Systinet Developers' Corner Tutorial

Subscriptions - A UDDI V3 Mantra

Arulazi Dhesiaseelan aruld@acm.org

October 14, 2003

Legal Notice

This document and the information contained herein are the property of Systinet Corporation and shall not be reproduced or copied in whole or in part without written permission of Systinet Corp.

Disclaimer

The information in this document is preliminary and is subject to change without notice and should not be construed as a commitment by Systinet Corporation. SYSTINET CORPORATION SHALL HAVE NO LIABILITY FOR THIS DOCUMENT, INCLUDING ANY LIABILITY FOR NEGLIGENCE. SYSTINET CORPORATION MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR IN ANY OTHER COMMUNICATION. SYSTINET CORPORATION SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

Acknowledgements

Systinet, WASP, "The Web Services Infrastructure Company" and "Web Services That Work" are trademarks of Systinet Corp.

Microsoft, Windows and Windows NT and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names mentioned in these documents may be trademarks or service marks of others.

Contents

1 Introduction

Universal Description, Discovery and Integration (UDDI) is gaining momentum due to its wide spread adoption in Web Services technology. It has become the de facto standard for Web Service publishing and discovery mechanisms. UDDI Version 2(V2) has been promoted as an OASIS standard. UDDI Version 3(V3) specification is the "UDDI Spec TC Committee Specification" in OASIS. UDDI V3 adds a lot of valuable additions to the existing UDDI V2 specifications. The UDDI V3 effort has primarily focused in overcoming the short falls faced by adopters and registry implementers. The major features of UDDI V3 include:

- Support for a Multi-Registry Environment
- Support for Digital Signatures
- SuperInquiry Features
- Policy Abstractions
- Subscriptions

In this tutorial, I will be discussing "UDDI Subscriptions", a key V3 feature. I assume that readers are familiar with XML Schemas and UDDI terminologies, in order to appreciate the content. As of this writing, only Systinet's WASP UDDI 4.5.2 provides Subscriptions support as a part of its existing UDDI V2 private registry offering. Systinet's WASP UDDI is a secure, platform-independent UDDI registry service designed for private use within enterprises or between trusted parties, thus enabling collaborative commerce.

This tutorial uses Systinet's WASP UDDI 4.5.2 as a standalone server with embedded HSQL installed in a Windows 2000 environment. For more information on how to set up WASP UDDI to run this demo see Appendix A.

2 Subscription Types

The UDDI V3 subscription mechanism enables you to track changes to the core UDDI data structures that are registered in the UDDI registry. This can be either through synchronous requests to the registry or asynchronous notifications from the registry to the client. This is illustrated in Figure 1.

Subscribers are allowed to make synchronous requests to the registry in order to get the data that interests them. Prior to this call, they need to save a subscription with filtering criteria which limits the scope of the subscription to a subset of registry data.

On the other hand, when notifications happen asynchronously subscribers implement a subscription listener service which notifies them periodically whenever a change occurs in the registry data. The notification interval is specified in the subscription request, along with binding information. You need to specify the bindingKey within the save subscription request, whose accessPoint holds the SOAP end point URL or an e-mail address. You need to process the incoming HTTP/SOAP or e-mail notifications as desired.

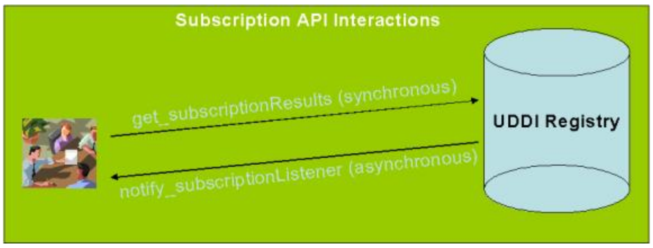
Subscription API Interactions
get_subscriptionResults (synchronous)
notify_subscriptionListener (asynchronous)
UDDI Registry

Figure 1: Subscription Types

Whenever a new subscription is created, a subscription key is assigned which is unique across the registry. Subscriptions are owned by the subscriber who creates them.

Figure 1: Subscription types.

You can track the following registry data with the help of the subscription APIs.

