6 :: Vývoj

6:1 :: Vývoj služeb ESB

6:1:1 :: Jmenné konvence

WSO2 artifact

Proxy služby pass through, hub, pipe:

- ndic prefix + název bez I, bez camelCase – příklad:
  - IInputService = ndic_input_service
  - IInputCameraService = ndic_input_camera_service
- Aby správně fungoval deploy a undeploy, ponechat tento název i pro název souboru, v kterém je služba definována a artifactu v artifact.xml a pom.xml

WSDL

- ndic prefix + zdroj systém + název. Příklad: DIC.Contracts.Input.IInputService.wsdl = ndic_vars_input_service.wsdl

Služby jms interní

"ndic" prefix + typ transportu inbound endpointu + název + service. Příklad: ndic_jms_guaranteed_delivery_service

Endpointy

- ndic prefix + cíl+ název + endpoint. Příklad:
  - ndic_vars_input_service_endpoint (endpoint směřující na modul varsu)
  - ndic_int_input_service_endpoint (endpoint směřující na internu sběrnici)
  - ndic_queue_my_name_endpoint (endpoint na activemq queue)
  - ndic_topic_my_name_endpoint (endpoint na activemq topic)

Sekvence

- ndic prefix + název + sequence. Příklad:
  - ndic_message_prior_sequence
  - ndic_generate_wso2id_sequence

6:1:2 :: Tvorba mocků

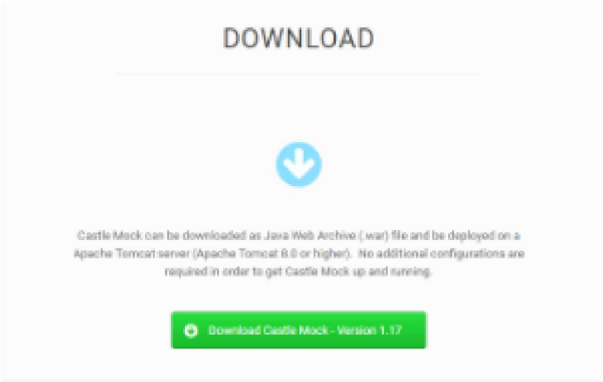
Pro tvorbu mocků je na projektu využit nástroj Castle Mock

Prerekvizity:

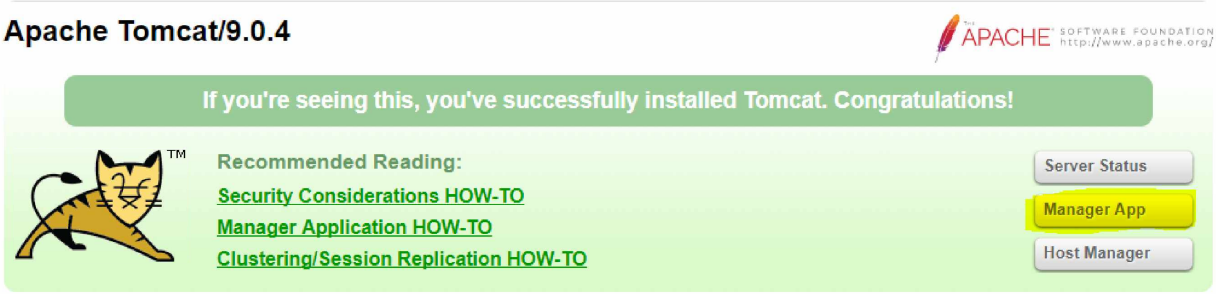
- Nainstalovaný Tomcat server.

