OASIS

UDDI Spec TC

Technical Note

Generating a JAX-RPC Client for UDDI 3.0.2

Revision 0.3

Document identifier:

uddi-spec-tc-tn-jax-rpc

This Version:

http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-jax-rpc-20050126.htm

Latest Version:

http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-jax-rpc.htm

Author:

John Colgrave, IBM (colgrave@uk.ibm.com)

Editors:

Matthew Dovey, Oxford (matthew.dovey@oucs.ox.ac.uk)
Mirek Novotny, Systinet (mirek.novotny@systinet.com)

Abstract:

This document describes the additions and changes to the UDDI 3.0.2 WSDL and schemas to enable a Java client to be generated according to the requirements of the JAX-RPC 1.1 Specification.

Status:

This document is updated periodically on no particular schedule.

Committee members should send comments on this technical note to the uddi-spec@lists.oasis-open.org list. Others should subscribe to and send comments to the uddi-spec-comment@lists.oasis-open.org list. To subscribe, send an email message to uddi-spec-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any intellectual property claims have been disclosed that may be essential to implementing this technical note, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the UDDI Spec TC web page (http://www.oasis-open.org/committees/uddi-spec/).

Table of Contents

1 Introduction

This technical note describes how to generate a Java client for UDDI 3.0.2 using only the mandatory mappings from WSDL and XML Schema to Java in the JAX-RPC 1.1 Specification.

1.1 Problem statement

There are several incompatible Java clients for UDDI V2 which prevents portability of UDDI applications and tools written in Java. This Technical Note aims to avoid a repetition of this for UDDI V3 by encouraging the use of a single JAX-RPC programming model for UDDI V3.

1.2 Terminology

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in **[RFC2119]**.

1.3 Intended Audience

This Technical Note is intended for anyone who intends to produce a Java client for UDDI 3.0.2.

2 Technical note Solution

2.1 Definitions

The reader is assumed to be familiar with WSDL, XML Schema and the JAX-RPC 1.1 specification. No new definitions are introduced with this Technical Note.

2.2 Technical note behavior

The approach adopted is to make the minimum number of changes necessary to the UDDI WSDL and schemas to allow a Java client to be generated with the minimum number of occurrences of SOAPElement.

An important aspect is the mapping of XML namespaces to Java packages. Supporting portable Java applications requires that the same Java packages are used for the various XML namespaces. All of the code derived from WSDL is placed into a single package, org.uddi.v3.wsdl. Each XML Schema namespace is mapped to a different Java package. The following table shows the mappings used:

XML Schema Namespace	Java Package Name
http://www.w3.org/2000/09/xmldsig#	org.w3.schema.xmldsig
urn:uddi-org:api_v3	org.uddi.v3.schema.api
urn:uddi-org:custody_v3	org.uddi.v3.schema.custody
urn:uddi-org:sub_v3	org.uddi.v3.schema.sub

2.2.1 Style of Client

The JAX-RPC specification describes three types of client:

1. A client using generated stubs.
2. A client using a dynamic proxy.
3. A client using a Dynamic Invocation Interface (DII).

This Technical Note describes a client using generated stubs. As modified schemas are required, it is anticipated that UDDI client vendors will generate code from the modified schemas and package the generated code into a client for a particular runtime. This type of client is the most similar to the existing V2 Java clients. Note that the programming model does not depend on the runtime of either the client or the UDDI registry. All that is required is that the tool that generates the client code, and the associated runtime, conform to the JAX-RPC 1.1 specification.

Support for the dynamic proxy type of client is possible but requires that the WSDL and modified schemas be made available to every client.

Support for the DII type of client is also possible but also requires that the WSDL and modified schemas be made available, or a very low-level programming model is used in which the structure of the requests and responses is encoded in the application code.

There are two types of environment in which a JAX-RPC client can execute:

1. A J2SE environment.
2. A J2EE environment (a J2EE client container or one of several J2EE server containers).

This Technical Note describes the use of the J2SE environment. The main difference between the two environments is how the service is obtained. In the J2SE environment the

application uses the ServiceFactory to create an instance of the generated service. In the J2EE environment, the application uses a JNDI lookup to obtain an instance of the service.

