



IBM Software Group

IBM® WebSphere® Application Server V6

Java™ API for XML Registries (JAXR)



@business on demand.

© 2004 IBM Corporation
Updated January 25, 2005

Goals

- Explain the purpose of JAXR
- Explore the JAXR specification
- Compare other Universal Description, Discovery and Integration (UDDI) APIs with JAXR
- Describe how to use the JAXR API

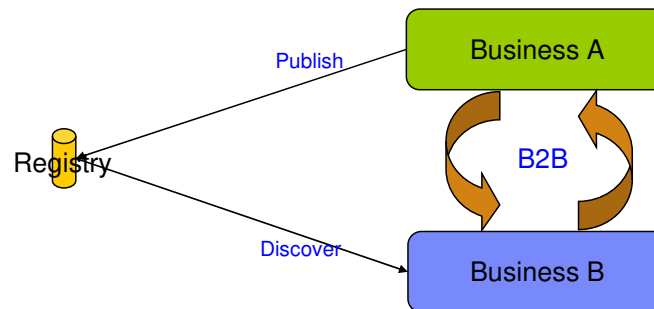
Agenda

- Discuss Registries
- Explain the concept of JAXR
- JAXR in more detail
 - ▶ JAXR capability profiles
 - ▶ JAXR architecture
 - ▶ The JAXR information model
- UDDI4J and the IBM Java Client for UDDI v3
- Using the JAXR API
 - ▶ Connecting to a Registry
 - ▶ Managing Registry objects
 - ▶ Searching a Registry
 - ▶ JAXR security

What is a Registry?

What is a Registry?

- An infrastructure that enables the publishing and discovery of Web Services
- Facilitates business-to-business (B2B) interactions



What is JAXR?

What is JAXR?

- A standard API for accessing Registries within a Java platform
- Provides a union of best features of dominant registry specs (particularly UDDI and ebXML)
 - ▶ Current version (v1.0), supports UDDI v2 only
 - ▶ Does not map precisely to UDDI (For a precise mapping, UDDI provides the IBM UDDI version 3 Client for Java and UDDI4J)
- Developed by the JSR093 expert group and it is part of the J2EE 1.4 specification



The stated goals of the specification are to:

1. Define a general purpose Java API for accessing business registries that allows any JAXR client to access and interoperate with any business registry that is accessible via a JAXR provider.
2. Define a pluggable provider architecture that enables support for diverse registry specifications and standards.
3. Support a union of the best features of dominant registry specifications rather than a common intersection of features. *JAXR is not a least common denominator API.*
4. Ensure support for dominant registry specifications such as ebXML and UDDI, while maintaining sufficient generality to support other types of registries, current or future.
547
5. Ensure synergy with other Java specifications related to XML.

Important Note: the current JAXR specification is written against UDDI v2.

Currently there are a variety of specifications for XML registries. These include:

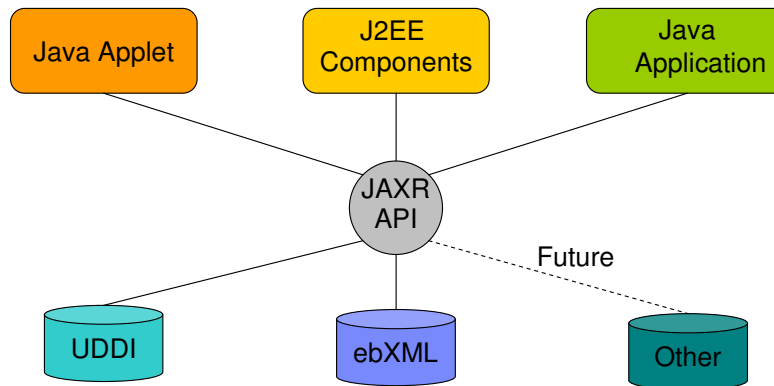
The ebXML Registry and Repository standard, which is sponsored by the Organization for the Advancement of Structured Information Standards (OASIS) and the United Nations Centre for the Facilitation of Procedures and Practices in Administration, Commerce and Transport (U.N./CEFACT)

The Universal Description, Discovery, and Integration (UDDI) project, which is being developed by a vendor consortium

eCo Framework, developed as part of the eCo Framework Project. This was chartered by CommerceNet in August, 1998, with Commerce One as the primary corporate sponsor,

What is JAXR? (cont.)