- businessEntity
- businessService
- bindingTemplate
- tModel
- related businessEntity
- publisherAssertion

WASP UDDI's Subscription mechanism lets users monitor new, changed, and deleted entries of the above mentioned data structures. Currently, WASP UDDI 4.5.2 provides this monitoring facility for all the structures shown above, except for publisher assertions.

3 Subscription Parameters

There are a number of subscription parameters that are defined in the UDDI Subscription API Specification. These parameters help the users to monitor the changes to the registry data that interests them. Listing 0 defines the XML schema for the subscription parameters.

subscriptionKey: This is the subscription key that gets generated when a user saves a subscription to the registry. This key is also used to renew or change an existing subscription. The XML schema type for this element is xsd:anyURI.

subscriptionFilter: This parameter specifies the filtering criteria for the data structure that interests the user. The filter can contain any of the UDDI get_xx and find_xx APIs. WASP UDDI uses the following APIs that can be used within the subscription filter.

- find_business

- find_relatedBusinesses
- find_services
- find_bindings
- find_tmodel
- get_businessDetail
- get_serviceDetail
- get_bindingDetail
- get_tModelDetail

expiresAfter: This parameter specifies the time after which the subscription becomes invalid. The XML Schema type for this element is xsd:dateTime.

notificationInterval: This parameter is required only when asynchronous notifications are used. This parameter specifies how often notifications are sent to the subscriber. The XML Schema type for this element is xsd:duration.

maxEntities: This parameter specifies the maximum number of entities in a notification which are returned to a subscription listener. The XML Schema type for this element is xsd:int.

bindingKey: This parameter is required only when asynchronous notifications are used. This parameter specifies the bindingTemplate which the node uses to deliver notifications to subscription listeners. Only HTTP and SMTP transports are supported to deliver notifications. When using HTTP, the Web Service defined should implement the notify_subscriptionListener API, so that notifications are delivered to the subscription listeners by invoking the Web Service. On the other hand, when using SMTP, the notifications are delivered through e-mail which is specified in the bindingTemplate.

brief: This parameter specifies the level of information that are returned to a subscription listener. When this parameter is set to "true", the subscription results will return a keyBag that matches the entries in the subscriptionFilter. For example, if find_binding call is specified in the subscriptionFilter element, then the subscription will contain only the binding keys when the brief is enabled. The XML schema type for this element is xsd:boolean.

Listing 0: XML Schema definition for subscription parameters.

```
<xsd:element
    name="subscriptionKey" type="uddi_sub:subscriptionKey"
        final="restriction"/>
<xsd:simpleType
    name="subscriptionKey" final="restriction">
            <xsd:restriction
    base="xsd:anyURI"/>
</xsd:simpleType>
<xsd:element
    name="subscriptionFilter" type="uddi_sub:
        subscriptionFilter" final="restriction"/>
<xsd:complexType
```

```
        name="subscriptionFilter" final="restriction">
                <xsd:choice>
                            <xsd:element
        ref="uddi:find_binding"/>
                            <xsd:element
        ref="uddi:find_business"/>
                            <xsd:element
        ref="uddi:find_relatedBusinesses"/>
                            <xsd:element
        ref="uddi:find_service"/>
                            <xsd:element
        ref="uddi:find_tModel"/>
                            <xsd:element
        ref="uddi:get_bindingDetail"/>
                            <xsd:element
        ref="uddi:get_businessDetail"/>
                            <xsd:element
        ref="uddi:get_serviceDetail"/>
                            <xsd:element
        ref="uddi:get_tModelDetail"/>
                            <xsd:element
        ref="uddi:get_assertionStatusReport"/>
                </xsd:choice>
    </xsd:complexType>
    <xsd:element
        name="notificationInterval"
        type="uddi_sub: notificationInterval"
        final="restriction"/>
    <xsd: simpleType
        name="notificationInterval" final="restriction">
                <xsd:restriction
        base="xsd:duration"/>
    </xsd:simpleType>
    <xsd:element
        name="maxEntities" type="uddi_sub:maxEntities"/>
    <xsd:simpleType
        name="maxEntities" final="restriction">
                <xsd:restriction
        base="xsd:int"/>
    </xsd:simpleType>
    <xsd:element
        name="expiresAfter" type="uddi_sub:expiresAfter" final="
            restriction"/>
    <xsd:simpleType
        name="expiresAfter" final="restriction">
                <xsd:restriction
        base="xsd:dateTime"/>
    </xsd:simpleType>
    <xsd:element
        name="brief" type="uddi_sub:brief" final="restriction"/>
    <xsd:simpleType
        name="brief" final="restriction">
                <xsd:restriction
        base="xsd:boolean"/>
```

```
</xsd:simpleType>
```

