All About UDDI:
Publishing and Finding Web Services

Mark Colan
e-business @evangelist
mcolan@us.ibm.com
http://ibm.com/developerworks/speakers/colan
Agenda

Background: Web services and SOA
  ▶ The role of discovery in a Service Oriented Architecture
  ▶ Review of alternative discovery mechanisms

Overview of UDDI version 1
  ▶ UDDI.org process and participants
  ▶ Architectural introduction

Six species of UDDI
  ▶ Focus on the role of “private” UDDI
  ▶ See how private UDDI aids in usage of Web Services

The future: UDDI version 2 and version 3

Acknowledgement: thanks to Tom Bellwood, Steve Graham, Vince Brunssen, Doug Tidwell, and others for much of this material
Why do we need Web Services?

Not everyone will use the same

- Computer Hardware
- Operating system
- Programming language
- Distributed object system
- Database or other application

...but we still want to integrate everyone's systems.

- internally: Enterprise Application Integration
- externally: Business Process Integration and Automation
- pervasive devices, collaboration, and more
Are we on the same page?

Web Service
Simple Object Access Protocol (SOAP)
Web Service Description Language (WSDL)
Web services
...goals and requirements

Goals of Web services technologies:

- platform- and vendor-neutral
- move from monolithic, custom-coded apps to choreographed, scripted, reconfigurable components
- flexibility, agility, ease of integration
- loosely coupled systems to manage inevitable change
- late binding comes for free

Requirements

- standards are a requirement to make it work smoothly
- tools to speed integration and assembly
- universal platforms to support deployment

Service Oriented Architecture

- the design pattern for Web services discovery
- service description is the key to integration and interoperability
**UDDI Roles and Operations**

**Service Registry**
- provides support for publishing and locating services
- like telephone yellow pages

**Service Provider**
- provides e-business services
- **PUBLISHes** availability of these services through a registry

**Service Requestor**
- **FINDs** required services in the Service Registry
- **BINDs** to services from Service Provider

**UDDI defines publish and find messages ("API")**
- the service provider defines **bind** operations according to application-specific service requirements
Simple XML messages for request and response

- we only care about message format and content, not about the underlying backend implementation
  - this is the key to loose coupling
- we can use protocols other than HTTP
The key to interoperability

How does the Service Requestor know the format of
  ▶ the expected request message(s)?
  ▶ the response message(s)?

By the Service Description (WSDL document)
WSDL: Web Services Description Language

WSDL describes operational information
- where the service is located (service implementation definition)
- what the service does (service interface definition)
**WSDL Speeds Implementation**

WSDL Document

WSDL Speeds Implementation

Authored, or generated (e.g. introspection)

WebSphere Studio Application Developer

WebSphere App Server

App-specific code (you add this)

Client Application Code

Service Proxy

Service Implementation Template

Back-end Processes

Mark Colan, IBM Corporation All About UDDI Page 10
**WSEL: Web Services Endpoint Language**

**Endpoint property descriptions**
- Non-operational characteristics of the Web services deployment environment
- Why I should or should not invoke this service
- Examples:
  - quality of service descriptions
  - security characteristics
  - duration, retry, failure notification

**No specification yet -- this is a work in progress**
- brief description: see WSFL Specification Appendix C
- ibm.com/software/solutions/webservices/pdf/WSFL.pdf
How does the Requestor get the WSDL?

Well, that's the key to the SOA "find" operation

- and the core of this presentation

What are the ways a requestor an get the WSDL?

- let's have a look.
Web Service Discovery Mechanisms

High function

static find

UDDI (private or operator)

WSDL-repository

DISCO/ADS

e-mail, FTP, HTTP GET
Simple / Static Discovery

Simple ways of getting the WSDL document:

- "I'll email you the WSDL"
- "FTP the WSDL from this site"
- "Here's an URL to the document" (or we found it on the provider's web site)
- Sneakernet (WSDL on a floppy)

In all of these cases:

- the requestor get's the required WSDL
- the requestor does not get much runtime choice
- works for very static binding
- doesn't address what to do if the service (especially the service location) changes
Elaboration on the "found it on the web site" theme

**Microsoft DISCO:**

```xml
<?xml version="1.0" ?>
<disco:discovery
 xmlns:disco="http://schemas.xmlsoap.org/disco"
 xmlns:wsdl="http://schemas.xmlsoap.org/disco/wsdl">
<wsdl:contractRef
 ref="http://MyWebServer/UserName.asmx?WSDL"/>
</disco:discovery>
```

**IBM ADS:**

- Similar approach
- Ideal for web crawlers locating Web services

...but DISCO is dead!
WSDL Repository sites:
www.salcentral.com

WSDL / SOAP Web Services Search Engine

The worlds largest brokerage for schemas, reviews and quality assurance information on web services.