- The goal of JAXR is to enable interoperability between diverse clients and Registries



Although the goal of JAXR is to provide a generic API for any XML based registry, the 1.0 specification maps primarily to ebXML and UDDI. The ability to provide JAXR APIs for "Other" registries should improve as the specification develops

JAXR in More Detail

JAXR Capability Profiles

- JAXR API methods are categorized by a Capability Profile
- Two Capability Profiles currently defined
 - ▶ Level 0
 - ▶ Level 1
- JAXR provider must declare the capability level for its implementation
- Level 1 support includes Level 0 support
- IBM's JAXR implementation is Level 0 compliant
 - ▶ Level 0 support is for UDDI Registries



The JAXR API categorizes its API methods by a small number of capability profiles. Currently only two capability profiles are defined (level 0 and level 1). The capability level is defined in the API documentation for each method in a class or interface in the JAXR API.

There is no assignment of capability level to interfaces and classes in the JAXR API. Capability assignment is done at the method level only.

A JAXR provider must declare the capability level for its implementation of the JAXR API. A JAXR client may discover a JAXR provider's capability level by invoking methods on an interface named `CapabilityProfile` as defined by the JAXR API. If a JAXR provider declares support for a specific capability level then it implicitly declares support for lower capability levels. For example, a JAXR provider that declares support for the level 1 profile implicitly declares support for level 0 profile.

A JAXR provider must implement the functionality described by the JAXR API for each method that is assigned a capability level that is less than or equal to the capability level declared by the JAXR provider.

A JAXR provider must implement all methods that are assigned a capability level that is greater than the capability level declared by the JAXR provider, to throw an `UnsupportedCapabilityException`. A JAXR provider must never implement any other behavior for methods assigned a greater than the capability level declared by the JAXR provider. The reason for this restriction is that it is necessary to ensure portable behavior for JAXR clients for any JAXR provider within a specific capability level.

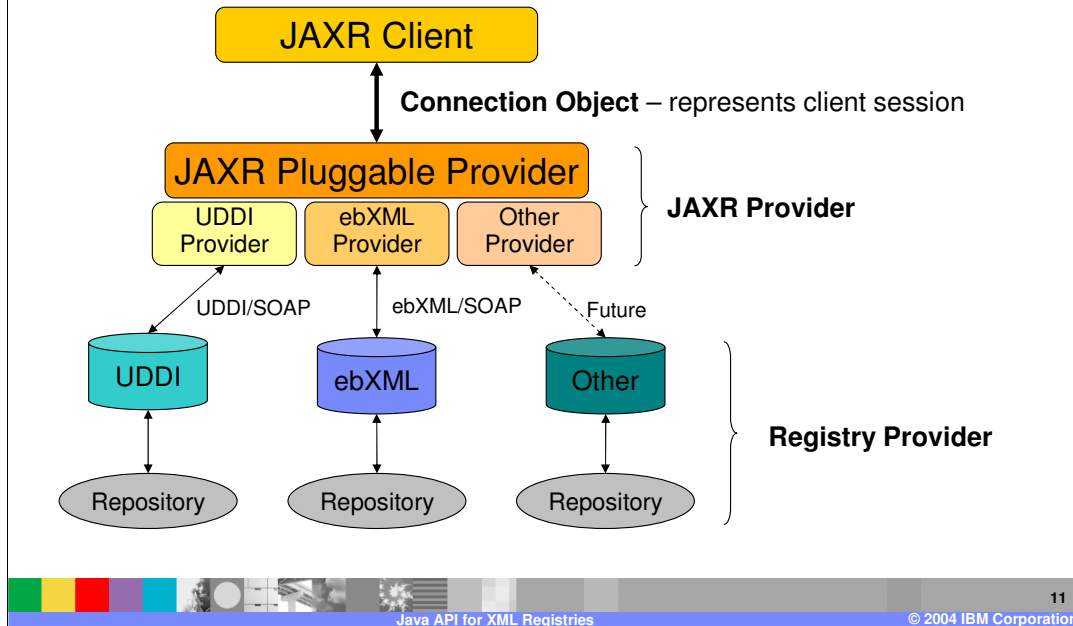
Support for the level 0 profile is required to be supported by all JAXR providers. The methods assigned to this profile provide the most basic registry capabilities. JAXR providers for UDDI must be level 0 compliant.

