Secure Token Service

Table of Contents

Overview

Secure Token Service (STS) is a web-service entity responsible for issuing, validation, renewing and cancelling security tokens as described by the WS-Trust 1.3 specification. Details on WS-Trust 1.3 specification can be found in [1]. STS exposes web-service endpoints for requesting / validating Security Tokens (ST) issued by it. In its current implementation we support STS backed by Dorian or LDAP Identity Store. The ST generated by the STS is a standard SAML2 Token that also complies with IHE's XUA specification. With simple configuration STS can be used in applications for Authentication, Identity Assertion and Management.

Features

- WS-Trust 1.3 compliant (Token renewing and cancellation not supported currently)
- Supported backend include:
    - caGRID 's Dorian
    - LDAP
    - Simple Property file containing username/password (not recommended for Production)
- SOAP endpoints for requesting/validating ST.
- RESTful endpoints for requesting/validating ST.
- Java API to interact with STS over HTTP
- *IHE's XUA Profile compliance is being tested*

System Requirements

- JDK 1.6
- Apache Maven
- JBoss Application Server 6.0.0
- Eclipse IDE *(optional)*

Documentation

Programmers Guide:

The API for interfacing with the STS is very straightforward. There are two allowable operations:

- issueToken : To request a security token that can be presented by the requester to the service provider embedding it in the request.
- validateToken : To request the STS to validate a security token. Validation will be performed by the service provider.

Both requests must be accompanied with the credentials (username/password) of the requester.

RESTful API

(Assume STS is hosted on https://localhost:8443/SecurityTokenServicePF)

IssueToken

GET `https://localhost:8443/SecurityTokenServicePF/rest/STS/issueToken?targetService=http%3A%2F%2Fservices.testcorp.org%2Fprovider1`

Returns: A SAML2 assertion

ValidateToken

POST `https://localhost:8443/SecurityTokenServicePF/rest/STS/validateToken`

Form Parameters: `token=<saml2 assertion to be validated>`

```
Returns:  'http://docs.oasis-open.org/ws-sx/ws-trust/200512/status/valid' when valid token presented
          'http://docs.oasis-open.org/ws-sx/ws-trust/200512/status/invalid' when invalid token
          presented
          '<error>error message</error>' when exception thrown by server
```

Please bear in mind , the STS RESTful API uses HTTP Basic Authentication scheme , thus each request must provide username/password when making calls to the STS. Example on how to do that can be found in jUnits provided with the distribution. Also, the communication is SSL encrypted so you will have to store the server's certificate in the trust store of the RESTful client .

We have provided jUnit test cases that show you how to issue/validate security tokens. These jUnits are located in

`<STS Download Folder>/src/test/java`

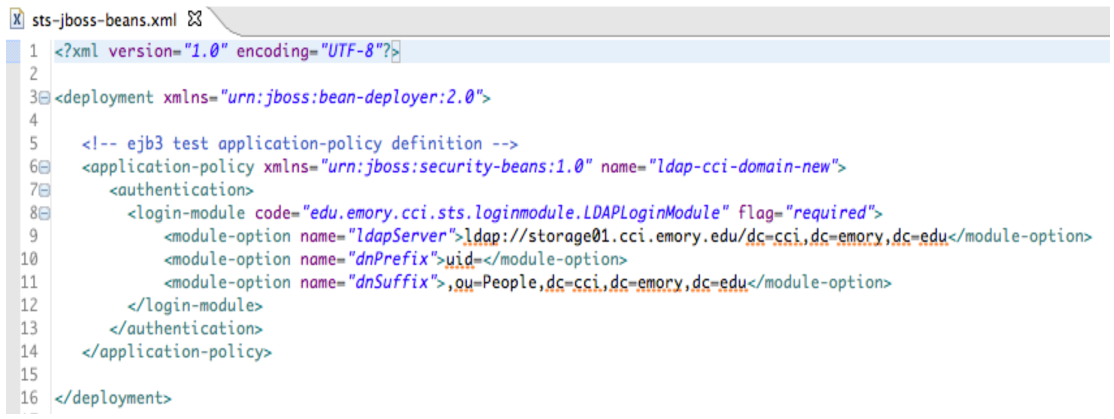
Administrator/Deployment Guide:

STS can be deployed with Maven either using Eclipse IDE equipped with Maven Plugin or through command-line option. **Some familiarity with Maven is assumed.**

Step 1: Check out the source code from the repository
Step 2: Configure the backend for deployment:

Configuration for use with LDAP

- Edit `SecureTokenService/ldap-config/WEB-INF/sts-jboss-beans.xml`



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

<deployment xmlns="urn:jboss:bean-deployer:2.0">

   <!-- ejb3 test application-policy definition -->
   <application-policy xmlns="urn:jboss:security-beans:1.0" name="ldap-cci-domain-new">
      <authentication>
        <login-module code="edu.emory.cci.sts.loginmodule.LDAPLoginModule" flag="required">
            <module-option name="ldapServer">ldap://storage01.cci.emory.edu/dc=cci,dc=emory,dc=edu</module-option>
            <module-option name="dnPrefix">uid=</module-option>
            <module-option name="dnSuffix">,ou=People,dc=cci,dc=emory,dc=edu</module-option>
        </login-module>
      </authentication>
   </application-policy>

</deployment>
```

- Module Option:
  ldapServer: Url of the LDAP server
  dnPrefix: Prefix of DN associated with the LDAP account
  dnSuffix: Suffix of the DN associated with the LDAP account.

Configuration for use with Dorian

- Edit `SecureTokenService/ldap-config/WEB-INF/sts-jboss-beans.xml`
- Module Option:
  identityProviderUrl: Url of the Dorian service to be used for authentication.
  Edit `SecureTokenService/dorian-config/WEB-INF/classes/picketlink-sts.xml:`
  Change the highlighted property to the GridGrouper Service location

Step 3: Run Maven Goal "package" under following profile (See Maven command help in following section) :

- "profile-dorian" for STS with Dorian Backend
- "profile-ldap" for STS with LDAP backend