Search for a web service by entering simple search criteria i.e. 'calculator', 'email', 'sms' etc.

Search for Web Service

Web Service Users
- Find web services
- Buy access to services
- Watch web services
- Get web service support
- Use dotNET search

Web Service Providers
- Promote web services
- Sell web services
- Test web services
- Support web services
- Use dotNET search

“The Napster of Web Services”
Cool SalCentral Features

- View by company/WSDL
- Latest/most popular WSDL
- Details on SPEC level compliance
- Includes a categorization mechanism
- Analysis/reviews etc.
- Notification when WSDL changes
WSDL Repository sites: xmethods.net

What is this site for?
Emerging standards such as SOAP will enable a new generation of "web services" that allow systems to make remote procedure calls to other systems over the Internet. For example, a corporate inventory management system might publish a service that allows a customer system to check real-time inventory levels. This site lists publicly accessible web services.

Updates
- XMethods now has a UDDI interface built using GLUE. A UDDI browser can be found at SQLData
- Brian Berns has integrated XFS with Windows.
- Sign up to be notified of new services via email

SOAP Service List
(★ New  ● Active  ● Inactive)

<table>
<thead>
<tr>
<th>Owner</th>
<th>Status</th>
<th>Service Name</th>
<th>Description</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>★ Infobiquity</td>
<td>★</td>
<td>NearestTileDealer</td>
<td>Locates Nearest Tile Dealer in USA/Canada</td>
<td>MS .NET</td>
</tr>
<tr>
<td>★ eByz</td>
<td>★</td>
<td>Ampent Lease Calculator</td>
<td>monthly lease payment calculator</td>
<td>Tomcat</td>
</tr>
<tr>
<td>★ Bob Swart</td>
<td>★</td>
<td>Dr Bob's Programming Clinic Headlines</td>
<td>News headlines on six topics.</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ PILLAR</td>
<td>★</td>
<td>Date Time Requestor Service</td>
<td>Date/Time service</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ Bob Swart</td>
<td>★</td>
<td>Number To Words (in Dutch)</td>
<td>Converts Numbers to Dutch words</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ Bob Swart</td>
<td>★</td>
<td>Romulan Numbers XLII</td>
<td>Convert roman numbers &lt;-&gt; digital numbers</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ Bob Swart</td>
<td>★</td>
<td>Tic Tac Toe</td>
<td>The game of Tic-Tac-Toe</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ QualTech IT</td>
<td>★</td>
<td>CEPInfo</td>
<td>Brazilian post office zip code (CEP) info</td>
<td>Delphi</td>
</tr>
<tr>
<td>★ Seanco</td>
<td>★</td>
<td>ElectronicProductsFinder</td>
<td>Electronics product finder.</td>
<td>MS .NET</td>
</tr>
<tr>
<td>★ Seanco</td>
<td>★</td>
<td>VideoGamesFinder</td>
<td>Video game product finder.</td>
<td>MS .NET</td>
</tr>
<tr>
<td>★ Seanco</td>
<td>★</td>
<td>SportingGoodsFinder</td>
<td>Sporting goods product finder.</td>
<td>MS .NET</td>
</tr>
</tbody>
</table>
What is UDDI?

Universal Description, Discovery, and Integration

A project to speed interoperability and adoption for web services

- Standards-based specifications for service description and discovery
- A set of Internet-based implementations
- Partnership among industry and business leaders - more than 310 companies have signed up so far
- 2000+ business entries are there now
- Initiated by IBM, Microsoft, and Ariba

UDDI has two pieces:

- the UDDI Business Registry (hosts the data)
- the API and data model (provides access to the data)
Problems UDDI Solves

Broader B2B
A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols.

Discover Services

Smarter Search
A flower shop in Australia wants to be “plugged in” to every marketplace in the world, but doesn’t know how.