Postup instalace Castle Mock

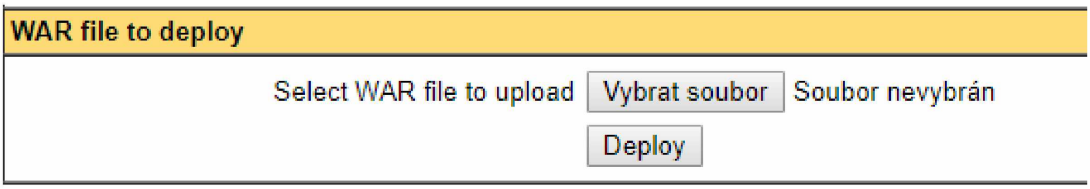
1. Stáhnout castlemock.war soubor ze stránky Castle Mock: https://castlemock.com/



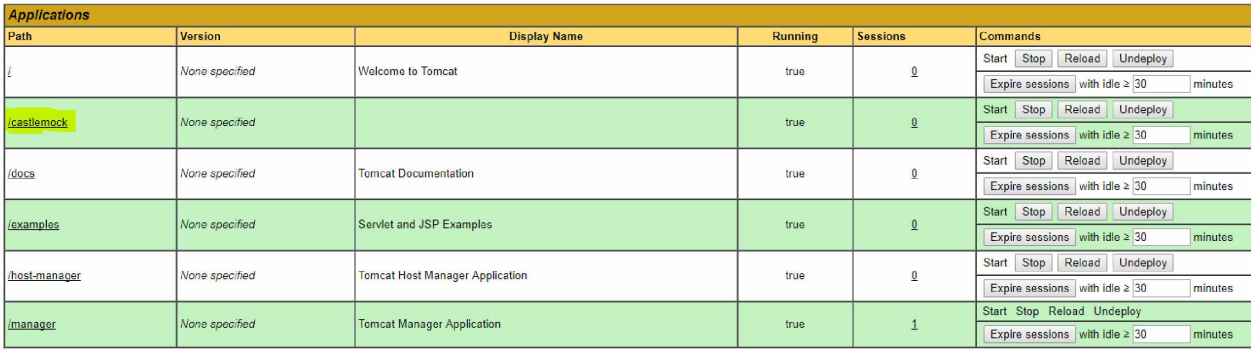
2. Otevřít Tomcat a přepnout do sekce Manager App



3. Nasadit stáhnutý soubor



4. Pokud vše proběhlo v pořádku, zobrazí se castlemock mezi běžícími aplikacemi v aplikačním seznamu Tomcatu viz. localhost:8080/manager/html



5. Pokud se připojíte na localhost:8080/castlemock, je instalace hotová.

Alternativní postup instalace

6. Stáhnout castlemock.war soubor ze stránek Castle Mock
7. Do adresáře <tomcat_home>\webapps nakopírovat soubor castlemock.war
8. Pokud se soubor správně (sám) rozbalí a lze se připojit na localhost:8080/castlemock, je instalace hotová

V případě nefunkčnosti instalace

1. Odstranit složku castlemock a soubor castlemock.war z adresáře <tomcat_home>\webapps
2. Pro jistotu znova stáhnout soubor castlemock.war z oficiálních stránek a zkusit alternativní postup instalace.

Postup vytvořeni SOAP projektu v Castle Mocku

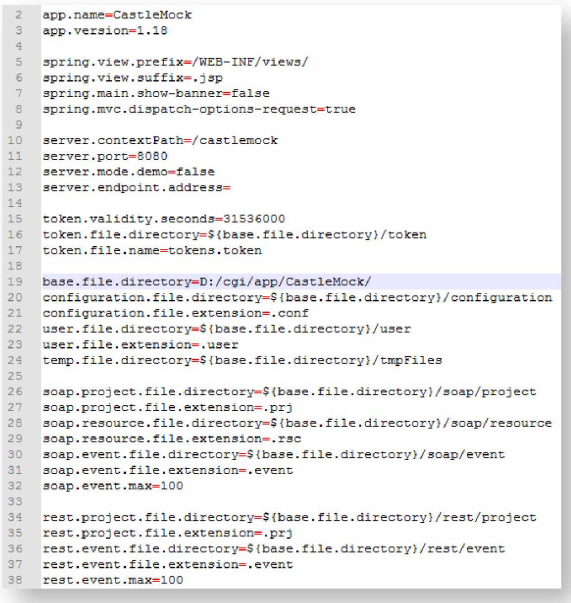
Je popsán na stránkách Castle Mock v záložce use case soap.