2.2.2 New WSDL

2.2.2.1 Service WSDL

JAX-RPC tools require a WSDL service file as input so that must be created as that is not part of the standard WSDL description of UDDI. Only some of the portTypes defined for UDDI 3.0.2 are relevant, so the WSDL service consists of ports for only the following bindings:

1. UDDI_Inquiry_SoapBinding
2. UDDI_Publication_SoapBinding
3. UDDI_Security_SoapBinding
4. UDDI_CustodyTransfer_SoapBinding
5. UDDI_Subscription_SoapBinding

The service WSDL is shown in figure 1.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"

             xmlns:uddi_api_v3_binding="urn:uddi-org:api_v3_binding"
             xmlns:uddi_custody_v3_binding="urn:uddi-org:custody_v3_binding"
             xmlns:uddi_sub_v3_binding="urn:uddi-org:sub_v3_binding"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             targetNamespace="urn:uddi-org:v3_service">

    <import namespace="urn:uddi-org:api_v3_binding"
            location="uddi_api_v3_binding.wsdl"/>
    <import namespace="urn:uddi-org:custody_v3_binding"
            location="uddi_custody_v3_binding.wsdl"/>
    <import namespace="urn:uddi-org:sub_v3_binding"
            location="uddi_sub_v3_binding.wsdl"/>

    <service name="UDDI_Service">


        <port name="UDDI_Inquiry_Port"
              binding="uddi_api_v3_binding:UDDI_Inquiry_SoapBinding">
            <soap:address location="http://localhost/uddi/inquire/"/>
        </port>


        <port name="UDDI_Publication_Port"
              binding="uddi_api_v3_binding:UDDI_Publication_SoapBinding">
            <soap:address location="http://localhost/uddi/publish/"/>
        </port>


        <port name="UDDI_Security_Port"
              binding="uddi_api_v3_binding:UDDI_Security_SoapBinding">
            <soap:address location="http://localhost/uddi/security/"/>
        </port>


        <port name="UDDI_Custody_Port"
          binding="uddi_custody_v3_binding:UDDI_CustodyTransfer_SoapBinding">
            <soap:address location="http://localhost/uddi/custody/"/>
        </port>


        <port name="UDDI_Subscription_Port"
              binding="uddi_sub_v3_binding:UDDI_Subscription_SoapBinding">
            <soap:address location="http://localhost/uddi/subscription/"/>
        </port>


    </service>

</definitions>
```


2.2.3 Changes to Schemas

Changes to the following schemas were made:

1. uddi_v3.xsd
2. uddi_v3custody.xsd
3. uddi_v3subscription.xsd

The changes can be summarized as:

1. Removing the use of xsd:choice
2. Removing the use of the final attribute

These are the only XML Schema constructs used for which support in JAX-RPC 1.1 is described as Optional in the JAX-RPC 1.1 Specification. In practice this means that JAX-RPC tools will either not generate code for these constructs or will generate non-standard code.

The details of the changes to each schema file are in the following sections, with the exception of the removal of the use of the final attribute as that was pervasive.

2.2.3.1 Changes to uddi_v3.xsd

The use of xsd:choice was removed from the bindingTemplate complexType.

This was the original definition:

```
<xsd:complexType name="bindingTemplate" final="restriction">
    <xsd:sequence>
        <xsd:element ref="uddi:description"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:choice>
            <xsd:element ref="uddi:accessPoint"/>
            <xsd:element ref="uddi:hostingRedirector"/>
        </xsd:choice>
        <xsd:element ref="uddi:tModelInstanceDetails" minOccurs="0"/>
        <xsd:element ref="uddi:categoryBag" minOccurs="0"/>
        <xsd:element ref="dsig:Signature"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="bindingKey" type="uddi:bindingKey" use="optional"/>
    <xsd:attribute name="serviceKey" type="uddi:serviceKey" use="optional"/>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="bindingTemplate">
    <xsd:sequence>
        <xsd:element ref="uddi:description"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:accessPoint" minOccurs="0"/>
        <xsd:element ref="uddi:hostingRedirector" minOccurs="0"/>
        <xsd:element ref="uddi:tModelInstanceDetails" minOccurs="0"/>
        <xsd:element ref="uddi:categoryBag" minOccurs="0"/>
        <xsd:element ref="dsig:Signature"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="bindingKey" type="uddi:bindingKey" use="optional"/>
    <xsd:attribute name="serviceKey" type="uddi:serviceKey" use="optional"/>
</xsd:complexType>
```

The use of xsd:choice was removed from the categoryBag complexType.

This was the original definition:

```
<xsd:complexType name="categoryBag" final="restriction">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="uddi:keyedReference" maxOccurs="unbounded"/>
            <xsd:element ref="uddi:keyedReferenceGroup"
                         minOccurs="0"
                         maxOccurs="unbounded"/>
```



```
        </xsd:sequence>
        <xsd:element ref="uddi:keyedReferenceGroup" maxOccurs="unbounded"/>
    </xsd:choice>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="categoryBag">
    <xsd:sequence>
        <xsd:element ref="uddi:keyedReference"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:keyedReferenceGroup"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

The use of xsd:choice was removed from the instanceDetails complexType.

This was the original definition:

```
<xsd:complexType name="instanceDetails" final="restriction">
    <xsd:sequence>
        <xsd:element ref="uddi:description"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:choice>
            <xsd:sequence>
                <xsd:element ref="uddi:overviewDoc" maxOccurs="unbounded"/>
                <xsd:element ref="uddi:instanceParms" minOccurs="0"/>
            </xsd:sequence>
            <xsd:element ref="uddi:instanceParms"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="instanceDetails">
    <xsd:sequence>
        <xsd:element ref="uddi:description"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:overviewDoc"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:instanceParms" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

The use of xsd:choice was removed from the keysOwned complexType.

This was the original definition:

```
<xsd:complexType name="keysOwned" final="restriction">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element ref="uddi:fromKey"/>
            <xsd:element ref="uddi:toKey" minOccurs="0"/>
        </xsd:sequence>
        <xsd:element ref="uddi:toKey"/>
    </xsd:choice>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="keysOwned">
    <xsd:sequence>
        <xsd:element ref="uddi:fromKey" minOccurs="0"/>
        <xsd:element ref="uddi:toKey" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

The use of xsd:choice was removed from the overviewDoc complexType.

This was the original definition:

```
<xsd:complexType name="overviewDoc" final="restriction">
    <xsd:choice>
        <xsd:sequence>
```



```
            <xsd:element ref="uddi:description" maxOccurs="unbounded"/>
            <xsd:element ref="uddi:overviewURL" minOccurs="0"/>
        </xsd:sequence>
        <xsd:element ref="uddi:overviewURL"/>
    </xsd:choice>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="overviewDoc">
    <xsd:sequence>
        <xsd:element ref="uddi:description"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:overviewURL" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

The use of xsd:choice was removed from the find_relatedBusinesses complexType.

This was the original definition:

```
<xsd:complexType name="find_relatedBusinesses" final="restriction">
    <xsd:sequence>
        <xsd:element ref="uddi:authInfo" minOccurs="0"/>
        <xsd:element ref="uddi:findQualifiers" minOccurs="0"/>
        <xsd:choice>
            <xsd:element ref="uddi:businessKey"/>
            <xsd:element ref="uddi:fromKey"/>
            <xsd:element ref="uddi:toKey"/>
        </xsd:choice>
        <xsd:element ref="uddi:keyedReference" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="maxRows" type="xsd:int" use="optional"/>
    <xsd:attribute name="listHead" type="xsd:int" use="optional"/>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="find_relatedBusinesses">
    <xsd:sequence>
        <xsd:element ref="uddi:authInfo" minOccurs="0"/>
        <xsd:element ref="uddi:findQualifiers" minOccurs="0"/>
        <xsd:element ref="uddi:businessKey" minOccurs="0"/>
        <xsd:element ref="uddi:fromKey" minOccurs="0"/>
        <xsd:element ref="uddi:toKey" minOccurs="0"/>
        <xsd:element ref="uddi:keyedReference" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="maxRows" type="xsd:int" use="optional"/>
    <xsd:attribute name="listHead" type="xsd:int" use="optional"/>
</xsd:complexType>
```

2.2.3.2 Changes to uddi_v3custody.xsd

The use of xsd:choice was removed from the discard_transferToken complexType.

This was the original definition:

```
<xsd:complexType name="discard_transferToken" final="restriction">
    <xsd:sequence>
        <xsd:element ref="uddi:authInfo" minOccurs="0"/>
            <xsd:choice>
                <xsd:element ref="uddi_custody:transferToken"/>
                <xsd:element ref="uddi_custody:keyBag"/>
            </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="discard_transferToken">
    <xsd:sequence>
        <xsd:element ref="uddi:authInfo" minOccurs="0"/>
        <xsd:element ref="uddi_custody:transferToken" minOccurs="0"/>
        <xsd:element ref="uddi_custody:keyBag" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```


2.2.3.3 Changes to uddi_v3subscription.xsd

The attributes elementFormDefault=”qualified” and attributeFormDefault=”qualified” were removed as they are incorrect.