Integrate them Together

Easier Aggregation
A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

Publish for Accessibility

Web Service Visibility
Provide a standards-based profile for all electronic services that are provided. Includes web sites, other electronic resources.
UDDI Vision and Process

Start with existing standards
- TCP/IP, HTTP, XML
- Industry-specific schemas
- Shared vision of open protocols

Augment and implement via a Web Service
- Common web services “stack”
- Shared implementation to avoid confusing customers
- Public specs, open service, inclusive process

Transition to a Standards Body
- Manage design process for 3 revs
- License control and IP to a 3rd party
**UDDI Implementation**

**Manufacturers**

**Flower Shops**

**Marketplaces**

**UDDI Business Registry**

- Programmatic descriptions of businesses and the services they support
- Programmatic descriptions of Web service specifications
- Programming model and schema
- Platform & language neutral
- Uses XML, HTTP, and SOAP
- Free on the Internet
How UDDI version 1 works

1. SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services.

2. Businesses populate the registry with descriptions of the services they support.

3. UBR assigns a programmatically unique identifier to each service and business registration.

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies.

5. Business uses this data to facilitate easier integration with each other over the Web.
**UDDI at Work**

1. Mitch & Co creates online website with local ASP

2. ASP registers MitchCo with UBR

3. Marketplaces and search engines query UBR, cache MitchCo data, and bind to its services

4. Consumers and businesses discover MitchCo and do business with it
Registry Data

Businesses register public information about themselves

Standards bodies, programmers, businesses register information about their service types

White Pages

Yellow Pages

Green Pages

Service Type Registrations ("tModels")
White Pages

Contact Information about a Service Provider

<table>
<thead>
<tr>
<th>Business Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Description</td>
</tr>
<tr>
<td>✅ list of multi-language text strings</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contact info</th>
</tr>
</thead>
<tbody>
<tr>
<td>✅ names, phone numbers, fax numbers, web sites…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Known Identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>✅ list of identifiers that a business may be known by</td>
</tr>
<tr>
<td>✅ DUNS, Thomas, other</td>
</tr>
</tbody>
</table>

Yellow Pages

Green Pages

Service Type Registrations ("tModels")
Yellow Pages

Business categories

3 standard taxonomies in V1
1. Industry: NAICS (Industry codes - US Govt.)
2. Product/Services: UN/SPSC (ECMA)
3. Location: Geographical taxonomy

Implemented as name-value pairs
- allows any valid taxonomy identifier to be attached to the business white page
**Green Pages**

*How to bind to service provider*

Technical info about services, specified by a business: how to "do e-commerce" with them

- references to specifications for Web services
- support for pointers to various file and URL based discovery mechanisms if required

Nested model

- Business processes
- Service descriptions
- Binding information

Programming, platform, implementation agnostic
Service Type Registrations

Pointer to the namespace where service type is described
- What programmers read to understand how to use the service
- e.g. URL to WSDL description

Identifier for who published the service

Identifier for the service type registration
- called a tModelKey
- Used as a signature by web sites that implement those services

White Pages
Yellow Pages
Green Pages
Registry Operation

Peer nodes (websites)
- Companies register with any node
- Registrations replicated on a daily basis
- Complete set of “registered” records available at all nodes

Common set of SOAP APIs supported by all nodes

Compliance enforced by business contract
Current Operators

IBM UDDI Business Registry Sites:
- Official Registry: http://ibm.com/services/uddi
- Test Registry: http://ibm.com/services/uddi/testregistry

Microsoft UDDI Business Registry Sites:
- Official Registry: http://uddi.microsoft.com
- Test Registry: http://test.uddi.microsoft.com

Each supports both SOAP and web page access

Future operators:
- Hewlett-Packard
- SAP
UDDI and SOAP

Create, View, Update, and Delete registrations
UDDI.org Specifications

White Papers:
- Executive White Paper
- Technical White Paper

Specifications:
- Programmer's API (version 1.00 and 2.00)
- Data Structure Specification (version 1.00 and 2.00)
- Replication Specification (version 2.00)
- Operator's Specification (version 2.00)

Best Practices papers:
- Using WSDL in a UDDI Registry
- Providing a Taxonomy for use in UDDI version 2.00
UDDI version 1 data structures: overview

- `<businessEntity>`
  - name, contacts,
  - description, identifiers, categories

- `<businessService>`
  - `sService` (1..n)
  - `<bindingTemplate>` (1..n)
    - technical information

- `<tModel>`
  - name
  - description
  - URL pointers to specifications

- businessEntity
- businessService
- bindingTemplate
- tModel
UDDI version 1 data structures:
businessEntity

- businessKey (UUID)
- name
- authorizedName
- operator
- description(s)

UUID - Universally Unique IDentifier

Summary:
- Information needed to interact with a given business
- Access messages: save_business, find_business, get_businessDetail, delete_business
**UDDI version 1 data structures: businessService**

