IntelliJ IDEA nebo novější
- Apache Ant 1.8.2 nebo novější
- Internetové připojení

3.1 Konfigurace vývojového prostředí

- Java JDK 17
    - Nainstaluj Java SDK do systému (C:\programs\java).
    - Vytvoř systémovou proměnnou JAVA_HOME – Adresář, kam byla nainstalovaná Java (C:\programs\java\jdk-17.0.2)
    - Na %JAVA_HOME%\bin na systémovou PATH
- Maven2 2.2.1
    - Nainstaluj Maven2 do systému (C:\programs\m2).
    - Vytvoř systémovou proměnnou M2_HOME -adresář, kam byl nainstalovaný Maven2 (C:\programs\m2\apache-maven-2.2.1)
    - Nastav systémovou proměnnou M2 na %M2_HOME%\bin
        - Přidat %M2% na systémovou PATH

- Vytvoř systémovou proměnnou MAVEN_OPTS – umožňuje vložit systémové parametry Javy do build procesu. V tomto procesu se budou generovat STUBy pro komunikaci mezi klientem a webovou služnou. Aby tyto soubory byly v kódování UTF-8, je nutné přidat do této proměnné následující hodnotu: -Dfile.encoding=UTF-8
- Upravit soubor setting.xml tak, aby Maven2 byl schopen se spojit s „artefactory“ a stáhnout knihovny, na kterých je projekt závislý. Soubor je uložen v adresáři %M2_HOME%\conf (viz dokumentace k Maven2).

4 Projekt referenčního agenta

Pro vytváření implementace projektu, byl zvolený nástroj IDE IntelliJ IDEA. Kompilace projektu a jeho závislosti na externích knihovnách, je řízena nástrojem Maven2.

4.1 ReferentialAgent

Agent je implementovaný v projektu ReferentialAgent. Implementace obsahuje ukázku volání webových služeb ISZR E278, E256, E99 a E100. Pro účely demonstrace, je webová služby E278 (RobCtiPodleUdaju2) volána synchronně a E256 (RosCtiIco2) volán asynchronně a E99 a E100 synchronně. Asynchronní volání je implementováno aktivním čekáním. Druhou možností asynchronního volání je použití call-back funkce. Její implementace je v agentu pouze naznačena.

Implementace používá pro volání webových služeb technologii jax-ws. STUBy jsou generovány nástrojem wsimport.

4.2 Princip řešení

Pro představení principu volání služeb eGON rozhraní byla zvolena technologie jax-ws, která je standardním prostředkem pro volní webových služeb v prostředí Java. Tato technologie je integrální součástí prostředí Java. Díky této technologii pak implementace klienta obsahuje pouze business logiku. Tato technologie řeší následující oblasti:

- Komunikaci – navázání a ukončení spojení mezi klientem a serverem.
- Předání parametrů – vytvoření SOAP zprávy, její odeslání, přijetí, validaci a parsování.
- Předávání výjimek, které vzniknou během volání webové služby.
- Rozhraní – silně typové třídy a rozhraní, která na straně klienta reprezentují server. Jedná se o proxy. Zde se tato proxy nazývá STUB. Je generována nástrojem wsimport na základě wsdl, které popisuje danou webovou službu. Nástroj wsimport je součástí technologie jax-ws.

Obecné použití této technologie:

- Vygenerování tříd (STUBu), provolání webové služby na základě wsdl popisu dané webové služby.
- Použití vygenerovaných tříd ve vlastním projektu pro volání dané webové služby.

Výhody tohoto řešení:

- Implementátor se nezabývá komplexním technologickým popisem na základě wsdl a XSD.
- Implementátor se zaměřuje na sémantiku, nikoliv na syntaxi.
- Generované rozhraní lze snadno nahradit jeho novou verzí. Rozhraní se vygeneruje na základě nového wsdl.
- Generováním rozhraní snižujeme možnost vzniku chyby programátorem.

4.3 Struktura

Struktura projektu vychází ze specifikace nástroje Maven2:

- /src/main/java – Zdrojové soubory Java
- /src/jaxws – Konfigurace XML bindingu
- /target – Binární výstup build procesu (vznikne při kompilaci projektu)
- /generate – Vygenerované STUBy
- /ssl – Soubory pro šifrovanou komunikaci
- /cfg – Konfigurační soubory demo aplikace
- /wsdl – Soubory, které definují rozraní webových služeb

Volání webových služeb jsou implementovány jako potomci třídy cz.autocont.iszr.demo. BaseDemoCall. Implementace jsou uloženy v package cz.autocont.iszr.demo.eXX, kde XX je kód webové služby.

4.4 Kompilace

Podrobnosti kompilace projektu jsou uvedeny v komentáři v souboru pom.xml. Tento soubor je uložený v kořenovém adresáři projektu a definuje strukturu a závislosti projektu. Pro úspěšnou kompilaci je nutné:

- Nastavit maven tak, aby byl schopen stahovat soubory, na kterých je projekt závislý (soubor settings.xml)
- Nastavit proměnné prostředí:
  - Systémová proměnná MAVEN_OPTS.
  - wsdl.directory v souboru pom.xml - umístění WSDL souborů pro generování STUBu

Při kompilaci je možné aktivovat tyto profily:

- console-app – vznikne:
  - soubor /target /demo.jar, který je možné spustit z příkazové řádky ○ adresář /target/lib, který obsahuje knihovny, na nichž je demo.jar závislý.