The use of xsd:choice was removed from the keyBag complexType.

This was the original definition:

```
<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>
```

and this is the new definition:

```
<xsd:complexType name="keyBag">
    <xsd:sequence>
        <xsd:element ref="uddi_sub:deleted"/>
        <xsd:element ref="uddi:tModelKey"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:businessKey"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:serviceKey"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element ref="uddi:bindingKey"
                     minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

The use of xsd:choice was removed from the subscriptionResultsList complexType.

This was the original definition:

```
<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:choice minOccurs="0">
            <xsd:element ref="uddi:bindingDetail"/>
            <xsd:element ref="uddi:businessDetail"/>
            <xsd:element ref="uddi:serviceDetail"/>
            <xsd:element ref="uddi:tModelDetail"/>
            <xsd:element ref="uddi:businessList"/>
            <xsd:element ref="uddi:relatedBusinessesList"/>
            <xsd:element ref="uddi:serviceList"/>
            <xsd:element ref="uddi:tModelList"/>
            <xsd:element ref="uddi:assertionStatusReport"/>
        </xsd:choice>
        <xsd:element ref="uddi_sub:keyBag"
                     minOccurs="0"
                     maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="someResultsUnavailable"
                   type="xsd:boolean"
                   use="optional"/>
</xsd:complexType>
```

and this is the new definition:

```
<xsd:complexType name="subscriptionResultsList">
    <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>
```

The use of xsd:choice was removed from the subscriptionFilter complexType.

This was the original definition:

```
<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>
```

and this is the new definition:

```
<xsd:complexType name="subscriptionFilter">
    <xsd:sequence>
        <xsd:element ref="uddi:find_binding" minOccurs="0"/>
        <xsd:element ref="uddi:find_business" minOccurs="0"/>
        <xsd:element ref="uddi:find_relatedBusinesses" minOccurs="0"/>
        <xsd:element ref="uddi:find_service" minOccurs="0"/>
        <xsd:element ref="uddi:find_tModel" minOccurs="0"/>
        <xsd:element ref="uddi:get_bindingDetail" minOccurs="0"/>
        <xsd:element ref="uddi:get_businessDetail" minOccurs="0"/>
        <xsd:element ref="uddi:get_serviceDetail" minOccurs="0"/>
        <xsd:element ref="uddi:get_tModelDetail" minOccurs="0"/>
        <xsd:element ref="uddi:get_assertionStatusReport" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

2.2.4 Generated Code

This section describes the code generated from the WSDL and a representative sample of code generated from the schemas. The code has been edited to make it easier to read and so is unlikely to match exactly the code produced by any particular JAX-RPC 1.1 tool but it should be functionally equivalent.

2.2.4.1 Code Generated from WSDL

2.2.4.1.1 Generated Service Interface

This is the Java interface generated from the wsdl:service definition:


```
package org.uddi.v3.wsdl;

import javax.xml.rpc.Service;
import javax.xml.rpc.ServiceException;

public interface UDDI_Service extends Service
{
    UDDI_Security_PortType getUDDI_Security_Port() throws ServiceException;

    UDDI_CustodyTransfer_PortType getUDDI_Custody_Port() throws
ServiceException;

    UDDI_Inquiry_PortType getUDDI_Inquiry_Port() throws ServiceException;

    UDDI_Publication_PortType getUDDI_Publication_Port() throws
ServiceException;

    UDDI_Subscription_PortType getUDDI_Subscription_Port() throws
ServiceException;
}
```