- **categoryBag**
  - owning businessKey (UUID)
  - serviceKey (UUID)
  - name
  - description(s)

- **bindingTemplate**

**Summary:**
- Describes a group of Web services
- Access messages: save_service, find_service, get_serviceDetail, delete_service
**UDDI version 1 data structures: businessTemplate**

A mechanism for associating tModels (specs.) with a Service, and for identifying where and how to invoke the service:

- owning serviceKey
- bindingKey (UUID)
- description(s)

**Summary:**
- The technical details necessary to invoke a Web service
- Access messages: save_binding, find_binding, get_bindingDetail, delete_service
UDDI version 1 data structures: tModel

**tModel**
(a.k.a. Service Type)

Identifies information about specifications:
- tModelKey (UUID)
- name
- authorizedName
- operator
- description(s)

Summary:
- Information about specifications for services
- Access messages: save_tmodel, find_tmodel, get_tmodelDetail, delete_tmodel

e.g. URL to WSDL document
Publishing a tModel with WSDL description

<tModel authorizedName="..." operator="..." tModelKey="...">
  <name>StockQuote Service</name>
  <description xml:lang="en">
  WSDL description of a standard stock quote service interface
  </description>
  <overviewDoc>
    <description xml:lang="en">WSDL source document.</description>
    <overviewURL>http://stockquote-definitions/stq.wsdl</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="uuid:C1ACF26D-9672-4404-9D70-39B756E62AB4"
      keyName="uddi-org:types" keyValue="wsdlSpec"/>
  </categoryBag>
</tModel>

Registry APIs (SOAP Messages)

Publishers API

Save things

- save_business
- save_service
- save_binding
- save_tModel

Delete things

- delete_business
- delete_service
- delete_binding
- delete_tModel

Security

- get_authToken
- discard_authToken

Inquiry API

Find things

- find_business
- find_service
- find_binding
- find_tModel

Get Details about things

- get_businessDetail
- get_serviceDetail
- get_bindingDetail
- get_tModelDetail
**UDDI Query Patterns**

**Browse pattern**
- broad search to specific drill-down
- find_xxx APIs

**Drill-down pattern**
- given a key, retrieve all details
- get_xxx APIs

**Invocation pattern**
Requestor should cache bindingTemplates. Since information can change, cache may need to be refreshed.
- Requestor starts with a cached bindingTemplate
  - Use accessPoint data and knowledge of service
- Invoke service
  - If fails (bad location), go to UDDI to refresh data
  - Retry invoke service call
**Service Requestor algorithm**

- **Implement code** to interact with Web services which support a particular specification (standard)

- **find_business**: pass it a tModelBag referencing the tModel(s) that define the desired specification
  - Returns serviceInfos that have BindingTemplates that reference these tModel(s)

- **find_binding** using the serviceKey from one of the serviceInfos

- **get_bindingDetail** to obtain the access point

- **Invoke** the Web service using its access point or hosting redirector

Each of these steps need code that builds and sends messages, then parses response message.
**UDDI4J: an Open-source Java API**

Open-source Java bindings for UDDI messages
- Creates SOAP messages via Java method calls with an API that maps to UDDI message elements
- Other housekeeping chores to make your UDDI implementation work easier

Read Doug Tidwell's "UDDI4J: Matchmaking for Web services" to get started

UDDI4J source and binaries available
- [oss.software.ibm.com](oss.software.ibm.com) (IBM's open source software site)
- OSI-approved open-source licence
- originally part of the WSTK; now available on our open source site.
UDDI4J Sample Code:
Find businesses starting with "S" and list

UDDIProxy proxy = new UDDIProxy();
proxy.setInquiryURL
   ("http://www.ibm.com/services/uddi/testregistry/inquiryapi");
proxy.setPublishURL
   ("https://www.ibm.com/services/uddi/testregistry/protect/publishapi");
BusinessList bl = proxy.find_business("S", null, 0);

Vector businessInfoVector = bl.getBusinessInfos().getBusinessInfoVector();

for (int i = 0; i < businessInfoVector.size(); i++)
{
    BusinessInfo businessInfo = (BusinessInfo)businessInfoVector.elementAt(i);
    System.out.println(businessInfo.getNameString());
}
UDDI4J Sample Code:
Publish business info, then verify it

UDDIPrxy proxy = new UDDIPrxy();
AuthToken token = proxy.get_authToken("userid", "password");
Vector entities = new Vector();
BusinessEntity be = new BusinessEntity(""");
be.setName("Sample business");
entities.addElement(be);
BusinessDetail bd = proxy.save_business(token.getAuthInfoString(), entities);