4 Server-side APIs

There are four server-side APIs that are defined in the subscription specification. The server checks for changes and sends notification to its subscribers when it is explicitly requested through these API calls. Each of these server-side APIs is synchronous in nature.

1. save_subscription
2. get_subscriptions
3. get_subscriptionResults
4. delete_subscription

4.1 save_subscription

This API saves a new subscription or updates an existing subscription. This API call returns a subscription structure. The XML schema for this API is defined in Listing 1.

Listing 1: XML Schema definition for save_subscription API.

```
<xsd:element
    name="save\_subscription" type="uddi_sub:save_subscription
        " final="restriction"/>
<xsd:complexType
    name="save_subscription" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi:authInfo" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:subscription" maxOccurs="unbounded"/>
            </xsd:sequence>
</xsd:complexType>
<xsd:element
    name="subscription" type="uddi_sub:subscription" final="
        restriction"/>
<xsd:complexType
    name="subscription" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi_sub:subscriptionKey" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:subscriptionFilter" minOccurs="0"/>
                        <xsd:element
    ref="uddi:bindingKey" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:notificationInterval" minOccurs="0"/>
```

```
                        <xsd:element
    ref="uddi_sub:maxEntities" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:expiresAfter" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute
    name="brief" type="uddi_sub:brief" use="optional"/>
</xsd:complexType>
```

4.2 get_subscriptions

This API returns the complete list of existing subscriptions owned by the subscriber. Listing 2 defines the XML schema for this API.

Listing 2: XML Schema definition for get_subscriptions API.

```
<xsd:element
    name="get\_subscriptions" type="uddi_sub:get_subscriptions
        " final="restriction"/>
<xsd:complexType
    name="get_subscriptions" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi:authInfo" minOccurs="0"/>
            </xsd:sequence>
</xsd:complexType>
```

4.3 get_subscriptionResults

This API helps the subscriber to retrieve all the information associated with an existing subscription. With the help of the element coveragePeriod, the subscriber can obtain the changes that occurred during the specified time period. The outcome of this call is a subscriptionResultList. Listing 3 defines the XML schema for this API.

Listing 3: XML Schema definition for get_subscriptionResults API.

```
<xsd:element
    name="get_subscriptionResults"
    type="uddi_sub:get_subscriptionResults"
    final="restriction"/>
<xsd:complexType
    name="get_subscriptionResults" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi:authInfo" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:subscriptionKey"/>
                        <xsd:element
    ref="uddi_sub:coveragePeriod"/>
                        <xsd:element
    ref="uddi_sub:chunkToken" minOccurs="0"/>
            </xsd:sequence>
</xsd:complexType>
<xsd:element
    name="coveragePeriod" type="uddi_sub:coveragePeriod" final
        ="restriction"/>
<xsd:complexType
    name="coveragePeriod" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi_sub:startPoint" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:endPoint" minOccurs="0"/>
            </xsd:sequence>
</xsd:complexType>
<xsd:element
    name="chunkToken" type="uddi_sub:chunkToken" final="
        restriction"/>
<xsd:simpleType
    name="chunkToken">
            <xsd:restriction
    base="xsd:string">
                        <xsd:maxLength
    value="255"/>
            </xsd:restriction>
</xsd:simpleType>
```

4.4 delete_subscription

This API deletes an existing subscription. Upon successful completion, this API returns a dispositionReport indicating E_success. Listing 4 defines the XML schema for this API.