Defaultní cesta k SOAP adresářům v Castle Mocku

- Cílový adresář s projekty je v: <User-account-name>\.castlemock\soap\project
- Adresář s logy je v: <User-account-name>\.castlemock\soap\event
- Adresář s přidáním WSDL se nachází zde: <User-account-name>\.castlemock\soap\resource

Jak změnit defaultní cestu ke Castle Mocku (application.properties)

1. Defaultně se složka nachází v adresáři: <User-account-name>/.castlemock
2. V soboru application.properties je možné tuto adresu změnit, tento soubor najdeme v adresáři: <tomcat_home>\webapps\castlemock\WEB-INF\classes
3. Náhled na upravený application.properties z DEV-INT-LB.



```
app.name=CastleMock
app.version=1.18

spring.view.prefix=/WEB-INF/views/
spring.view.suffix=.jsp
spring.main.show-banner=false
spring.mvc.dispatch-options-request=true

server.contextPath=/castlemock
server.port=8080
server.mode.demo=false
server.endpoint.address=

token.validity.seconds=31536000
token.file.directory=${base.file.directory}/token
token.file.name=tokens.token

base.file.directory=D:/cgi/app/CastleMock/
configuration.file.directory=${base.file.directory}/configuration
configuration.file.extension=.conf
user.file.directory=${base.file.directory}/user
user.file.extension=.user
temp.file.directory=${base.file.directory}/tmpFiles

soap.project.file.directory=${base.file.directory}/soap/project
soap.project.file.extension=.prj
soap.resource.file.directory=${base.file.directory}/soap/resource
soap.resource.file.extension=.rsc
soap.event.file.directory=${base.file.directory}/soap/event
soap.event.file.extension=.event
soap.event.max=100

rest.project.file.directory=${base.file.directory}/rest/project
rest.project.file.extension=.prj
rest.event.file.directory=${base.file.directory}/rest/event
rest.event.file.extension=.event
rest.event.max=100
```

Jak změnit název projektu

Každý nově vytvořený projekt dostává automaticky vygenerovaný název tzv. id. Tento je možné manuálně změnit:

1. Je potřebné vypnut server Tomcat na kterém běží Castle Mock.
2. Změnit název souboru s projektem - soubor s projektem najdeme v adresáři: <User-account-name>\.castlemock\soap\project
3. Dále pomocí textového editoru je potřebné změnit název i v samotném souboru projektu.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <soapProject>
3      <created>2018-01-31T08:30:33.021+01:00</created>
4      <description></description>
5      <id>stary-nazov</id> <!-- Nazov suboru -->
6      <name>Testovaci projekt</name> <!-- Nazov projektu -->
7      <updated>2018-01-31T08:30:33.021+01:00</updated>
```

4. Soubor uložíme a zapneme server Tomcat. Hotovo.

Kde psát response

V adresáři s projektem je potřebné otevřít soubor soubor .prj s projektem a změnit tělo odpovědi - viz. obrázek:

```
<operation>
    <currentResponseSequenceIndex>0</currentResponseSequenceIndex>
    <defaultBody>&lt;soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://schemas.xmlsoap.org/wsdl/"&gt;
                 &lt;soapenv:Header/&gt;
                 &lt;soapenv:Body&gt;
                 &lt;web:GetAllSettingsResult&gt;?&lt;/web:GetAllSettingsResult&gt;
                 &lt;/soapenv:Body&gt;
                 &lt;/soapenv:Envelope&gt;
    </defaultBody>
    <forwardedEndpoint>http://ndic-app1.rsdlan.local/ndic-app1.rsdlan.local/DIC.Contracts.Settings.ISettingsService.svc</forwardedEndpoint>
    <httpMethod>POST</httpMethod>
    <id>adresa-operacie</id>
    <identifier>GetAllSettings</identifier>
    <mockResponses>
        <mockResponse>
            <body>Priestor pre response</body> <!-- telo odpovede -->
            <httpHeaders/>
            <httpStatusCode>200</httpStatusCode>
            <id>adresa-odpovede-operacie</id> <!-- nazov adresy odpovede -->
            <name>AllSettingsResult mocked response</name> <!-- nazov odpovede -->
            <status>ENABLED</status>
            <usingExpressions>true</usingExpressions>
        </mockResponse>
    </mockResponses>
    <name>GetAllSettings</name>
    <networkDelay>0</networkDelay>
    <originalEndpoint>http://ndic-app1.rsdlan.local/ndic-app1.rsdlan.local/DIC.Contracts.Settings.ISettingsService.svc</originalEndpoint>
    <responseStrategy>RANDOM</responseStrategy>
    <simulateNetworkDelay>false</simulateNetworkDelay>
    <soapVersion>SOAP11</soapVersion>
    <status>MOCKED</status>
</operation>
```

Struktura response v šabloně:

<operations>

<operation>

...

...

...

</operation>

<operation>

...

...

...

</operation>

</operations>

Poznámky:

Po uploadu nového WSLD se staré z *resource* smaže, ale už přidané činnosti zůstanou nezměněné. Nově přidané WSDL je možné najít v adresáři „resource“.

6:1:3 :: Generování proxy služeb

Důležité soubory

CONF\soap_wsdls_resources_list

- ei_templates - soubory potřebné pro generátor
  - ei_artifact_template.xml
  - ei_endpoint_template.xml
  - ei_proxy_service_template.xml
- gr_templates - sobory potřebné pro generátor
  - gr_artifact_template.xml
- generator.properties
- wsdls.csv - sobor potřebný pro generátor
- wsdl_comparator.bat
- wsdl_properties_file_transformer.bat

Jak funguje generování

Generování proxy služeb má více částí. Jde o stahování WSDL pak na jejich základě se generují endpointy a proxy služby. Vstupem jsou i výše zmíněné soubory.

Důležitým prvkem je wso2_proxies_generator. Ten generuje synapse artefakty, proxy.xml a následně aktualizuje pom soubory. Projekty, které obsahují generované soubory, obsahují sufix „generated“.

Celý tento proces je automatizovaný a vykonávaný jenkinsem WSO-wsdl-update-pipeline. Samotný build a deploy nejsou součástí tohoto jobu.

Proxy generátor

Je implementace v Javě wso2_proxies_generator, která jednak generuje endpointy, proxy, artifact.xml, ale také aktualizuje pom soubory car artefaktů. Vstupními konfigurace do této služby jsou generator.properties

Názvy služeb a proxy servis jsou totožné s tím, co je jako namespace v csv souboru. Generátor property soubor obsahuje všechny potřebné nastavení potřebné ke generování. Hlavně k tomu,

co má generovat a hlavně kde. V property souboru jsou všechny potřebné komentáře k tomu aby generátor fungoval (project.root.path nesmí byt nikdy commitnuta).

cmd execution:

java -jar wso2_proxies_generator.jar <wsdls.csv> <generator.properties> generate

Soubory <wsdls.csv> a <generator.properties> musí být uvedeny včetně absolutní cesty.

Vygenerované soubory (ve wso2/ESB) by se neměli pushovat na git. V jenkinsu je vhodnější spustit task WSO-wsdl-update-pipeline. Pokud nastane chyba „[Wsdl-changes-publisher] the 'build only if scm changes' feature is disabled.“, je třeba, aby task spustil někdo, kdo na to má dostatečná práva. V případě, že byly provedeny změny v generátoru je třeba nejprve pushnout a spustit jenkins job CGI-SUPPORT-PROJECT-BUILD, přitom select boxu vybrat wso2-proxies-generator