// now retrieve what we published for verification
Vector businessEntities = bd.getBusinessEntityVector();
BusinessEntity returnedBusinessEntity =
    (BusinessEntity)(businessEntities.elementAt(0));
System.out.println("Returned businessKey:" +
    returnedBusinessEntity.getBusinessKey());
Importance of Standards

The need for standard interfaces is obvious

- Industry verticals need to standardize services to use common specifications
- WSDL-aware tooling standardizes the creation & use of Web Services

So customers code to a common tModel

- Locate multiple service providers
- Select one, use it
- Assuming common tModels return same results, can switch to another provider for availability or performance reasons
Why do we need a Private UDDI?

So, given a public UDDI Business Registry, why would anyone need a Private UDDI?

- UDDI Business Registry node is not “target rich”
- Early web services adopters have identified need to run their own UDDI
- Control of their own web services meta-data
  - How published, Who can access

A private UDDI is a UDDI spec compliant registry operated outside the UBR

- i.e. privately owned and operated
Six Species of UDDI Registries

All support standard UDDI API, different roles/content

- Public Operators (UBR)
- E-marketplace
- Portal
- Partner-catalog
- Internal Enterprise App. Integration
- Test
Public UDDI Business Registry (UBR)
a.k.a. "operator cloud"

Most of the UDDI.org press refers to this

Pros:
- Serious players establishing Web services repository and service description standards
- Will promote ISV and business activity in web services
- OK for early design time browser-based lookups only

Cons:
- Diffuse (not “target rich”)
- All sorts of businesses: from flowers to fish sellers
- Spammers, posers, laggards, experimenters
- Will not support dynamic finds reliably

Operators follow business contract to participate
M2M e-business on the transactional web will not directly use the UBR
The e-marketplace UDDI

A "target rich" environment

A UDDI node where industry organizations advertise their Web services
  - e-marketplace
  - a standards body
  - consortium of organizations that participate and compete in the industry

Access could be restricted to specific players

Could use custom, industry-specific taxonomies

Allows e-marketplace to do value-add
  - QOS guarantees
  - Standard industry-specific tModels
  - Better-business-bureau style vetting of business entities

This is where “finds” in serious M2M B2B will start
  - Analysts: the future of e-business is with e-marketplaces
The Web Services Portal, or Transactional Web presence UDDI

The UDDI on acme.com’s fire wall

Literally this UDDI is to the transactional web as www.acme.com is to the content web

- acme.com’s presence on the transactional web

Contains the meta-data for all the Web services that acme.com provides to external partners

When acme.com wants to deploy a Web service, acme.com publishes the Web service description to its portal UDDI
The Partner Catalog UDDI

Behind the firewall UDDI
- rolodex-like function for Web services

Very target-rich environment

Contains service descriptions only for those services and business partners for which a company has formal agreements/relationships
- better reflection of current business partnering

Content can be copied from e-marketplace UDDI, cloud, or partner’s portal UDDI
- pasted into partner catalog UDDI

A company's M2M applications do finds against this UDDI
The EAI UDDI

UDDI dedicated to internal EAI behind the firewall

Contains web services meta-data for acme.com’s Enterprise Application Integration

Similar to partner catalog UDDI

- No external web services are described in this UDDI
- Only your company’s internal web services appear here
EAI UDDI and Late Binding

SOAP+WSDL+UDDI is useful for an application or data integration strategy: offers loose coupling and late binding

existing enterprise functions / data

SOAP+WSDL interfaces

query

set data

commercial software e.g.: IBM DB2 Universal Database

EAI Catalog private UDDI registry

layered service

UDDI methods

FIND

BIND
The Test UDDI

This is the UDDI that developers use to design, code and test their software

- service providers
- service requestors

Web services are tested here before their meta data is promoted from here to final deployment

- into portal UDDI, E-marketplace UDDI or UBR

Use local implementation (private UDDI), or use test registries on Internet
UDDI Specification Roadmap

UDDI version 1
- Public registry sites now in production

UDDI version 2
- Specification published 6/2001
- V2 UDDI Business Registry beta impls: late 2001
- Hewlett-Packard and SAP deploy additional registry sites

UDDI version 3
- Requirements gathering underway
- Specifications to be published YE/2001
- Beta registry sites – spring, 2002
- Transition specification to industry standards body
New specifications (visit uddi.org):
- Programmer's API Specification V2.00
- Data Structure Specification V2.00
- Replication Specification V2.00
- Operator's Specification V2.00