Support for the level 1 profile is optional for JAXR providers. The methods assigned to this profile provide more advanced registry capabilities that are needed by more demanding JAXR clients. JAXR providers for ebXML must be level 1 compliant.

Examples of Level 1 capability:

WASv6 JAXR Top The JAXR RegistryPackage (used to group logically related Registry Objects together) is Level 1 only Page 10 of 24

JAXR Architecture



The JAXR client may be any stand-alone Java application or an enterprise component based on J2EE technology. The JAXR client uses the JAXR API to access a registry via a JAXR provider.

A Connection object represents a client session with a registry provider using a JAXR provider. It maintains state information for a specific connection. From the Connection, the client obtains the *RegistryService* interface. This in turn provides access to *Capability specific* interfaces, for example, Life cycle management, query management (more details later).

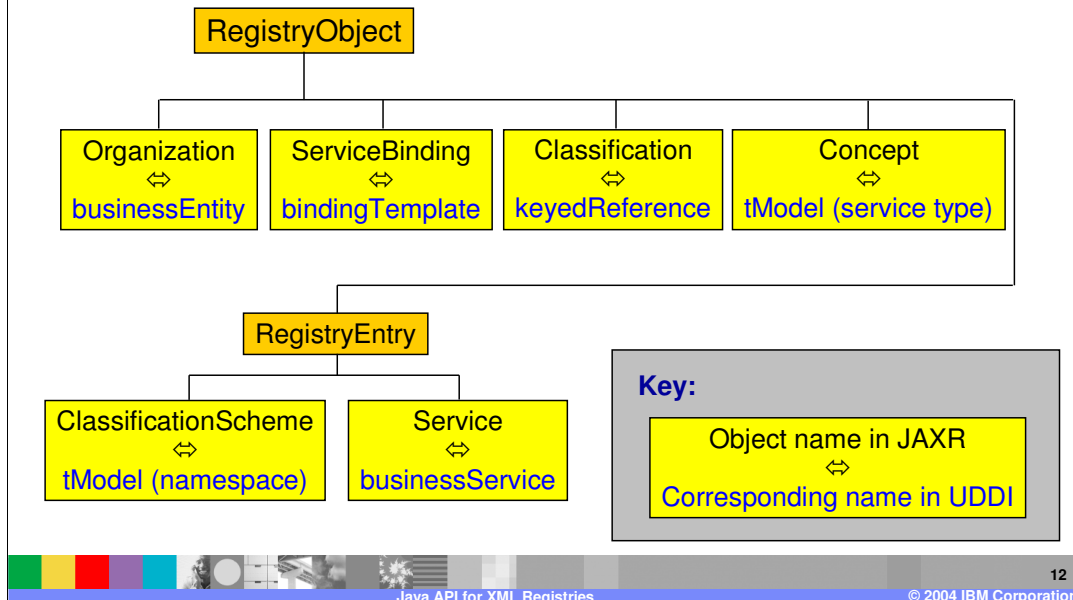
The JAXR Pluggable provider implements features of the JAXR API that are independent of any specific registry type. The Pluggable provider provides a single abstraction for multiple registry-specific JAXR providers. It allows the client to avoid being exposed to the fact that there are multiple registry-specific JAXR providers performing the actual registry access. For example, the JAXR Pluggable provider is provided with a pluggable *ConnectionFactory* implementation that can create JAXR Connections using the appropriate registry-specific JAXR provider.

Registry providers are implementations of various registry specifications, like UDDI, ebXML

The specification makes a distinction between a Registry and a Repository. A Repository is a holder of submitted content while a Registry is a catalog that describes the submitted content in the Repository.