```
<PicketLinkSTS xmlns="urn:picketlink:identity-federation:config:1.0"
          STSName="EmoryConceptSTS" TokenTimeout="72000" EncryptToken="false">
          <KeyProvider ClassName="org.picketlink.identity.federation.core.impl.KeyStoreKeyManager">
                    <Auth Key="KeyStoreURL" Value="sts_keystore.jks"/>
                    <Auth Key="KeyStorePass" Value="testpass"/>
                    <Auth Key="SigningKeyAlias" Value="sts"/>
                    <Auth Key="SigningKeyPass" Value="keypass"/>
                    <ValidatingAlias Key="http://services.testcorp.org/provider1" Value="service1"/>
          </KeyProvider>
          <TokenProviders>
        <TokenProvider ProviderClass="edu.emory.cci.sts.tokenprovider.XUACompliantSAML2TokenProvider"
            TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0"
                  TokenElement="Assertion"
                  TokenElementNS="urn:oasis:names:tc:SAML:2.0:assertion">
                  <Property Key="AttributeProvider"
Value="edu.emory.cci.sts.attributeprovider.GridGrouperSAMLAttributeProvider"/>
                  <Property Key="gridGrouperUrl" Value="https://cagrid-
dev.cci.emory.edu:7443/wsrf/services/cagrid/GridGrouper"/>
                  <Property Key="ifsUrl" Value="https://cagrid-dev.cci.emory.edu:8443/wsrf/services/cagrid/Dorian"/>
                  </TokenProvider>
          </TokenProviders>
          <ServiceProviders>
                    <ServiceProvider Endpoint="http://services.testcorp.org/provider1" TokenType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0"
                              TruststoreAlias="service1"/>
          </ServiceProviders>
</PicketLinkSTS>
```

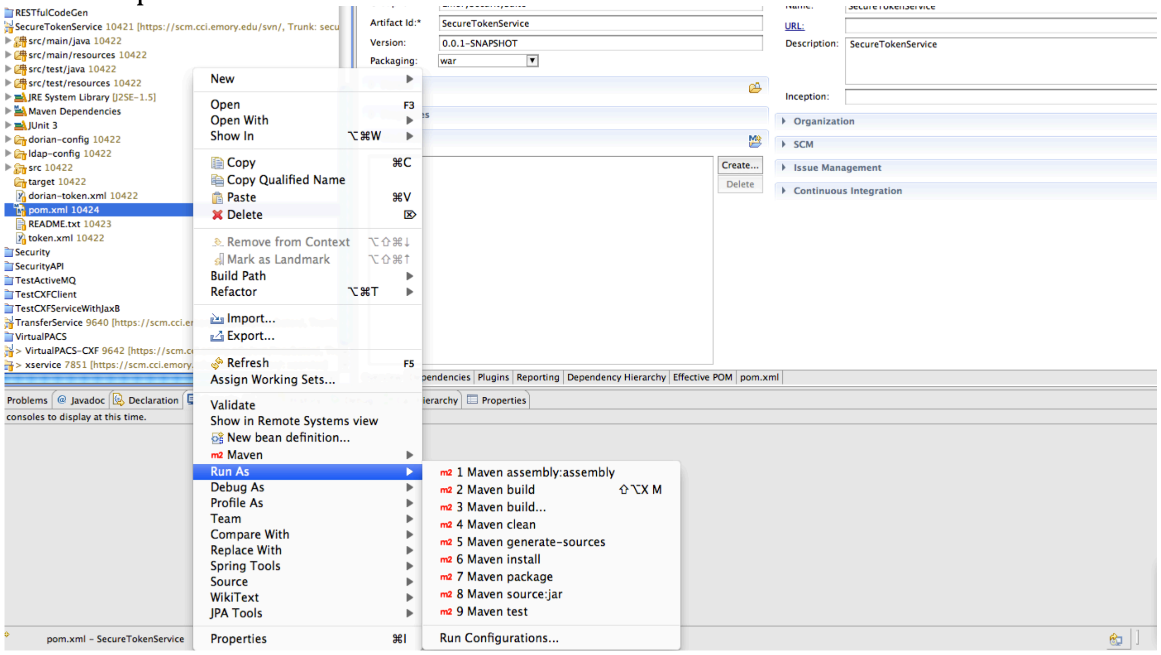
Step 4: Step 3 should generate a WAR file SecurityTokenService{ProfileName}.war in the SecureTokenService/target directory. Deploy this file on JBoss AS 6.0.0

