

# **jUDDI 3.0**

---

## **User Guide**

ASF-JUDDI-USRGUIDE-19/12/08

# Contents

## Table of Contents

<b>Contents.....</b>	<b>2</b>	<b>Setup.....</b>	<b>5</b>
<b>About This Guide.....</b>	<b>3</b>	Introduction.....	5
What This Guide Contains.....	3	Using the JAR.....	6
Audience.....	3	Using the WAR files.....	7
Prerequisites.....	3	Using the Tomcat Bundle.....	8
Organization.....	3	Using jUDDI as Web Service.....	9
Documentation Conventions.....	3	Using jUDDI with your application.....	11
Additional Documentation.....	4	<b>Index.....</b>	<b>12</b>
Contacting Us.....	4		

# 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 up jUDDI 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
<i>Italic</i>	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.
<b>Bold</b>	Emphasizes items of particular importance.
Code	Text that represents programming code.
<b>Function   Function</b>	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:  <code>persistPolicy (Never   OnTimer   OnUpdate   NoMoreOftenThan)</code>
<b>Note:</b>	A note highlights important supplemental information.
<b>Caution:</b>	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](mailto:juddi-user@ws.apache.org)

# 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 Service framework.

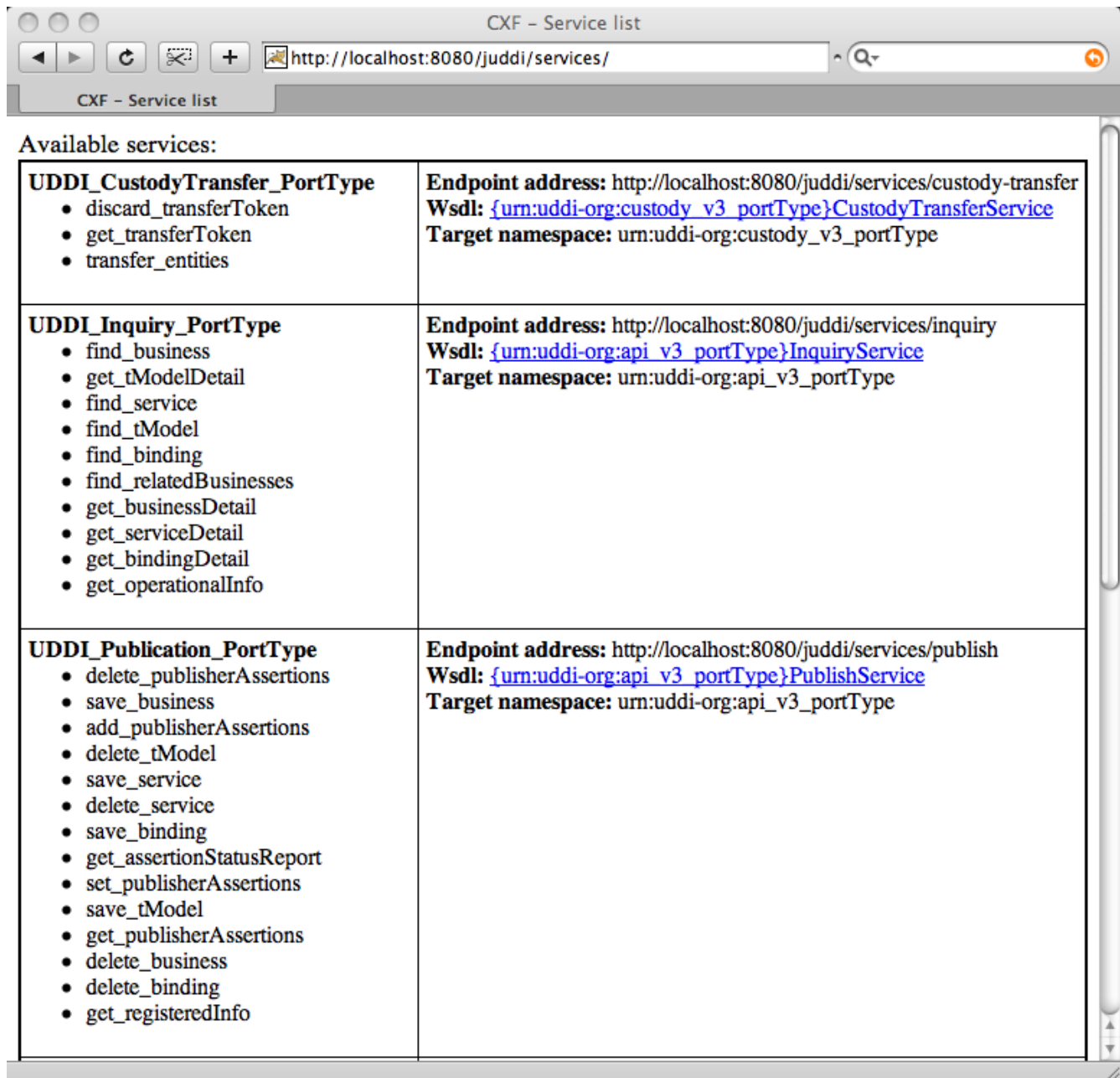
To get started using the Tomcat bundle, unzip the juddi-tomcat-bundle.zip, and start Tomcat :