Bear in mind also the note under Slide 8 regarding the capability of the specification to enable JAXR providers for "Other" Registries

The JAXR Information Model



This is a subset of the complete JAXR information model showing some example objects by way of illustration – these are all Java interfaces in the API model.

It also shows examples of how JAXR objects map to UDDI objects.

The Concept object maps to the type of tModel that represents a “fingerprint” of a Service.

The ClassificationScheme object maps to the type of tModel that represents a classification namespace, like NAICS.

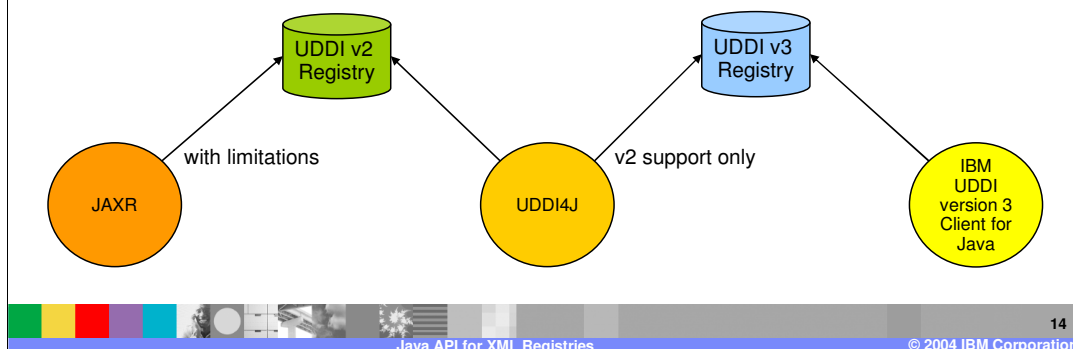
The mapping of JAXR to UDDI is discussed in detail in Appendix D of the JAXR specification (URL on summary slide).

Some interfaces require additional metadata such as version information. These have the RegistryEntry interface as their base interface

IBM UDDI Version 3 Client for Java and UDDI4J

JAXR compared to UDDI Java APIs

- There are some limitations in the mapping of the JAXR Information Model to UDDI
- UDDI4J maps precisely to UDDI v2 and therefore provides a more complete API capability
- IBM UDDI version 3 Client for Java maps precisely to UDDI v3
 - ▶ JAXR is for UDDI v2 only, no mapping to UDDI v3 APIs
- JAXR aims to provide portability across different XML Registries



JAXR is actually closely aligned with ebXML rather than UDDI

Mapping Limitation examples:

- There are two different JAXR objects which map to the tModel object.
- The JAXR PostalAddress object has a lot more structure than the corresponding UDDI address object, so many of its attributes can't be mapped.
- The JAXR RegistryPackage (used to group logically related RegistryObjects together) has no equivalent in UDDI.

Attempts by a client to work with objects/attributes that can't be mapped will result in an `UnsupportedCapabilityException` being thrown.

Portability – as long as the code only uses JAXR elements common to all Registries being accessed.

Furthermore, JAXR only supports access to a v2 UDDI Registry. WebSphere 6.0 will provide the **IBM UDDI version 3 Client for Java** (IJC4U3) which is the v3 equivalent of v2.

Using the JAXR API

Connecting to a Registry

1. Obtain a `ConnectionFactory` object
2. Set properties on the `ConnectionFactory` (example: the Registry URL)
3. Obtain a `Connection` from the `ConnectionFactory` (`createConnection` method)

Connecting to a Registry: Example

```
//specify the ConnectionFactory implementation classname
System.setProperty("javax.xml.registry.ConnectionFactoryClass",
    "com.ibm.xml.registry.uddi.ConnectionFactoryImpl");
//instantiate a ConnectionFactory object
ConnectionFactory connectionFactory = ConnectionFactory.newInstance();
//instantiate a Properties object and add the Inquiry and Publish URLs
Properties props = new Properties();
props.setProperty("javax.xml.registry.queryManagerURL",
    "http://localhost:9080/uddisoap/inquiryapi");
props.setProperty("javax.xml.registry.lifeCycleManagerURL",
    "https://localhost:9080/uddisoap/publishapi");
//add the properties to the ConnectionFactory
factory.setProperties(props);
//create the Connection
Connection connection = factory.createConnection();
```

