jUDDI 3.0

User Guide

ASF-JUDDI-USRGUIDE-19/12/08

Contents

Table of Contents

About This Guide

What This Guide Contains

The User Guide document describes use of jUDDI – installation and setup.

Audience

This guide is most relevant to engineers who are responsible for setting upjUDDI 3.0 installations.

Prerequisites

None.

Organization

This guide contains the following chapters:

- **Chapter 1, UDDI Introduction**

Documentation Conventions

The following conventions are used in this guide:

Convention	Description
Italic	In paragraph text, italic identifies the titles of documents that are being referenced. When used in conjunction with the Code text described below, italics identify a variable that should be replaced by the user with an actual value.
Bold	Emphasizes items of particular importance.
`Code`	Text that represents programming code.
Function | Function	A path to a function or dialog box within an interface. For example, “Select File | Open.” indicates that you should select the Open function from the File menu.
( ) and |	Parentheses enclose optional items in command syntax. The vertical bar separates syntax items in a list of choices. For example, any of the following three items can be entered in this syntax: `persistPolicy (Never | OnTimer | OnUpdate | NoMoreOftenThan)`
Note: **Caution**:	A note highlights important supplemental information. A caution highlights procedures or information that is necessary to avoid damage to equipment, damage to software, loss of data, or invalid test results.

Table 1 Formatting Conventions

Additional Documentation

None on the subject.

Contacting Us

Email: juddi-user@ws.apache.org

Chapter 1

Setup

Introduction

Within jUDDI, there are three downloadable files (juddi-core.jar, juddi.war, and juddi-tomcat.zip). You should determine which one to use depending on what level of integration you want with your application and your platform / server choices.

Using the JAR

The juddi-core module produces a JAR which contains the jUDDI source and a jUDDI persistence.xml configuration. jUDDI's persistence is being actively tested with both OpenJPA and with Hibernate.

If you are going to use only the JAR, you would need to directly insert objects into jUDDI through the database backend or persistence layer, or configure your own Webservice provider with the provided WSDL files and classes.

Using the WAR files

As with the JAR, you need to make a decision on what framework you would like to use when building the WAR. There will eventually be two WAR files shipped – one using CXF and one using Axis 2. For the alpha release, only CXF has been tested thoroughly.

Simple copy the WAR to the deploy folder of your server (this release has been tested under Apache Tomcat 5.5.23), start your server, and follow the directions under “using jUDDI as a Web Service”.