Příklady kompilace projektu:

mvn clean package –P console-app

Pokud build proces proběhne správně, pak projekt obsahuje následující prvky:

- /generate – Vygenerované STUBy
- /target – Binární výstup build procesu
- /target/demo.jar – Spustitelná verze referenčního agenta

4.4.1 Generování STUBu

STUBy jsou generované pomocí wsimport. Tento nástroj se spouští jako plugin v Mavenu z pom.xml . STUBy se generují ve fázi „generate-sources“, když je aktivní profil „generate-stub“. Kvůli konfliktu pojmenování metod u metody E278 bylo nutné upravit ve vygenerovaných třídách metody v daných factory, proto po přegenerování jsou nutné dodatečné zásahy. Dodané vygenerované třídy jsou funkčí a lze se z nich inspirovat.

Příklad generování STUBu:

mvn generate-sources –P generate-stub mvn

clean generate-sources -P generate-stub

4.5 Konfigurace aplikace

Konfigurace aplikace je uložena v adresáři /cfg:

- /cfg/ISZR.properties – konfigurace dema, pro prostředí Egon

4.6 SSL

Pro komunikaci se službami ISZR se používá SSL, kde server i klient vlastní certifikát a jsou tímto certifikátem ověřeni. Certifikáty jsou uloženy v úložišti certifikátů. Pro vytvoření úložiště, je použitý nástroj keytool. Pro správnou SSL komunikaci klient potřebuje:

- Serverový certifikát. Tento certifikát musí být nainstalovaný jako důvěryhodný. Server, který je uvedený v certifikátu, se musí shodovat se serverem, na kterém je vystavena služba ISZR. Tento certifikát musí být nainstalovaný jak o důvěryhodný.
- Certifikát certifikační autority, který podepsala serverový certifikát. Tento certifikát musí být nainstalovaný v úložišti jako „Důvěryhodná CA“
- Klientský certifikát, kterému důvěřuje server ISZR. Tento certifikát musí obsahovat privátní klíč a úplnou cestu certifikačních autorit.

Všechny testovací certifikáty se nacházejí na stránkách zřizovatele ISZR.

Příklad vytvoření úložiště pro komunikaci přes SSL, je uvedený v /ssl/make.bat. Před spuštěním je potřeba nakonfigurovat cesty a heslo k certifikátům v souboru setup-egon-env.bat a v make.bat.

- pro správně fungování je nutné mít definovanou globální proměnou JAVA_HOME případně v /ssl/make.bat nastavit cestu k Java proměnná JAVA_HOME (příklad: set JAVA_HOME=C:\Program Files\Java\jdk-18.0.1.1)

Následně je nutné spustit *„make.bat egon“*
Tímto se vytvoří soubory keystore.egon.jks a truststroe.egon.jks.
Heslo a umístění úložiště poté uvést do konfigurace (např.: /cfg/ISZR.properties).

4.6.1 SSL a Java

ISZR používá pro SSL komunikaci variantu, kde je ověřený jak server, tak klient. V takovém případě je nutné vytvořit pro klienta dvě úložiště certifikátů:

- Trust store – toto úložiště obsahuje certifikáty, kterým klient věří. Defaultně se očekává certifikát kořenové certifikační autority „RefAgent-RootCA.cer“. Dále očekává certifikát certifikační autority, která podepsala certifikát serveru. ISZR používá dvou úrovňová hierarchii. Bude to tedy „RefAgent-SubCA.cer“. Vše se očekává ve složce /ssl/egon.
- Key store – toto úložiště bude obsahovat klientský certifikát včetně privátního klíče a celé cesty certifikačních autorit. Server musí tomuto certifikátu důvěřovat, kořenová certifikační autorita bude stejná (tzn.: RefAgent-RootCA).

Konfigurace SSL komunikace probíhá pomocí systémových parametrů Javy. Tato parametry je možné vložit, při spouštění dema, na příkazovou řádku. Nebo je vložit do konfiguračního souboru (viz.: /cfg/ISZR.properties). Tyto parametry jsou:

- -Djavax.net.ssl.trustStore – uložení souboru trust store
- -Djavax.net.ssl.trustStorePassword – heslo do trust store
- -Djavax.net.ssl.keyStore – uložení souboru key store
- -Djavax.net.ssl.keyStorePassword – heslo do key store

4.6.2 Vytvoření trust store

Trust store vytvoříme pomocí keytool v adresáři /ssl. Jako první vložíme certifikát RefAgent-RootCA. Certifikát označíme jako důvěryhodný.

keytool -alias iszr_root -importcert -file %ROOT_CER% -keystore %TRUSTSTORE% -storepass %PASSWD% -noprompt -trustcacerts