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



## Using jUDDI as Web Service

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



Available services:	
<b>UDDI_CustodyTransfer_PortType</b> <ul style="list-style-type: none"><li>• discard_transferToken</li><li>• get_transferToken</li><li>• transfer_entities</li></ul>	<b>Endpoint address:</b> <a href="http://localhost:8080/juddi/services/custody-transfer">http://localhost:8080/juddi/services/custody-transfer</a> <b>WSDL:</b> <a href="#">{urn:uddi-org:custody_v3_portType}CustodyTransferService</a> <b>Target namespace:</b> urn:uddi-org:custody_v3_portType
<b>UDDI_Inquiry_PortType</b> <ul style="list-style-type: none"><li>• find_business</li><li>• get_tModelDetail</li><li>• find_service</li><li>• find_tModel</li><li>• find_binding</li><li>• find_relatedBusinesses</li><li>• get_businessDetail</li><li>• get_serviceDetail</li><li>• get_bindingDetail</li><li>• get_operationalInfo</li></ul>	<b>Endpoint address:</b> <a href="http://localhost:8080/juddi/services/inquiry">http://localhost:8080/juddi/services/inquiry</a> <b>WSDL:</b> <a href="#">{urn:uddi-org:api_v3_portType}InquiryService</a> <b>Target namespace:</b> urn:uddi-org:api_v3_portType
<b>UDDI_Publication_PortType</b> <ul style="list-style-type: none"><li>• delete_publisherAssertions</li><li>• save_business</li><li>• add_publisherAssertions</li><li>• delete_tModel</li><li>• save_service</li><li>• delete_service</li><li>• save_binding</li><li>• get_assertionStatusReport</li><li>• set_publisherAssertions</li><li>• save_tModel</li><li>• get_publisherAssertions</li><li>• delete_business</li><li>• delete_binding</li><li>• get_registeredInfo</li></ul>	<b>Endpoint address:</b> <a href="http://localhost:8080/juddi/services/publish">http://localhost:8080/juddi/services/publish</a> <b>WSDL:</b> <a href="#">{urn:uddi-org:api_v3_portType}PublishService</a> <b>Target namespace:</b> urn:uddi-org:api_v3_portType

The services page shows you the available endpoints and methods available. You should be able to send some sample requests to jUDDI to test:

Using any SOAP client, you

Untitled

WSDL:

Service: **InquiryService**

Method:

EndpointURI:

Binding Style:

SOAPAction:

Namespace:

Parameters:

authInfo:

tModelKey:

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.

# Authentication

## Introduction

---

## Default Authentication

---

## XMLDocAuthentication

---

## CryptedXMLDocAuthentication

---

## JBoss Authentication

---

The `JBossAuthenticator` class is provided in the `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

---