Using Maven

From Command Line (From project's home directory)

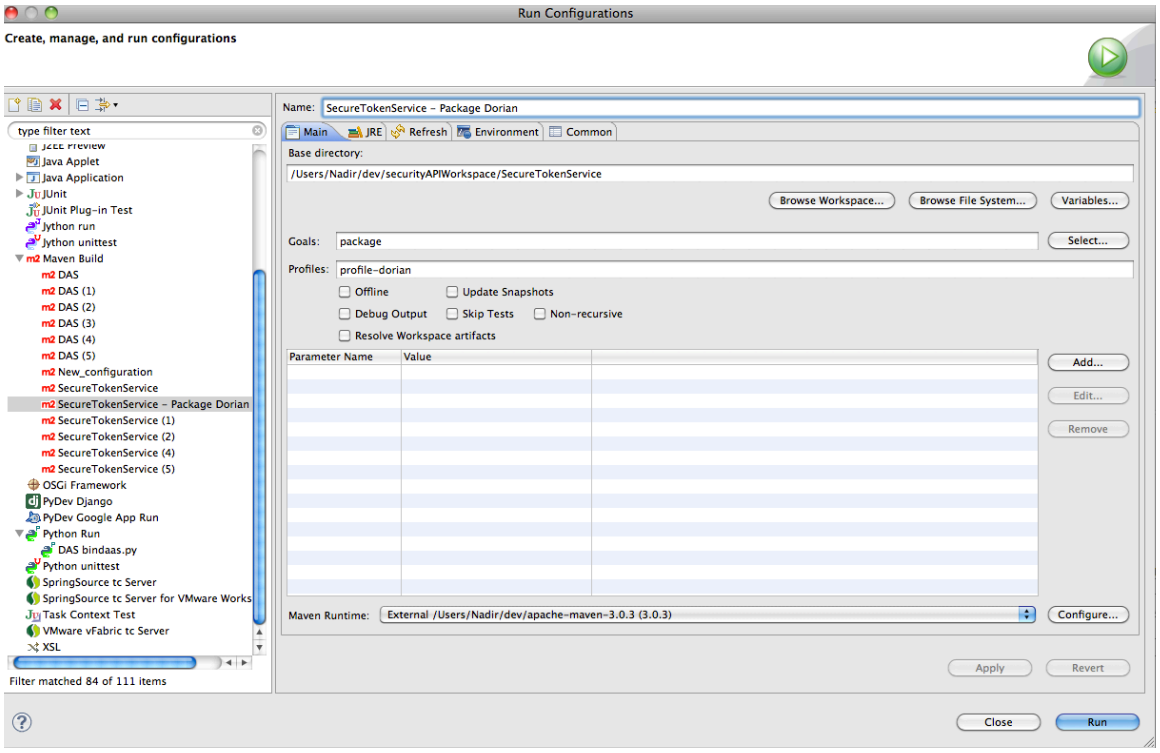
```
mvn package -Dbackend=ldap
```

From Eclipse



Artifact Id:* SecureTokenService
Version: 0.0.1-SNAPSHOT
Packaging: war
URL:
Description: SecureTokenService
Inception:

Choose Maven Build option



Run Configurations

Create, manage, and run configurations

type filter text

- J2EE Preview
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Jython run
- Jython unittest
- Maven Build
  - DAS
  - DAS (1)
  - DAS (2)
  - DAS (3)
  - DAS (4)
  - DAS (5)
  - New_configuration
  - SecureTokenService
  - SecureTokenService - Package Dorian
  - SecureTokenService (1)
  - SecureTokenService (2)
  - SecureTokenService (4)
  - SecureTokenService (5)
- OSGi Framework
- PyDev Django
- PyDev Google App Run
- Python Run
  - DAS bindaas.py
- Python unittest
- SpringSource tc Server
- SpringSource tc Server for VMware Works
- Task Context Test
- VMware vFabric tc Server
- XSL

Filter matched 84 of 111 items

Name: SecureTokenService - Package Dorian

Main | JRE | Refresh | Environment | Common

Base directory:
/Users/Nadir/dev/securityAPIWorkspace/SecureTokenService

Browse Workspace... | Browse File System... | Variables...

Goals: package

Profiles: profile-dorian

- [ ] Offline
- [ ] Update Snapshots
- [ ] Debug Output
- [ ] Skip Tests
- [ ] Non-recursive
- [ ] Resolve Workspace artifacts

Parameter Name	Value

Add... | Edit... | Remove

Maven Runtime: External /Users/Nadir/dev/apache-maven-3.0.3 (3.0.3)

Configure...

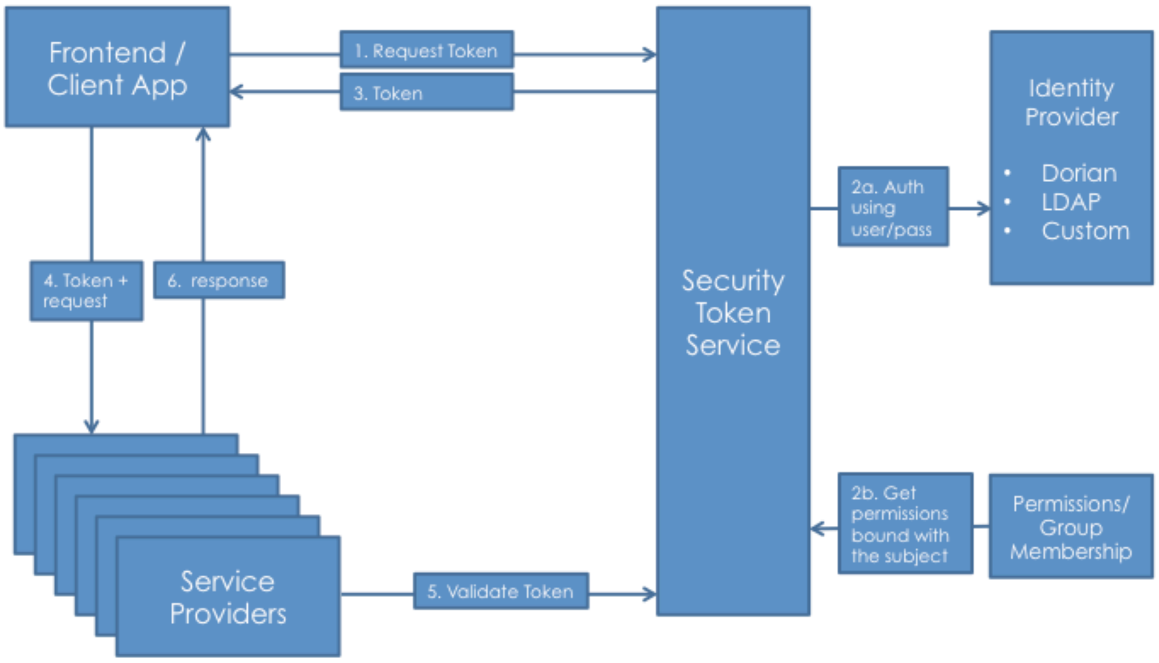
Apply | Revert

Close | Run

Developer Guide:

Our STS implementation is developed on top of Jboss's PicketLink project.
For more information about the architecture and how to extend please visit [5].