Listing 4: XML Schema definition for delete_subscription API.

```
<xsd:element
    name="delete_subscription"
    type="uddi_sub:delete_subscription"
    final="restriction"/>
<xsd:complexType
    name="delete_subscription" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi:authInfo" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:subscriptionKey" maxOccurs="unbounded"/>
            </xsd:sequence>
</xsd:complexType>
```

5 Subscription Demo using Server-side APIs

This section provides an end-to-end demo by using the server-side subscription APIs. The following steps determine a simple subscription API usage scenario.

- Step 1 : Save a business entity with name=’BE1’
- Step 2 : Save a subscription with filter find business by name=’BE
- Step 3 : get_subscriptionResult call should contain business ’BE1’
- Step 4 : Delete business entity with name=’BE1’
- Step 5: get_subscriptionResult call should contain deletion of business ’BE1’
- Step 6 : Delete subscription should return success.

Listing 5: Saves a subscription with subscriptionFilter having find_business call.

```
<subscriptions>
  <subscription>
    <subscriptionKey>62858f10-bac6-11d7-8822-b8a03c50a862</
        subscriptionKey>
    <subscriptionFilter>
      <find_business
    generic="2.1" xmlns="urn:systinet-org:api_acl">
        <name
    xmlns="">BE</name>
      </find_business>
    </subscriptionFilter>
    <maxEntities>10</maxEntities>
    <expiresAfter>2003-08-07T01:05:53.711+09:00</expiresAfter>
  </subscription>
</subscriptions>
```

Listing 6: get_subscriptionResults should return the business entity registered in Step 1.

```
<subscriptionResultsList
    someResultsUnavailable="false">
  <chunkToken>0</chunkToken>
  <coveragePeriod>
    <startPoint>2003-07-21T00:23:45.012+09:00</startPoint>
    <endPoint>2003-07-21T00:25:45.062+09:00</endPoint>
  </coveragePeriod>
  <subscription
    brief="false">
    <subscriptionKey>62858f10-bac6-11d7-8822-b8a03c50a862</
        subscriptionKey>
    <subscriptionFilter>
      <find_business
    generic="2.1" xmlns="urn:systinet-org:api_acl">
        <name
    xmlns="">BE</name>
      </find_business>
    </subscriptionFilter>
    <maxEntities>10</maxEntities>
    <expiresAfter>2003–08–07T01:05:53.711+09:00</expiresAfter>
  </subscription>
  <businessList
    generic="2.0" operator="" xmlns="urn:systinet-org:api_acl"
        >
    <businessInfos
    xmlns="">
      <businessInfo
    businessKey="6e3d48c0-bac6-11d7-8822-b8a03c50a862">
        <name
    xml:lang="en">BE1</name>
        <serviceInfos/>
      </businessInfo>
    </businessInfos>
  </businessList>
</subscriptionResultsList>
```

Listing 7: get_subscriptionResults should return the business entity deleted in Step4.

```
<subscriptionResultsList
    someResultsUnavailable="false">
  <chunkToken>0</chunkToken>
  <coveragePeriod>
    <startPoint>2003–07–21T00:23:45.743+09:00</startPoint>
    <endPoint>2003–07–21T00:25:45.773+09:00</endPoint>
  </coveragePeriod>
  <subscription
    brief="false">
    <subscriptionKey>62858f10–bac6–11d7–8822–b8a03c50a862</
        subscriptionKey>
    <subscriptionFilter>
      <find_business
    generic="2.1" xmlns="urn:systinet-org:api_acl">
        <name
    xmlns="">BE</name>
      </find_business>
    </subscriptionFilter>
    <maxEntities>10</maxEntities>
    <expiresAfter>2003–08–07T01:05:53.711+09:00</expiresAfter>
  </subscription>
  <keyBag>
    <deleted>true</deleted>
    <businessKey>6e3d48c0–bac6–11d7–8822–b8a03c50a862</
        businessKey>
  </keyBag>
</subscriptionResultsList>
```

Listing 8: delete_subscription returning a dispositionReport with E_success.

```
<dispositionReport
    generic="2.0" operator="" xmlns="urn:uddi-org:api_v2">
  <result xmlns="" errno="0">
    <errInfo
    errCode="E_success">No failure occurred.</errInfo>
  </result>
</dispositionReport>
```

6 Client-side API

The client-side API notify_subscriptionListener makes asynchronous notifications to the subscribers. In this approach, the server periodically checks for changes and then notifies the subscription listener via HTTP or SMTP. The periodicity is defined in the notificationInterval parameter.