Pak vložíme certifikát, který podepisuje certifikáty serveru, tzn. RefAgent-SubCA.

keytool -alias iszr_sub_root -importcert -file %SUB_ROOT_CER% -keystore %TRUSTSTORE% storepass %PASSWD%

4.6.3 Vytvoření key store

Key store vytvoříme pomocí keytool v adresáři /ssl. K jeho vytvoření potřebujeme klientský certifikát s privátním klíčem a celou cestou certifikačních autorit. Je vhodné použít formát souboru pfx (pkcs12). Pokud tento soubor nemáme k dispozici, může jej vytvořit ze souboru typu „cer“ následujícím způsobem:

1. Do Internet exploreru nainstalujeme postupně certifikát RefAgent-RootCA, RefAgent-SubCA a klientský certifikát. U certifikátu RefAgent-RootCA je nutné ručně vybrat úložiště. Zvolíme úložiště důvěryhodných certifikačních autorit. Zbylé dva certifikáty nainstalujeme s defaultními volbami. U klientského certifikátu je nutné označit, že jeho privátní klíč je exportovatelný.
2. Provedeme export klientského certifikátu do souboru formátu pfx/pkcs12. Je nutné označit, že exportujeme privátní klíč a celou cestu certifikačních autorit. Internet explorer nás vyzve k zadání hesla pro přístup k privátnímu klíči. Java potřebuje, aby toto heslo bylo shodné s heslem pro přístup do úložiště. Pokud bude heslo odlišné, je možné jej později změnit.

Key store pak vytvoříte následujícím příkazem. Pro účely dema zvolíte heslo „aaaaaa“.

keytool -importkeystore -srckeystore %CLIENT_PFX% -srcstoretype pkcs12 -srcstorepass %CLIENT_PASS% -destkeystore %KEYSTORE% -deststoretype jks -deststorepass %PASSWD%

Pokud je heslo pro je heslo pro přístup k privátnímu klíči odlišné od hesla pro přístup do key store, je nutné změnit heslo pro přístup k privátnímu klíči. Heslo změníme takto:

1. Zjistíme, pod jakým aliasem byl privátní klíč uložený do key store *keytool -list -keystore %KEYSTORE% -storepass %PASSWD%*
2. Změníme heslo pro přístup k privátnímu klíči.

keytool -keypasswd -alias %CLIENT_ALIAS% -keypass %CLIENT_PASS% -new %PASSWD% keystore %KEYSTORE% -storepass %PASSWD%

4.7 /SSL/make.bat

Pro usnadnění práce při vytváření truststore a keystore, projekt referenčního agenta obsahuje skript /ssl/make.bat. Je nuné spustit pomocí .\make.bat egon. Skript předpokládá dvě úrovně certifikační autority. První parametr skriptu definuje prostředí, pro které chceme vytvořit keystore a truststore. Hodnoty daného prostředí jsou uloženy v souboru setup-egon-env.bat, kde egon je onen první parametr.

Příklad:

- Adresář /ssl obsahuje soubory make.bat, setup-egon-env.bat a podadresář /egon
- Keystore a truststore vytvoříte příkazem „make egon“

4.8 Spuštění referenčního agenta

Agenta se je možné spustit z příkazové řádky, nebo z vývojového prostředí. Pro spuštění agenta z příkazové řádky, je nutné provést kompilaci s aktivovaným profilem „console-app“. Pokud byla aplikace kompilovaná s aktivním profilem „console-app“, pak adresář target obsahuje soubor demo.jar. Data aplikace, která budou odeslána do webové služby, jsou uložena v souboru /cfg/ISZR.. Příklady spuštění aplikace příkazové řádky:

java -jar target\demo.jar -cfg="cfg\ISZR.properties"

Referenční agent vypíše na standardní výstup obsah truststore a keystore. Pak zavolá webové služby E278 a E256 a vypíše informace o tomto volání. Pak vyzve uživatele, aby zvolil:

- Volání služby E99
- Volání služby E100
- Ukončení aplikace

Aplikace referenčního agenta komunikuje s uživatelem přes standardní vstup a výstup. SOAP zprávy, které aplikace přijímá a odesílá, jsou uloženy do souborů, jejichž jména aplikace vypíše na standardní výstup (např.: /log/soap/E20_2012-03-09-125012_request.xml).

5 Externí odkazy

- IntelliJ IDEA- https://www.jetbrains.com/idea/
- Java - http://www.oracle.com/technetwork/java/index.html
- jax-ws - http://jax-ws.java.net/
- keytool - http://docs.oracle.com/javase/1.4.2/docs/tooldocs/windows/keytool.html
- Apache Maven2 - http://maven.apache.org/
- STUB
  - http://en.wikipedia.org/wiki/Java_API_for_XML-based_RPC

  - http://en.wikipedia.org/wiki/Remote_procedure_call#Sequence_of_events_during_a_RPC
  - http://en.wikipedia.org/wiki/Method_stub
- wsimport - http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html
- wsdl - http://www.w3.org/TR/wsdl