6:1:4 :: Přidání endpointu

Průvodce - Krok za krokem analogicky pro INT i EXT sběrnici:

1. XML soubor endpointu (předpokládejme ndic_queue_my_name_endpoint.xml) přidat do wso2\ESB\int\wso2_integration\ndic_integration\src\main\synapse-config\endpoints. Jméno souboru musí být shodné se jménem endpointu v soboru definovaném (tzn. název endpointu zde bude ndic_queue_my_name_endpoint).

   SOAP verze – Moduly NDIC aktuálně akceptují SOAP zprávy pouze verze 1.1. Proto je při vytváření address endpointu třeba definovat formát v elementu address: <address format="soap11".../>

2. Přidat záznam do wso2\ESB\int\wso2_integration\ndic_integration\artifact.xml

   <artifact name="ndic_queue_my_name_endpoint" groupId="com.cgi.ndic.int.wso2.int.endpoint" version="0.0.3-SNAPSHOT" type="synapse/endpoint" serverRole="EnterpriseServiceBus">

   <file>src/main/synapse-config/endpoints/ndic_queue_my_name_endpoint.xml</file>

   </artifact>

3. V souboru wso2\ESB\int\wso2_integration\ndic_integration_CompositeApplication\pom.xml přidat project/properties/property

   <com.cgi.ndic.int.wso2.int.endpoint_._ndic_queue_my_name_endpoint>capp/EnterpriseServiceBus</com.cgi.ndic.int.wso2.int.endpoint_._ndic_queue_my_name_endpoint>

4. Ve stejném souboru přidat dependencies/dependency

   <dependency>

   <groupId>com.cgi.ndic.int.wso2.int.endpoint</groupId>

   <artifactId>ndic_queue_my_name_endpoint</artifactId>

   <version>0.0.3-SNAPSHOT</version>

   <type>xml</type>

</dependency>

6:1:5 :: Přidání proxy služby

1. XML soubor proxy služby (předpokládejme ndic_my_name.xml) přidat do wso2\ESB\int\wso2_integration\ndic_integration\src\main\synapse-config\proxy-services. Jméno souboru musí být shodné se jménem proxy služby v soboru definované (tzn. název proxy služby zde bude ndic_my_name).
2. Přidat záznam do wso2\ESB\int\wso2_integration\ndic_integration\artifact-template.xml

   <artifact name="ndic_server_input" groupId="com.cgi.ndic.int.wso2.int.proxy-service" version="0.0.3-SNAPSHOT" type="synapse/proxy-service" serverRole="EnterpriseServiceBus">

   <file>src/main/synapse-config/proxy-services/ndic_server_input.xml</file>

   </artifact>
3. V souboru wso2\ESB\int\wso2_integration\ndic_integration_CompositeApplication\pom.xml přidat v project/properties/property

   <com.cgi.ndic.int.wso2.int.endpoint_._ndic_my_name>capp/EnterpriseServiceBus</com.cgi.ndic.int.wso2.int.endpoint_._ndic_my_name>
4. Ve stejném souboru přidat dependencies/dependency

   <dependency>

   <groupId>com.cgi.ndic.int.wso2.int.proxy-service</groupId>

   <artifactId>ndic_server_input</artifactId>

   <version>0.0.3-SNAPSHOT</version>

   <type>xml</type>

   </dependency>

6:1:6 :: WSDL do GR

1. Nakopírovat WSDL soubor do wso2\ESB\common\wso2_governance\ndic_governance_Registry
2. V souboru wso2\ESB\common\wso2_governance\artifact.xml přidat element artifact
3. V souboru wso2\ESB\common\wso2_governance\ndic_governance_CompositeApplication\pom.xml přidat element do elementu project/properties, jehož název bude groupId_._name, kde groupId a name jsou hodnoty atributů s tímto jménem v elementu artifact, přidaného v předchozím kroku.

   Příklad:

   Pokud byl v předchozím kroku přidán atribut <artifact name="ServerInput_wsdl" groupId="com.cgi.ndic.wso2.gr.resource"...>, pak v tomto kroku bude přidán element <com.cgi.ndic.common.wso2.gr.resource_._ServerInput_wsdl>capp/GovernanceRegistry</com.cgi.ndic.common.wso2.gr.resource_._ServerInput_wsdl>

4. Do stejného souboru přidat do project/dependencies element dependency, jehož groupId a version budou stejné jako v bodu 2 a element artifactId bude odpovídat atributu name z bodu 2

6:2 :: Vývoj monitoringu

6:2:1 :: Stream processor

Sidhi aplikace

Siddhi aplikace slouží pro přijímaní, zpracovávání a další práce s eventy. Siddhi aplikace lze psát v editoru, který je dostupný jako režim Stream Processoru. Níže je příklad jednoduché siddhi aplikace pro zpracování eventů se statistikami z EI.