6.1 notify_subscriptionListener

SubscriptionListener is a Web Service interface. The subscriber implements this interface and specifies the binding information in a subscription. This enables the node to deliver asynchronous notifications to the subscription listeners by invoking a Web Service. Upon successful completion, this API returns a dispositionReport indicating E_success. Listing 9 defines the XML schema for this API.

Listing 9: XML Schema definition for notify_subscriptionListener API.

```
<xsd:element
    name="notify_subscriptionListener"
    type="uddi_subr:notify_subscriptionListener"
    final="restriction"/>
<xsd:complexType
    name="notify_subscriptionListener" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi:authInfo" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:subscriptionResultsList"/>
            </xsd:sequence>
</xsd:complexType>
<xsd:element
    name="subscriptionResultsList"
    type="uddi_sub:subscriptionResultsList"
    final="restriction"/>
<xsd:complexType
    name="subscriptionResultsList" final="restriction">
            <xsd:sequence>
                        <xsd:element
    ref="uddi_sub:chunkToken" minOccurs="0"/>
                        <xsd:element
    ref="uddi_sub:coveragePeriod"/>
                        <xsd:element
    ref="uddi_sub:subscription"/>
                        <xsd:element
    ref="uddi:bindingDetail" minOccurs="0"/>
                        <xsd:element
    ref="uddi:businessDetail" minOccurs="0"/>
                        <xsd:element
    ref="uddi:serviceDetail" minOccurs="0"/>
                        <xsd:element
    ref="uddi:tModelDetail" minOccurs="0"/>
                        <xsd:element
```

```
        ref="uddi:businessList" minOccurs="0"/>
                                <xsd:element
        ref="uddi:relatedBusinessesList" minOccurs="0"/>
                                <xsd:element
        ref="uddi:serviceList" minOccurs="0"/>
                                <xsd:element
        ref="uddi:tModelList" minOccurs="0"/>
                                <xsd:element
        ref="uddi:assertionStatusReport" minOccurs="0"/>
                                <xsd:element
        ref="uddi_sub:keyBag" minOccurs="0" maxOccurs="unbounded"
            />
                    </xsd:sequence>
                    <xsd:attribute
        name="someResultsUnavailable" type="xsd:boolean" use="
            optional"/>
    </xsd:complexType>
    <xsd:element
        name="keyBag" type="uddi_sub:keyBag" final="restriction"/>
    <xsd:complexType
        name="keyBag" final="restriction">
                <xsd:sequence>
                                <xsd:element
        ref="uddi_sub:deleted"/>
                                <xsd:choice>
                                            <xsd:element
        ref="uddi:tModelKey" maxOccurs="unbounded"/>
                                            <xsd:element
        ref="uddi:businessKey" maxOccurs="unbounded"/>
                                            <xsd:element
        ref="uddi:serviceKey" maxOccurs="unbounded"/>
                                            <xsd:element
        ref="uddi:bindingKey" maxOccurs="unbounded"/>
                                </xsd:choice>
                    </xsd:sequence>
    </xsd:complexType>
```

7 Subscription Demo using Client-side API

This section provides an end-to-end demo by using the client-side subscription API for asynchronous notifications. The following steps determine a simple usage scenario for this subscription API.

STEP 1: Create a businessEntity.

STEP 2: Create a businessService with bindingTemplates which can represent a Web Service endpoint or E-mail address.

STEP 3: Create a subscription which holds the bindingKey in it.

STEP 4: Node makes notify_subscriptionListener call based on the subscription created in STEP 3.