New features:
- Description of Complex Organizations - business units, departments, divisions, and subsidiaries
- Better internationalization for describing businesses and services in multiple languages
- Additional categorization and identifier schemes
  - register 'checked' and 'unchecked' taxonomies
- Richer searching options: more expressive query parameters, using more fields and complex combinations of fields

UDDI version 2.00 UBRs will appear later this year
**UDDI Version 3.00**

**Requirements**

- Security
- Scalability
- Subscriber Node Support
- Support for Registrars
- Layered Web Services (Associations of business entities working together to provide a service)
- Workflow and Service Composition

**Target: early 2002**
Tooling

The problems:
- The details are complex
- We will be doing similar things for many services
- We need to speed up development

The solution:
- A new generation of service authoring and application assembly tools
- e.g. IBM WebSphere Studio Application Developer
WebSphere Studio Application Developer

Features:

▶ Generate client interface, server code template from WSDL descriptions of web services
▶ Wizards to guide development, debugging, and deployment of web service requestors and providers
▶ A number of new integrated XML tools make for a powerful and easy-to-use environment for developing web services applications.
▶ Well-integrated with WebSphere
▶ Complements Visual Age for Java environment

Availability:

How tools can help

You want to publish a service

- write a local implementation as a Java Bean
  - VAJ4 helps to speed this task
- Re-package it as a WSDL-compliant web service using WSAD Web Service wizard
- Use WSAD UDDI wizard to publish to UDDI registry
- WSAD also helps deploy to server

You want to write an application that uses a service

- Find the service in the UDDI Registry with the WSAD UDDI Explorer
- Import the service into WSAD
- WSDL description tells WSAD everything it needs to know to generate local service proxy code for you
- Use the service proxy the same as you would any local Java class
  - VAJ4 speeds the task of writing client application code
**WebSphere Private UDDI**

**A complete implementation of a UDDI registry**
- based on IBM's UDDI Business Registry
- underlying database: IBM DB2 Universal Database
- supports complete UDDI version 1 message spec

**Technology preview available for free download**
- ibm.com/websphere/developer/downloads/UDDIregistry.html
- FREE electronic support available
- Requires WebSphere App Server Advanced Edition version 4, and DB2 Universal Database, Personal Edition 7.1, which are also available for free download
  - The URL above makes all required pieces available

**Use locally (e.g. for testing and experimentation) or deploy on network**
**UDDI Resources**

**White papers, product offerings**
- http://www.ibm.com/webservices

**Software:**
- UDDI4J - open-source Java API to access UDDI
  - code: http://oss.software.ibm.com
- Private UDDI preview for developers edition
- Web Services ToolKit (WSTK)

**Articles, tutorials:** http://ibm.com/developerworks/webservices
- Steve Graham: Role of private UDDI nodes in Web services
  - Part 1: Six species of UDDI
  - Part 2: Private nodes and operator nodes
- Doug Tidwell: Introduction to UDDI4J
  - ibm.com/developerWorks/library/ws-uddi4j.html
UDDI Resources

From ibm.com/services/uddi:
- web access to UDDI registry
- developer links and resources
- news, products, services

Microsoft has a UDDI site, too.
- uddi.microsoft.com

UDDI.org
- specifications, whitepapers, best practices
Summary

Service discovery runs gamut from simple techniques to sophisticated UDDI

UDDI is more than just the Universal Business Registry

- Private UDDI is growing the use of UDDI API
- Private UDDI promotes Enterprise Application Integration and partner integration via Web Services
- Private UDDI can greatly aid in the deployment of Web Services solutions

Availability of second-tier tools like a private UDDI show the maturity of Web services technologies

The time to get started on Web services is NOW!

And IBM is here to help you.
Questions?

ibm.com/webservices
whitepapers on IBM's vision of dynamic e-business enabled by web services

ibm.com/developerWorks/webservices
Web Services Zone on developerWorks - resources for customers and developers on the use of XML

ibm.com/alphaworks
site for free emerging tools and technologies from IBM

oss.software.ibm.com
UDDI4J, WSDL4J open source Java class libraries

xml.apache.org
Apache SOAP and other open source XML tools

uddi.org
UDDI consortium - information and resources

xml.org
XML standard vocabularies repository

ebxml.org
electronic business in XML initiative

PDFs of this and my other presentations are available at this site

Mark Colan - mcolan@us.ibm.com
ibm.com/developerworks/speakers/colan