Managing Registry Objects

- **BusinessLifeCycleManager Interface:**
 - ▶ Defines API similar to UDDI Publisher's API
 - ▶ Provides methods to create, update and delete registry objects
 - ▶ For example:
 - **saveServices**(java.util.Collection services)
 - **deleteOrganizations**(java.util.Collection organizationKeys)
 - ▶ Methods return a **BulkResponse** instance containing:
 - Collection of response objects like keys of successfully saved registry objects
 - Collection of **RegistryExceptions**, if any

Managing Registry Objects: Example

```
//get a RegistryService object
RegistryService rs = connection.getRegistryService();
//get a BusinessLifecycleManager object
BusinessLifecycleManager blcm = rs.getBusinessLifecycleManager();
//create an Organization called "MyBank"
Organization org = blcm.createOrganization("MyBank");
//instantiate a Collection and add the Organization to it
Collection orgs = new ArrayList();
orgs.add(org);
//save the Organization in the registry
BulkResponse response = blcm.saveOrganizations(orgs);
//get the exceptions Collection
Collection exceptions = response.getException();
//check to see if any exceptions have been returned
```

When a client creates an organization, it does not include a key; the registry returns the new key when it accepts the newly created organization. The key(s) of the newly created object(s) is(are) returned in the BulkResponse and can be obtained by calling the `getCollection()` method on the BulkResponse object and then iterating through the Collection.

Searching a Registry

- BusinessQueryManager interface:
 - ▶ Defines API similar to UDDI Inquiry API
 - ▶ Provides find methods
 - ▶ Arguments specify the search criteria:
Example: findQualifiers, namePatterns
 - ▶ BulkResponse is returned



There is also a DeclarativeQueryManager interface, which provides a more flexible search API allowing SQL queries. However, this is at Level 1 and therefore not supported in the WebSphere JAXR provider.

Searching a Registry: Example

```
//get a BusinessQueryManager object
BusinessQueryManager bqm = rs.getBusinessQueryManager();
//instantiate a Collection and add findQualifier(s)
Collection findQualifiers = new ArrayList();
findQualifiers.add(FindQualifier.CASE_SENSITIVE_MATCH);
//instantiate a Collection and add namePattern(s)
Collection namePatterns = new ArrayList();
namePatterns.add("%Bank%");
//initiate the search
BulkResponse response = bqm.findOrganizations(findQualifiers,
    namePatterns, null, null, null, null);
//get the results Collection
Collection orgs = response.getCollection();
```

To search for organizations by name, you normally use a combination of find qualifiers (which affect sorting and pattern matching) and name patterns (which specify the strings to be searched). The `findOrganizations` method takes a collection of `findQualifier` objects as its first argument and a collection of `namePattern` objects as its second argument.

A client can use percent signs (%), as in the above example, to specify that the query string can occur anywhere within the organization name.

JAXR Security

- **Connection.setCredentials method**
 - ▶ Allows the client to set the security credentials for the user that is currently associated with the client:

```
PasswordAuthentication pa = new PasswordAuthentication("MyUser",
new char[] { 'M', 'y', 'p', 'a', 's', 's', 'w', 'o', 'r', 'd' });

Set credentials = new HashSet();

credentials.add(pa);

connection.setCredentials(credentials);
```

The authentication mechanisms that the IBM JAXR implementation supports are authToken ("UDDI_GET_AUTHTOKEN") and HTTP basic authentication ("HTTP_BASIC")

Username/password combination is specified using an instance of the `java.net.PasswordAuthentication` class.

Summary

- JAXR provides a standard Java API for accessing XML Registries
- JAXR provides Registry object management and search capabilities
- IBM UDDI version 3 Client for Java or UDDI4J provide a more complete mapping to UDDI objects
- The JAXR specification is available via the JAXR home page at
<http://java.sun.com/xml/jaxr/index.jsp>

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM (logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)/business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2004. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.