Listing 10: Saves a binding with Web service endpoint.

```
<save_binding
    generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo xmlns="">eJytk1uPqkgUhd/7V....</authInfo>
  <bindingTemplate
    xmlns="" bindingKey="" serviceKey="3246fb90-bb66-11d7-ae30
        -b8a03c50a862">
    <description>Web
    Service binding for my subscription.</description>
    <accessPoint
    URLType="http">://www.systinet.com/services/
        notify_subscriptionListener</accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo
    tModelKey="uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"/>
    </tModelInstanceDetails>
  </bindingTemplate>
</save_binding>
```

Listing 11: Saves a binding with E-mail address.

```
<save_binding
    generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo xmlns="">eJytk1uPqkgUhd/7V....</authInfo>
  <bindingTemplate
    xmlns="" bindingKey="" serviceKey="3246fb90-bb66-11d7-ae30
        -b8a03c50a862">
    <description>E—mail
    binding for my subscription.</description>
    <accessPoint
    URLType="email">mailto:systinetSubscriber@systinet.com</
        accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo
    tModelKey="uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"/>
    </tModelInstanceDetails>
  </bindingTemplate>
</save_binding>
```

Listing 12: Saves a subscription with bindingKey.

```
<save_subscription>
  <authInfo>eJytk1uPqkgUhd....</authInfo>
  <subscription
    brief="true">
    <subscriptionFilter>
      <find_business
    generic="2.1" xmlns="urn:systinet-org:api_acl">
        <name
    xmlns="">WASP UDDI Subscription</name>
      </find_business>
    </subscriptionFilter>
    <bindingKey>51ac4ad0–c9ad–11d7–bc84–b8a03c50a862</
       bindingKey>
    <notificationInterval>P1D</notificationInterval>
    <maxEntities>10</maxEntities>
    <expiresAfter>2003–08–26T13:33:53.344+09:00</expiresAfter>
  </subscription>
</save_subscription>
```

Listing 13: Node's response to the subscription listener web service.

```
<notify_subscriptionListener>
  <subscriptionResultsList>
    <coveragePeriod>
      <startPoint>20030801T00:00:00</startPoint>
      <endPoint>20030831T00:00:00</endPoint>
    </coveragePeriod>
    <subscription
    brief="true">
      <subscriptionFilter>
        <find_business
    generic="2.1" xmlns="urn:systinet-org:api_acl">
          <name
    xmlns="">WASP UDDI Subscription</name>
        </find_business>
      </subscriptionFilter>
      <bindingKey>51ac4ad0–c9ad–11d7–bc84–b8a03c50a862</
          bindingKey>
      <notificationInterval>P1D</notificationInterval>
      <maxEntities>10</maxEntities>
      <expiresAfter>2003–08–26T13:33:53.344+09:00</
          expiresAfter>
    </subscription>
    <keyBag>
      <deleted>false</deleted>
      <businessKey>matchingKey1</businessKey>
    </keyBag>
  </subscriptionResultsList>
</notify_subscriptionListener>
```

8 WASP UDDI Subscription APIs

WASP UDDI provides a set of Java APIs for using the subscription mechanism within its UDDI registry. The interfaces and classes are defined in the package "org.systinet.myuddi.subscription".

8.1 Saving a subscription

Listing 14 shows the usage of WASP UDDI subscription APIs when creating a subscription in the UDDI Registry.

Listing 14: Usage of the WASP UDDI's save_subscription call.

```
String subscriptionKey = null;

// Create a holder for all parameters of save_subscription
     API call.
SaveSubscription saveSubscription = new SaveSubscription()
    ;
```

```
// Create a holder structure to accomodate information
    related to the subscription.
Subscription subscription = new Subscription();

// Create a Subscription that will expire after one year
long currentTime = System.currentTimeMillis();
long year = 365∗24∗60∗60∗1000;
subscription.setExpiresAfter(new Date( currentTime + year
     ));

// Set notification should contain only 10 records
subscription.setMaxEntities(new MaxEntities(10));
subscription.setBrief(new Brief(true));

// Set the notification interval of the subscription to 1
    day.
// This is used in asynchronous subscriptions.
//subscription.setNotificationInterval(new Duration("P1D")
    );
// Attach the binding key to the subscription, so that
    notifications
// are sent to services that are specified in the
    accessPoint.
//subscription.setBindingKey(new BindingKey("3409e140–bb66
    –11d7–ae30–b8a03c50a862"));

// Create subscription filter structure for a subset of
    registry data.
SubscriptionFilter subscriptionFilter = new
    SubscriptionFilter();

// Create and fill find business structure
FindBusiness findBusiness = new FindBusiness();
findBusiness.addName(new Name("WASP UDDI Subscription"));
// subscriptionFilter holds findBusiness ...
subscriptionFilter.setFindBusiness(findBusiness);

// Set the find_business query as subscriptionFilter to
    the subscription
subscription.setSubscriptionFilter(subscriptionFilter);

Subscriptions subscriptions = new Subscriptions();
subscriptions.add(subscription);

saveSubscription.setSubscriptions(subscriptions);
saveSubscription.setAuthInfo(authToken.getAuthInfo());
Subscriptions response = subscriptionService.
    save_subscription(saveSubscription);
subscriptionKey = response.first().getSubscriptionKey().
    getValue();
System.out.println("Created Subscription with key : " +
    subscriptionKey);
```