Using the Tomcat Bundle

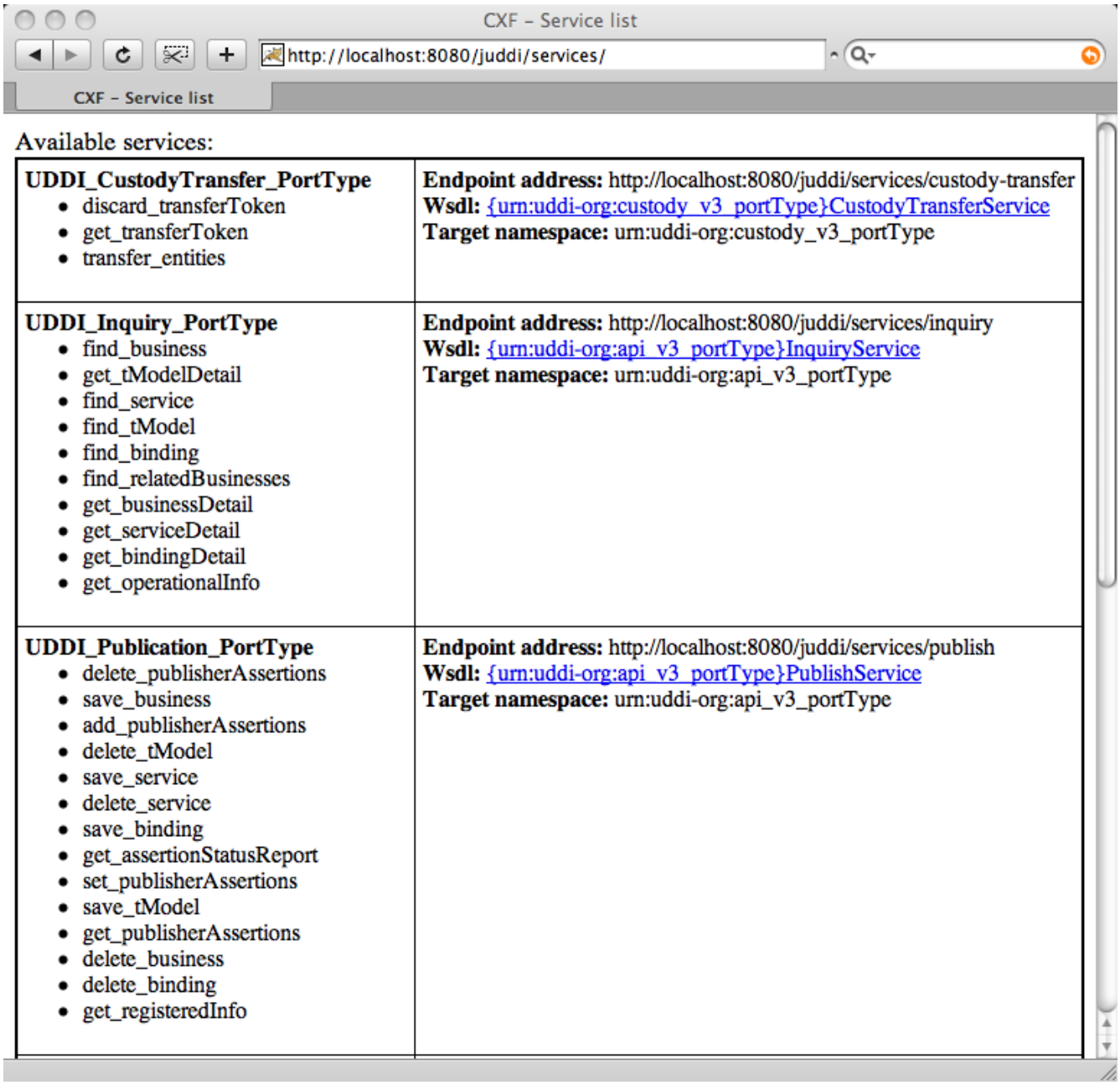
The jUDDI Tomcat bundle packages up the jUDDI WAR, Apache Derby, and a few necessary configuration files and provides the user with a pre-configured jUDDI instance. By default, the Hibernate is used as the persistence layer and CXF is used as a Web Serice framework.

To get started using the Tomcat bundle, unzip the juddi-tomcat-bundle.zip, and start Tomcat :