2.2.4.1.2 Inquiry portType Interface

This is the Java interface generated from the inquiry portType:

```
package org.uddi.v3.wsdl;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.uddi.v3.schema.api.*;

public interface UDDI_Inquiry_PortType extends Remote
{
    BindingDetail find_binding(Find_binding body) throws RemoteException,
DispositionReport;

    BusinessList find_business(Find_business body) throws RemoteException,
DispositionReport;

    RelatedBusinessesList find_relatedBusinesses(Find_relatedBusinesses body)
throws RemoteException, DispositionReport;

    ServiceList find_service(Find_service body) throws RemoteException,
DispositionReport;

    TModelList find_tModel(Find_tModel body) throws RemoteException,
DispositionReport;

    BindingDetail get_bindingDetail(Get_bindingDetail body) throws
RemoteException, DispositionReport;

    BusinessDetail get_businessDetail(Get_businessDetail body) throws
RemoteException, DispositionReport;

    OperationalInfos get_operationalInfo(Get_operationalInfo body) throws
RemoteException, DispositionReport;

    ServiceDetail get_serviceDetail(Get_serviceDetail body) throws
RemoteException, DispositionReport;

    TModelDetail get_tModelDetail(Get_tModelDetail body) throws
RemoteException, DispositionReport;
}
```

2.2.4.1.3 Publication portType Interface

This is the Java interface generated from the publication portType:

```
package org.uddi.v3.wsdl;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.uddi.v3.schema.api.*;

public interface UDDI_Publication_PortType extends Remote
{
    void add_publisherAssertions(Add_publisherAssertions body) throws
RemoteException, DispositionReport;

    void delete_binding(Delete_binding body) throws RemoteException,
DispositionReport;

    void delete_business(Delete_business body) throws RemoteException,
DispositionReport;

    void delete_publisherAssertions(Delete_publisherAssertions body) throws
RemoteException, DispositionReport;

    void delete_service(Delete_service body) throws RemoteException,
DispositionReport;

    void delete_tModel(Delete_tModel body) throws RemoteException,
DispositionReport;

    AssertionStatusReport get_assertionStatusReport(Get_assertionStatusReport
body) throws RemoteException, DispositionReport;

    PublisherAssertions get_publisherAssertions(Get_publisherAssertions body)
throws RemoteException, DispositionReport;

    RegisteredInfo get_registeredInfo(Get_registeredInfo body) throws
RemoteException, DispositionReport;

    BindingDetail save_binding(Save_binding body) throws RemoteException,
DispositionReport;

    BusinessDetail save_business(Save_business body) throws RemoteException,
DispositionReport;

    ServiceDetail save_service(Save_service body) throws RemoteException,
DispositionReport;

    TModelDetail save_tModel(Save_tModel body) throws RemoteException,
DispositionReport;

    PublisherAssertions set_publisherAssertions(Set_publisherAssertions body)
throws RemoteException, DispositionReport;
}
```

2.2.4.1.4 Security portType Interface

This is the Java interface generated from the security portType:

```
package org.uddi.v3.wsdl;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.uddi.v3.schema.api.*;

public interface UDDI_Security_PortType extends Remote
{
    void discard_authToken(Discard_authToken body) throws RemoteException,
DispositionReport;

    AuthToken get_authToken(Get_authToken body) throws RemoteException,
DispositionReport;
}
```


2.2.4.1.5 Custody Transfer portType Interface

This is the Java interface generated from the custody transfer portType:

```
package org.uddi.v3.wsdl;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.uddi.v3.schema.api.DispositionReport;
import org.uddi.v3.schema.custody.*;

public interface UDDI_CustodyTransfer_PortType extends Remote
{
    void discard_transferToken(Discard_transferToken body) throws
RemoteException, DispositionReport;

    TransferToken get_transferToken(Get_transferToken body) throws
RemoteException, DispositionReport;

    void transfer_custody(Transfer_custody body) throws RemoteException,
DispositionReport;

    void transfer_entities(Transfer_entities body) throws RemoteException,
DispositionReport;
}
```

2.2.4.1.6 Subscription portType Interface

This is the Java interface generated from the subscription portType:

```
package org.uddi.v3.wsdl;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.uddi.v3.schema.api.DispositionReport;

import org.uddi.v3.schema.sub.*;

public interface UDDI_Subscription_PortType extends Remote
{
    void delete_subscription(Delete_subscription body) throws
RemoteException, DispositionReport;

    SubscriptionResultsList get_subscriptionResults(Get_subscriptionResults
body) throws RemoteException, DispositionReport;

    Subscriptions get_subscriptions(Get_subscriptions body) throws
RemoteException, DispositionReport;

    Subscriptions save_subscription(Save_subscription body) throws
RemoteException, DispositionReport;
}
```

2.2.4.2 Code Generated from Schema

This section contains one example of each type of class generated from the UDDI schemas.