```
@App:name("FlowEntryApp")
@App:description("Plan of flow entry")
@source(type='wso2event', wso2.stream.id='org.wso2.esb.analytics.stream.FlowEntry:1.0.0', @map(type = 'wso2event'))
define stream FlowEntry(meta_compressed bool, meta_tenantId int, messageId string, flowData string);
@sink(type='log', prefix='My flowEntry:')
define stream TestOutputFlowEntry(messageId string, flowData string);
@info(name='FlowEntryOutput')
from FlowEntry(meta_compressed, meta_tenantId, flowData, messageId)
select messageId, flowData
insert into TestOutputFlowEntry;
```

Základní pojmy:

- **@App**
  - name - název aplikace
  - description - popis aplikace
- **@source**
  - type - typ zdroje, ze kterého zpráva přijde, každý typ má specifický protokol pro transport, lze použít mnoho druhů, např. http, wso2event, email apod. Více → https://docs.wso2.com/display/SP400/Collecting+Events
  - wso2.stream.id - slouží pro přepsání stream id příchozího eventu. Siddhi query při definování streamu nepodporuje v názvu speciální znaky (jako '.'). Při posílání statistik z EI je použit název streamu org.wso2.esb.analytics.stream.FlowEntry:1.0.0, takový název ale nelze při definování použít, jelikož jej Siddhi nepodporuje, použije se proto tato property, která přepíše název streamu a použije níže definovaný název FlowEntry. Vyhneme se tak erroru: "No StreamDefinition for streamId org.wso2.esb.analytics.stream.FlowEntry:1.0.0 present in cache"
- **@map**

  - type - mapování datových typů příchozích eventů. Pokud posíláme event s datovým typem wso2event, siddhi si typ namapuje a ví, jak s ním dále pracovat
- **define stream**
  - definujeme název input streamu, do kterého se příchozí data uloží. V závorce definujeme všechny data a jejich datové typy, které chceme ze zprávy zpracovat. Pro správné definování dat a jejich typů je třeba vědět formát streamu, který EI posílá. Streamy posílané z EI jsou uložené v json formátu, konkrétně v EI home/reposistory/deployment/server/eventstreams
  - prefix meta udává, že se jedná o atribut přenášený jako metadata
- **@sink**
  - sink slouží pro další publikování streamů přes různé transportní protokoly. Streamy lze publikovat například do emailu, přes http protokol, přes tcp protokol do dalšího Analytic serveru nebo jen do server logu, jako v našem příkladu
  - type - typ "protokolu", pomocí kterého se dále stream publikuje, log znamená vypsání do server logu
  - prefix - zobrazovaný údaj před každým zpracovaným streamem
- **define stream**
  - definujeme název output streamu, do kterého následně vložíme zpracované data a ten se pomocí sink publikuje dále. V závorce definujeme atributy a jejich datové typy, které chceme streamem publikovat
- **from**
  - Siddhi query pro zpracování streamu musí mít definovaný název input streamu, ze kterého bude brát data, i s atributy ale už bez datových typů
- **select**
  - Query pro vybrání určitých (nebo klidně všech) atributů z input streamu
- **insert into**
  - Query pro vložení dat do output streamu

Napojení EI na SP

Prerekvizity:

Nainstalovaný EI a SP

Pozor: Při změně certifikátu je třeba přegenerovat heslo v *publisherech* a *EventSink*:

1. V EI home/repository/deployment/server/eventpublishers složce nakonfigurovat dva soubory MessageFlowConfigurationPublisher.xml a MessageFlowStatisticsPublisher.xml
   a. u *password* atributu nastavit "encrypted" na false a změnit kryptované heslo na plain text původní heslo (admin)
2. Plain heslo se přegeneruje po reloadu souboru. Kryptované heslo zkopírovat a vložit do EventSink souboru (EI home/repository/deployment/server/event-sinks) místo hesla, které tam je

Konfigurace TrustStore a DataBridge:

1. Ze SP složky -> ${carbon_wso2sp_home}\resources\security zkopírovat soubor - client-truststore.jks
2. V EI → ${carbon_wso2ei_home}\repository\resources\security - vytvořit složku wso2sp_store
3. Zkopírovaný soubor client-truststore.jks vložit do této složky
4. V EI ve složce \conf\data-bridge nakonfigurovat data-agent-config.xml

V sekci

<Agent>

<Name>Thrift</Name>

Odkomentovat <TrustSore> a <TrustSorePassword> a upravit takto:

<TrustSore>D:\cgi\app\wso2ei-6.1.1\repository\resources\security\wso2sp_store\client-truststore.jks</TrustSore>

<TrustSorePassword>wso2carbon</TrustSorePassword>

Vytvoření EventSink:

1. V management consoli EI v sekci Configure→Event Sinks dát Add Event Sink

   Name: Jakékoliv identifikační jméno (např. wso2eventSink)

   Username: admin

   Password: admin

   Receiver URL: tcp://<SP_URL>:<Thrift TCP Port> - Například tcp://192.168.251.23:7611

   Authenticator URL: ssl://<SP_URL>:<Thrift SSL Port> - Například ssl://192.168.251.23:7711

   Po vytvoření Event Sink zapnout SP v režimu worker, po nastartování SP restartovat EI. V logu SP by se měla objevit hláška - INFO {org.wso2.carbon.databridge.core.DataBridge} - user admin connected

Poznámka: Přes UI v mngmt consoli nelze do URL zadávat DNS překlad místo IP adresy.

Vytvoření PublishEvent:

1. Pro odesílání wso2eventů z integrátoru do stream processoru je třeba do každé služby přidat Publish Event Mediator
2. Do bloku <InSequence> přidat <publishEvent> a nastavit následovně:

   <publishEvent>

   <eventSink>wso2eventSink</eventSink> (název EventSink, který jsme vytvořili v předchozím kroku)

   <streamName>TestProductionStream</streamName> (název Streamu, který použijeme v Siddhi aplikaci na Stream Processoru pro přijímání eventů - pro každou skupinu eventů separátní stream)

   <streamVersion>1.0.0</streamVersion>

   <attributes> (různé atributy - nastavení lišící se od toho, jaké informace chceme v eventu přenášet)

   <meta/>

   <correlation/>