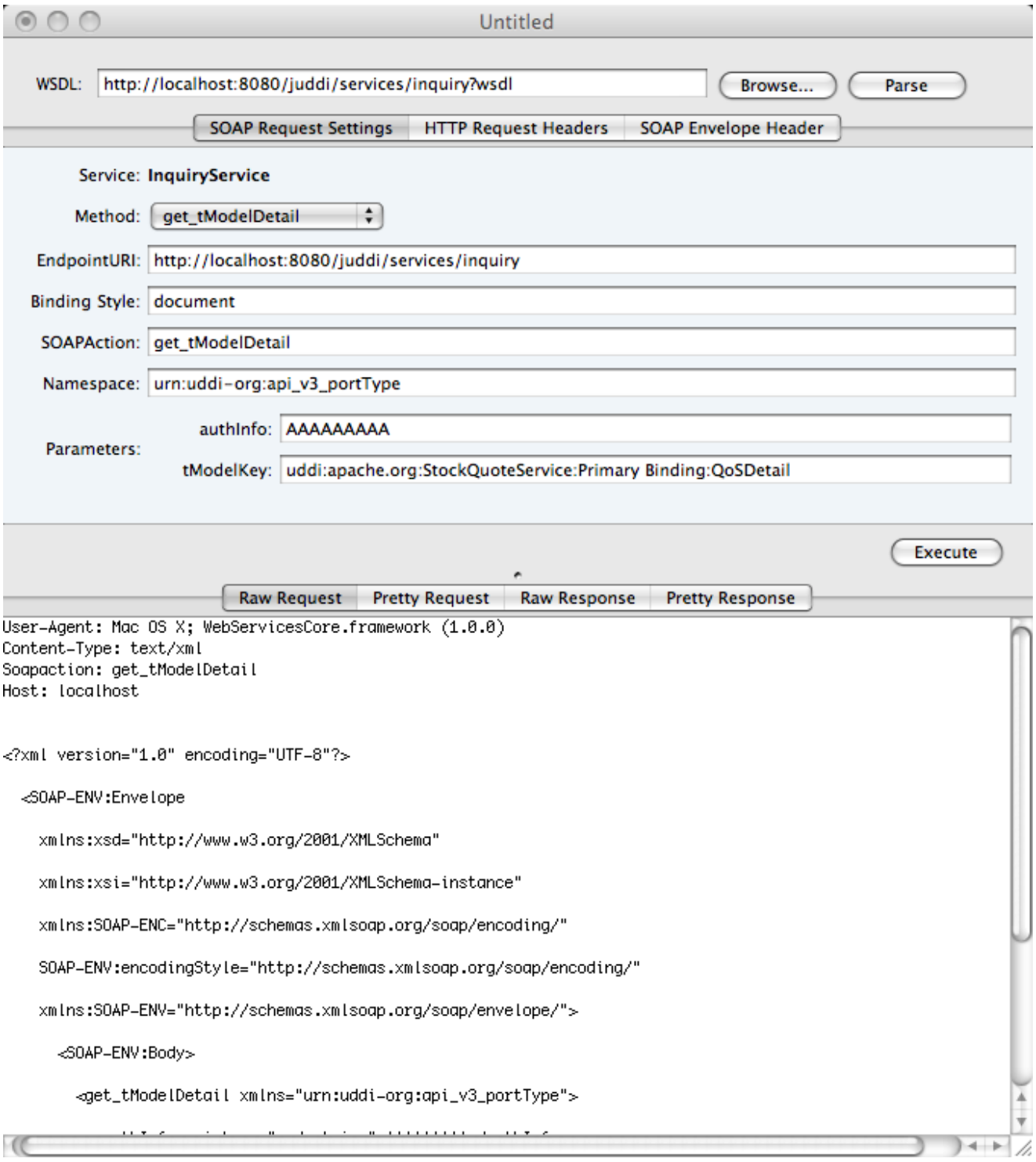
```
% cd apache-tomcat-5.5.23/bin
% ./startup.sh
```

Using jUDDI as Web Service

Browse to http://localhost:8080/juddi/services



The services page shows you the available endpoints and methods available. Using any SOAP client, you should be able to send some sample requests to jUDDI to test:


Untitled
WSDL:
http://localhost:8080/juddi/services/inquiry?wsdl
Browse...
Parse
SOAP Request Settings
HTTP Request Headers
SOAP Envelope Header
Service: InquiryService
Method:
get_tModelDetail
EndpointURI:
http://localhost:8080/juddi/services/inquiry
Binding Style:
document
SOAPAction:
get_tModelDetail
Namespace:
urn:uddi-org:api_v3_portType
Parameters:
authInfo:
AAAAAAAAA
tModelKey:
uddi:apache.org:StockQuoteService:Primary Binding:QoSDetail
Execute
Raw Request
Pretty Request
Raw Response
Pretty Response
User-Agent: Mac OS X; WebServicesCore.framework (1.0.0)
Content-Type: text/xml
Soapaction: get_tModelDetail
Host: localhost
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<get_tModelDetail xmlns="urn:uddi-org:api_v3_portType">

Using jUDDI with your application

As of the Alpha release, two of the UDDI v3 APIs should be active within jUDDI : inquiry and publish.