The implementation details have been removed, only the aspects relevant to the programming model are shown.

2.2.4.2.1 Request

This is the Java class corresponding to the find_business request:

```
package org.uddi.v3.schema.api;

public class Find_business
{
    public Find_business() {…}
```



```
    public Integer getMaxRows() {…}

    public void setMaxRows(Integer) {…}

    public Integer getListHead() {…}

    public void setListHead(Integer) {…}

    public String getAuthInfo() {…}

    public void setAuthInfo(String) {…}

    public FindQualifiers getFindQualifiers() {…}

    public void setFindQualifiers(FindQualifiers) {…}

    public Name[] getName() {…}

    public void setName(Name[]) {…}

    public IdentifierBag getIdentifierBag() {…}

    public void setIdentifierBag(IdentifierBag) {…}

    public CategoryBag getCategoryBag() {…}

    public void setCategoryBag(CategoryBag) {…}

    public TModelBag getTModelBag() {…}

    public void setTModelBag(TModelBag) {…}

    public Find_tModel getFind_tModel() {…}

    public void setFind_tModel(Find_tModel) {…}

    public DiscoveryURLs getDiscoveryURLs() {…}

    public void setDiscoveryURLs(DiscoveryURLs) {…}

    public Find_relatedBusinesses getFind_relatedBusinesses() {…}

    public void setFind_relatedBusinesses(Find_relatedBusinesses) {…}
}
```

2.2.4.2.2 Response

This is the Java class corresponding to the businessList response:

```
package org.uddi.v3.schema.api;

public class BusinessList
{
    public BusinessList() {…}

    public Boolean getTruncated() {…}

    public void setTruncated(Boolean) {…}

    public ListDescription getListDescription() {…}

    public void setListDescription(ListDescription) {…}

    public BusinessInfos getBusinessInfos() {…}

    public void setBusinessInfos(BusinessInfos) {…}

}
```

2.2.4.2.3 DispositionReport

This is the Java class corresponding to dispositionReport, which is now used only to return fault information:


```
package org.uddi.v3.schema.api;

public class DispositionReport extends Exception
{
    public DispositionReport(Boolean, Result[]) {…}

    public Boolean getTruncated() {…}

    public Result[] getResult() {…}
}
```

2.2.4.2.4 String with Attribute

This is the Java class corresponding to accessPoint:

```
package org.uddi.v3.schema.api;

public class AccessPoint
{
    public AccessPoint() {…}

    public String getUseType() {…}

    public void setUseType(String) {…}

    public String get_value() {…}

    public void set_value(String) {…}
}
```

2.2.4.2.5 Enumeration

This is the Java class corresponding to completionStatus:

```
package org.uddi.v3.schema.api;

public class CompletionStatus
{
    protected CompletionStatus(String value) {…}

    public static final String _value1 = "status:complete";
    public static final String _value2 = "status:fromKey_incomplete";
    public static final String _value3 = "status:toKey_incomplete";

    public static final CompletionStatus value1 = new
CompletionStatus(_value1);
    public static final CompletionStatus value2 = new
CompletionStatus(_value2);
    public static final CompletionStatus value3 = new
CompletionStatus(_value3);

    public java.lang.String getValue(){…}

    public int hashCode() {…}

    // The following are not required by the spec. but are in the examples

    public static CompletionStatus fromValue(String value)
          throws IllegalArgumentException {…}

    public static CompletionStatus fromString(String value)
          throws IllegalArgumentException {…}

    public String toString(){…}

    public boolean equals(Object obj) {…}
}
```

Note that because the enumeration constants cannot be mapped to Java identifiers that value1, value2 and value3 have had to be used.

2.3 Discussion

2.3.1 Assumptions

It is assumed that no outbound schema validation is required on the client as the generated code does not enforce checks such as maximum string length, or perform whitespace collapsing etc. Also, the removal of xsd:choice means that it is possible for developers to construct requests that will be rejected by the (assumed) schema validation that is performed in the server.

2.3.2 Other Comments

2.3.2.1 Default Endpoint

The generated service interface only supports the getPort methods that do not take parameters, and which return a stub preconfigured for the default endpoint. The service WSDL defined here contains dummy locations/endpoints but UDDI registry providers could choose to provide a customized version of the service WSDL with the correct endpoints specified and generate code from that customized WSDL, in which case the default configuration would (presumably) be correct.