```
        <payload/>
        <arbitrary/>
        </attributes>
        </publishEvent>
```

Definovaný PublishEvent:

```
<publishEvent>
   <eventSink>NDIC_SP</eventSink>
   <streamName>NDIC_EVENT_STREAM</streamName>
   <streamVersion>1.0.0</streamVersion>
   <attributes>
      <meta />
      <correlation />
      <payload>
         <attribute name="serviceName" type="STRING" defaultValue="" expression="$ctx:proxy.name" />
         <attribute name="sequenceName" type="STRING" defaultValue="" expression="get-property('sequenceName')" />
         <attribute name="activity" type="STRING" defaultValue="" expression="get-property('activity')" />
         <attribute name="processType" type="STRING" defaultValue="" expression="get-property('processType')" />
         <attribute name="status" type="STRING" defaultValue="200" expression="get-property('status')" />
         <attribute name="time" type="LONG" defaultValue="" expression="get-property('SYSTEM_TIME')" />
         <attribute name="originator" type="STRING" defaultValue="" expression="get-property('originator')" />
         <attribute name="wso2Id" type="STRING" defaultValue="" expression="get-property('wso2Id')" />
         <attribute name="messageUuid" type="STRING" defaultValue="" expression="get-property('MessageID')" />
         <attribute name="payload" type="STRING" defaultValue="" expression="get-property('payload')" />
         <attribute name="headers" type="STRING" defaultValue="" expression="get-property('headers')" />
         <attribute name="details" type="STRING" defaultValue="" expression="get-property('details')" />
```