Chapter 2

Authentication

Introduction

In order to enforce proper write access to jUDDI, each request to jUDDI needs a valid authToken. Note that read access is not restricted and therefore queries into the registries are not restricted.

To obtain a valid authToken a getAuthToken() request must be made, where a GetAuthToken object is passed. On the GetAuthToken object a userid and credential (password) needs to be set.

```
org.uddi.api_v3.GetAuthToken ga = new org.uddi.api_v3.GetAuthToken();
ga.setUserID(pubId);
ga.setCred("");

org.uddi.api_v3.AuthToken token = securityService.getAuthToken(ga);
```

The property `juddi.auth` in the `juddi.properties` configuration file can be used to configure how jUDDI is going to check the credentials passed in on the GetAuthToken request. By default jUDDI uses the DefaultAuthentication implementation. You can provide your own authentication implementation or use any of the ones mention below. The implementation needs to implement the `org.apache.juddi.auth.Authenticator` interface, and `juddi.auth` property should refer to the implementation class.

Default Authentication

The default authentication is simply a pass-though. The password is *not* used and a authToken is always returned. The DefaultAuthentication implement does not use a credential store.

```
juddi.auth = org.apache.juddi.auth.DefaultAuthentication
```

XMLDocAuthentication

The XMLDocAuthentication implementation needs a XML file on the classpath. The juddi.properties file would need to look like

```
juddi.auth = org.apache.juddi.auth.XMLDocAuthentication
juddi.usersfile = juddi-users.xml
```

where the name of the xml can be provided but it defaults to `juddi-users.xml`, and the content of the file would looks something like

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<juddi-users>
        <user userid="anou_mana" password="password" />
        <user userid="bozo" password="clown" />
        <user userid="sviens" password="password" />
</juddi-users>
```

CryptedXMLDocAuthentication

The CryptedXMLDocAuthentication implementation is similar to the XMLDocAuthentication implementation, but the passwords are encrypted

```
juddi.auth = org.apache.juddi.auth.CryptedXMLDocAuthentication
juddi.usersfile = juddi-users-encrypted.xml
```

```
juddi.cryptor = org.apache.juddi.cryptor.DefaultCryptor
```

where the name user credential file is `juddi-users-encrypted.xml`, and the content of the file would looks something like

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<juddi-users>
    <user userid="anou_mana" password="+j/kXkZJftwTFTBH6Cf6IQ=="/>
    <user userid="bozo" password="Na2Ait+2aW0="/>
    <user userid="sviens" password="+j/kXkZJftwTFTBH6Cf6IQ=="/>
</juddi-users>
```

The DefaultCryptor implementation uses `BEWithMD5AndDES` and `Base64` to encrypt the passwords. Note that the code in the `AuthenticatorTest` can be used to learn more about how to use this Authenticator implementation. You can plugin your own encryption algorithm by implementing the `org.apache.juddi.cryptor.Cryptor` interface and referencing your implementation class in the `juddi.cryptor` property.

JBoss Authentication

Finally is it possible to hook up to third party credential stores. If for example jUDDI is deployed to the JBoss Application server it is possible to hook up to it's authentication machinery. The `JBossAuthenticator` class is provided in the `docs/examples/auth` directory. This class enables juddi deployments on JBoss use a server security domain to authenticate users.

To use this class you must add the following properties to the juddi.properties file:

```
juddi.auth=org.apache.juddi.auth.JBossAuthenticator
juddi.securityDomain=java:/jaas/other
```

The `juddi.auth` property plugs the JbossAuthenticator class into the juddi the Authenticator framework. The `juddi.sercuity.domain`, configures the JBossAuthenticator class where it can lookup the application server's security domain, which it will use to perform the authentication. Note that JBoss creates one security domain for each application policy element on the `$JBOSS_HOME/server/default/conf/login-config.xml` file, which gets bound to the server JNDI tree with name `java:/jaas/<application-policy-name>`. If a lookup refers to a non existent application policy it defaults to a policy named `other`.

Index