There is a standard property that can be set to change the endpoint.

2.3.2.2 Enumeration Values

A better mapping to Java of completionStatus would be obtained if the enumeration constants were simple strings that did not include the ':' character, for example "complete" rather than "status:complete".

2.4 Example

This section contains a simple example that shows how to make a find_business request, in a J2SE environment, and access the response. The code of the main method is as follows:

```
public static void main(String[] args)
{
    try
    {
        UDDI_Inquiry_PortType inquiryPort = getInquiryPort(args[0]);
        if (inquiryPort != null)
        {
            Find_business findBusinessRequest = buildFindBusinessRequest();
            if (findBusinessRequest != null)
            {
                try
                {
                    BusinessList businessList =
                              inquiryPort.find_business(findBusinessRequest);
                    if (businessList != null)
                    {
                        printBusinessList(businessList);
                    }
                }
                catch (RemoteException re)
                {
                    re.printStackTrace(System.err);
                }
                catch (DispositionReport dr)
                {
                    dr.printStackTrace(System.err);
                }
            }
        }
    }
    catch (ServiceException serviceException)
    {
        serviceException.printStackTrace(System.err);
```



```
    }
}
```

The first thing the method does is to call the getInquiryPort method, which isolates the remaining code from the details of how to obtain the appropriate stub, whether in the J2SE or J2EE environments etc. This particular implementation of the method shows how to obtain a pre-generated stub in the J2SE environment:

```
private static UDDI_Inquiry_PortType getInquiryPort(String inquiryURL) throws
ServiceException
{
    UDDI_Inquiry_PortType inquiryPort = null;
    ServiceFactory serviceFactory = ServiceFactory.newInstance();
    if (serviceFactory != null)
    {
        Service service = serviceFactory.loadService(UDDI_Service.class);
        if (service != null)
        {
            UDDI_Service uddiService = (UDDI_Service)service;
            inquiryPort = uddiService.getUDDI_Inquiry_Port();
            Stub genericInquiryStub = (Stub)(inquiryPort);
            genericInquiryStub._setProperty(Stub.ENDPOINT_ADDRESS_PROPERTY,
                                            inquiryURL);
        }
    }
    return inquiryPort;
}
```

Note that a single argument is passed in which is the inquiry URL to be used. If stubs particular to a target registry were produced, this would be unnecessary as the default URL would presumably be set to the appropriate value.

Having obtained an instance of the inquiry port, the application then calls a method to construct a find_business request:

```
private static Find_business buildFindBusinessRequest()
{
    Find_business findBusinessRequest = new Find_business();
    Integer maxRows = new Integer(3);
    findBusinessRequest.setMaxRows(maxRows);
    String[] findQualifierArray = new String[1];
    findQualifierArray[0] = "approximateMatch";
    FindQualifiers findQualifiers = new FindQualifiers();
    findQualifiers.setFindQualifier(findQualifierArray);
    findBusinessRequest.setFindQualifiers(findQualifiers);
    Name name = new Name();
    name.set_value("I%");
    Name[] names = new Name[1];
    names[0] = name;
    findBusinessRequest.setName(names);
    return findBusinessRequest;
}
```

The request that is returned is equivalent to the following:

```
<find_business maxRows="3" xmlns="urn:uddi-org:api_v3">
    <findQualifiers>
        <findQualifier>approximateMatch</findQualifier>
    </findQualifiers>
    <name>I%</name>
</find_business>
```