The basic architecture of STS is shown below:



In the example shown above, a client (frontend / client app) wants to request data from one or more service providers in a secure manner. The sequence of events for this secure transaction is as follows:

1. Client asks the Secure Token Service (STS) for a token with the issueToken request (for examples on how to do this, see developer API)
2. STS uses the enclosed username/password to communicate with an identity provider (Dorian, LDAP etc) and obtain credentials.
3. STS uses these credentials to obtain any group memberships.
4. The credentials and any group memberships are packaged as a SAML2 assertion and the token is sent to the client.
5. The client attaches this token with it's request to one or more service providers.
6. The service providers can ask the STS to validate the token (for examples on how to do this, see developer API).
7. If the token is found to be valid, the service provider can proceed with responding to the client. Any authorization, including instance level authorization is the responsibility of the service and those authorization steps will be executed after the token has been validated.

Resources

Repository Location

Source Code location:
https://scm.cci.emory.edu/svn/ivi/trunk/security/SecureTokenService

Other Resources

1. WS-Trust 1.3 Specification document: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html
2. Maven Download: http://www.apache.org/dyn/closer.cgi/maven/binaries/apache-maven-3.0.3-bin.tar.gz
3. JBoss AS 6.0.0 (Note: We have not tested on any other version of Jboss, hence use 6.0.0 version): http://sourceforge.net/projects/jboss/files/JBoss/JBoss-6.0.0.Final
4. Eclipse IDE : http://www.eclipse.org/downloads/packages/eclipse-classic-371/indigosr1
5. JBoss Picketlink: http://www.jboss.org/picketlink