8.2 Getting Subscriptions

Listing 15 shows the usage of the WASP UDDI subscription APIs to get the subscriptions registered in the UDDI Registry.

Listing 15: Usage of the WASP UDDI's get_subscriptions call.

```
Subscriptions subscriptions = null;
String subscriptionKey = null;
// Fill in the request parameter — authInfo
GetSubscriptions getSubscriptions = new GetSubscriptions();
getSubscriptions.setAuthInfo(authToken.getAuthInfo());

 // Process get_subscriptions request
subscriptions = subscriptionService.get_subscriptions(
    getSubscriptions);
if (subscriptions != null) {
    subscriptionKey = subscriptions.first().
        getSubscriptionKey().getValue();
} else {
    System.out.println("No subscriptions found.");
}
```

8.3 Getting subscription results

Listing 16 shows the usage of the WASP UDDI subscription APIs to get the subscription results based on changes that occurred over a period of time in the UDDI Registry.

Listing 16: Usage of the WASP UDDI's get_subscriptionResults call.

```
 // Fill in the request parameters
 GetSubscriptionResults getSubscriptionResults = new
     GetSubscriptionResults();

 // Set key of existing subscription
getSubscriptionResults.setSubscriptionKey(new SubscriptionKey
    (getSubscriptions()));

 // Changes in last 2 minutes
 long endPoint = System.currentTimeMillis();
 long startPoint = endPoint — 2∗60∗1000;
 getSubscriptionResults.setCoveragePeriod(new CoveragePeriod(
     new Date(startPoint),
                  new Date(endPoint)));
 // Set user identity
 getSubscriptionResults.setAuthInfo(authToken.getAuthInfo());

 // Process get_subscriptionResults request
 SubscriptionResultsList list =
      subscriptionService.get_subscriptionResults(
         getSubscriptionResults);
```

8.4 Deleting a subscription

Listing 17 shows the usage of the WASP UDDI subscription APIs to delete a subscription in the UDDI Registry.

Listing 17: Usage of the WASP UDDI's delete_subscription call.

```
// Create list of subscriptionKeys with only one
   subscriptionKey
// – the given parameter
SubscriptionKeys keys = new SubscriptionKeys();
keys.add(new SubscriptionKey(subscriptionKey));

// Fill in the request parameters: subscriptionKey,
   authToken
DeleteSubscription deleteSubscription =
     new DeleteSubscription(authToken.getAuthInfo(), keys)
         ;

// Process delete_subscription request
DispositionReport dispositionReport =
    subscriptionService.delete_subscription(
       deleteSubscription);
```

9 Subscription Advantage

The subscription mechanism in UDDI V3 has a potential business advantage to its users. This mechanism helps registered users to get up to date information from the registry based on their interests. This has huge potential in the collaborative business environment where businesses participate in a vendor neutral manner.

There are several scenarios where a subscription mechanism helps to achieve business agility. Some of them are detailed below.

9.1 Monitoring existing businesses

Subscription allows for monitoring businesses that are already registered in the registry. A subscriber may be interested in the activity of a certain business to which he/she is subscribed in order to receive updates to that business which includes operations such as deletion/modification.

9.2 Notification about new businesses

One can save a subscription that notifies the user whenever a new business of interest is registered in the registry. This scenario is analogous to web sites posting new jobs or employers seeking resumes matching their requirements. Whenever a new job is created or posted on this website, subscribers or registered users are notified of this job based on their criteria. Also, whenever a subscriber or registered user posts his/her resumes in the web site, notifications are sent to employers seeking this criterion. This is the bi-directional advantage of this scenario which has potential business value to the customers of this job portal.