The find_business method is then called, with the request as the single parameter. The remainder of the code shows how to access the results of the call:

```
private static void printBusinessList(BusinessList businessList)
{
    if (businessList != null)
    {
        Boolean truncated = businessList.getTruncated();
        if (truncated != null)
        {
            System.out.println("truncated is " + truncated);
        }
```



```
        ListDescription listDescription = businessList.getListDescription();
        if (listDescription != null)
        {
            System.out.println("Have got listDescription:");
            System.out.println("includeCount is " +
                               listDescription.getIncludeCount());
            System.out.println("actualCount is " +
                               listDescription.getActualCount());
            System.out.println("listHead is " +
                               listDescription.getListHead());
        }
        BusinessInfos businessInfos = businessList.getBusinessInfos();
        if (businessInfos != null)
        {
            printBusinessInfos(businessInfos);
        }
    }
}

private static void printBusinessInfos(BusinessInfos businessInfos)
{
    if (businessInfos != null)
    {
        BusinessInfo[] businessInfoArray = businessInfos.getBusinessInfo();
        if (businessInfoArray != null)
        {
            int businessInfoArraySize = businessInfoArray.length;
            if (businessInfoArraySize > 0)
            {
                for (int i = 0; i < businessInfoArraySize; i++)
                {
                    printBusinessInfo(businessInfoArray[i]);
                }
            }
        }
    }
}

private static void printBusinessInfo(BusinessInfo businessInfo)
{
    if (businessInfo != null)
    {
        System.out.println("businessInfo:");
        URI businessKey = businessInfo.getBusinessKey();
        if (businessKey != null)
        {
            System.out.println("\tbusinessKey is " + businessKey);
        }
        Name[] names = businessInfo.getName();
        if (names != null)
        {
            printNames("\t", names);
        }
        Description[] descriptions = businessInfo.getDescription();
        if (descriptions != null)
        {
            printDescriptions("\t", descriptions);
        }
        ServiceInfos serviceInfos = businessInfo.getServiceInfos();
        if (serviceInfos != null)
        {
            printServiceInfos(serviceInfos);
        }
    }
}

private static void printNames(String indent, Name[] names)
{
    if (names != null)
    {
        int size = names.length;
        if (size > 0)
        {
            for (int i = 0; i < size; i++)
            {
```



```
                printName(indent, names[i]);
            }
        }
    }
}

private static void printName(String indent, Name name)
{
    if (name != null)
    {
        System.out.println(indent + "name:");
        String lang = name.getLang();
        if (lang != null)
        {
            System.out.println(indent + "\tlang is " + lang);
        }
        String value = name.get_value();
        if (value != null)
        {
            System.out.println(indent + "\tvalue is " + value);
        }
    }
}

private static void printDescriptions(String indent, Description[]
descriptions)
{
    if (descriptions != null)
    {
        int size = descriptions.length;
        if (size > 0)
        {
            for (int i = 0; i < size; i++)
            {
                printDescription(indent, descriptions[i]);
            }
        }
    }
}

private static void printDescription(String indent, Description description)
{
    if (description != null)
    {
        System.out.println(indent + "description:");
        String lang = description.getLang();
        if (lang != null)
        {
            System.out.println(indent + "\tlang is " + lang);
        }
        String value = description.get_value();
        if (value != null)
        {
            System.out.println(indent + "\tvalue is " + value);
        }
    }
}

private static void printServiceInfos(ServiceInfos serviceInfos)
{
    if (serviceInfos != null)
    {
        ServiceInfo[] serviceInfoArray = serviceInfos.getServiceInfo();
        if (serviceInfoArray != null)
        {
            int serviceInfoArraySize = serviceInfoArray.length;
            if (serviceInfoArraySize > 0)
            {
                for (int i = 0; i < serviceInfoArraySize; i++)
                {
                    printServiceInfo(serviceInfoArray[i]);
                }
            }
        }
    }
```



```
}

private static void printServiceInfo(ServiceInfo serviceInfo)
{
    if (serviceInfo != null)
    {
        System.out.println("\tserviceInfo:");
        URI serviceKey = serviceInfo.getServiceKey();
        if (serviceKey != null)
        {
            System.out.println("\t\tserviceKey is " + serviceKey);
        }
        URI businessKey = serviceInfo.getBusinessKey();
        if (businessKey != null)
        {
            System.out.println("\t\tbusinessKey is " + businessKey);
        }
        Name[] names = serviceInfo.getName();
        if (names != null)
        {
            printNames("\t\t", names);
        }
    }
}
```


3 References

3.1 Normative

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

Appendix A. Acknowledgments

The following individuals were members of the committee during the development of this technical note:

- Andrew Hately

Ian Hodges of IBM also provided helpful comments on an earlier draft of this Technical Note.

Appendix B. Revision History

Rev	Date	By Whom	What
0.1	July 16th 2004	John Colgrave	Initial draft for TC discussion.
0.2	August 11th 2004	John Colgrave	Some changes removed following the approval of CR-078.
0.3	December 7th 2004	John Colgrave	Updated following TC review.

Appendix C. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.