9.3 Replicating a subset of registry data

Replication and subscription serves different purpose within a UDDI registry. But subscription may replace replication whenever the node is not capable of replication. With the help of subscription, one can replicate a sub set of registry data without using the UDDI replication functionality.

9.4 Maintain data integrity among business partners

Subscription allows for maintaining data integrity among collaborating partners within the business eco system. This helps to achieve trust among the partners thus enabling collaborative commerce in a more efficient manner.

10 Conclusion

With the help of subscriptions, a subset of the registry data can be monitored for a specific business need. Subscription helps businesses to participate in a collaborative environment where they address the needs of a specific industry. More registries should soon become available with this powerful feature in place. The vision of UDDI is becoming a reality. With powerful features such as subscriptions, UDDI will change the way businesses collaborate.

11 References

UDDI Version 3 Features List
http://uddi.org/pubs/uddi_v3_features.htm
UDDI Version 3.0: UDDI Spec Technical Committee Specification, 19 July 2002
http://uddi.org/pubs/uddi-v3.00-published-20020719.htm
WASP UDDI 4.5.2 Product Overview
http://www.systinet.com/products/wasp_uddi/overview
WASP UDDI 4.5.2 Product Download (requires login)
http://www.systinet.com/products/wasp_uddi/download
WASP UDDI 4.5.2 Subscription API Reference Guide
http://www.systinet.com/doc/wasp_uddi/api/org/systinet/myuddi/subscription/package-summary.html
WASP UDDI 4.5.2 Product Guide
http://www.systinet.com/doc/wasp_uddi/uddi/index.html
Rocket ahead with UDDI V3
http://www-106.ibm.com/developerworks/webservices/library/ws-uddiv3/?dwzone=webservices

12 About the Author

Arulazi Dhesiaseelan has been working as a Senior Software Engineer for Hewlett Packard Company, India. He has a Master of Computer Applications Degree from PSG College of Technology, India. He has about 3 years of industry experience. He was also involved in the UDDI4J project hosted at http://uddi4j.org. He has been working on Web Service related technologies such as WSDL, UDDI and SOAP. Currently, he is involved in developing an open service framework for mobile infrastructures. He can be reached at aruld@acm.org.

13 APPENDIX A. Setting up WASP UDDI 4.5.2

Download WASP UDDI 4.5.2 from Systinet Products Web site (http://www.systinet.com/products/wasp_uddi/do
You need to be a registered user of the Systinet web site for downloading the developer version. You need to install JDK 1.4.x as a prerequisite. WASP UDDI 4.5.2 supports leading application servers including BEA WebLogic, IBM WebSphere, Oracle, Orion, Tomcat, Jboss, and Sun ONE Application Server. It supports for leading database engines including Oracle, MS SQL 2000, DB2, PostgreSQL, Sybase, Cloudscape, PointBase, and Hypersonic SQL. In addition to this, WASP 4.5.2 can be run as a standalone server with embedded Hypersonic SQL engine.

Once the installation is successful, open a browser pointing to the URL: http://localhost:8080/uddi/web. You should be seeing the Systinet's WASP UDDI Console. The publishing, inquiry subscription URLs are shown below.

- PUBLISHING_URL=https://localhost:8443/uddi/publishing
- INQUIRY_URL=http://localhost:8080/uddi/inquiry
- SUBSCRIPTION_URL=http://localhost:8080/uddi/subscription

This tutorial uses Systinet's WASP UDDI 4.5.2 as a standalone server with embedded HSQL installed in a Windows 2000 environment. You can download the SubscriptionDemo.java that is used in this tutorial. For running this demo, you need to set the classpath as shown below.

```
set classpath=C:\waspuddi452\lib\wasp.jar;C:\waspuddi452\
    dist\uddiclient.jar;
```

Start the WASP UDDI server by executing %WASP_HOME%/bin/serverstart.bat. For running the sample, open a command prompt and execute the following:

```
C:$\backslash$subscription$\backslash$samples>java
   uddi.v3.subscription.SubscriptionDemo
```

Make sure that config.properties is in the current directory.