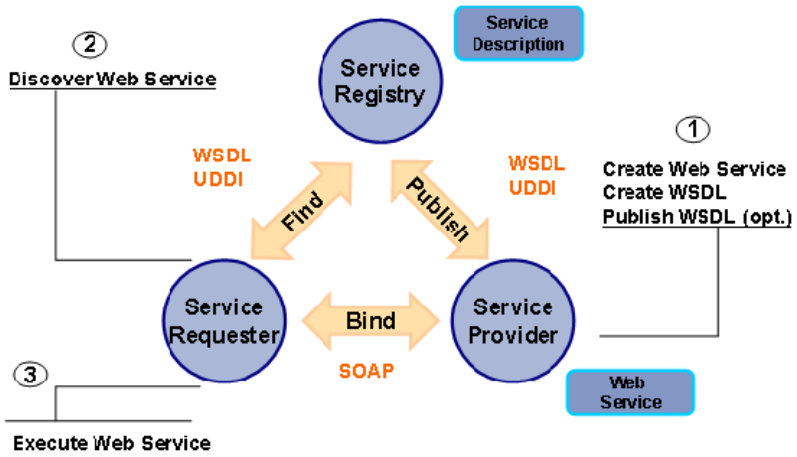
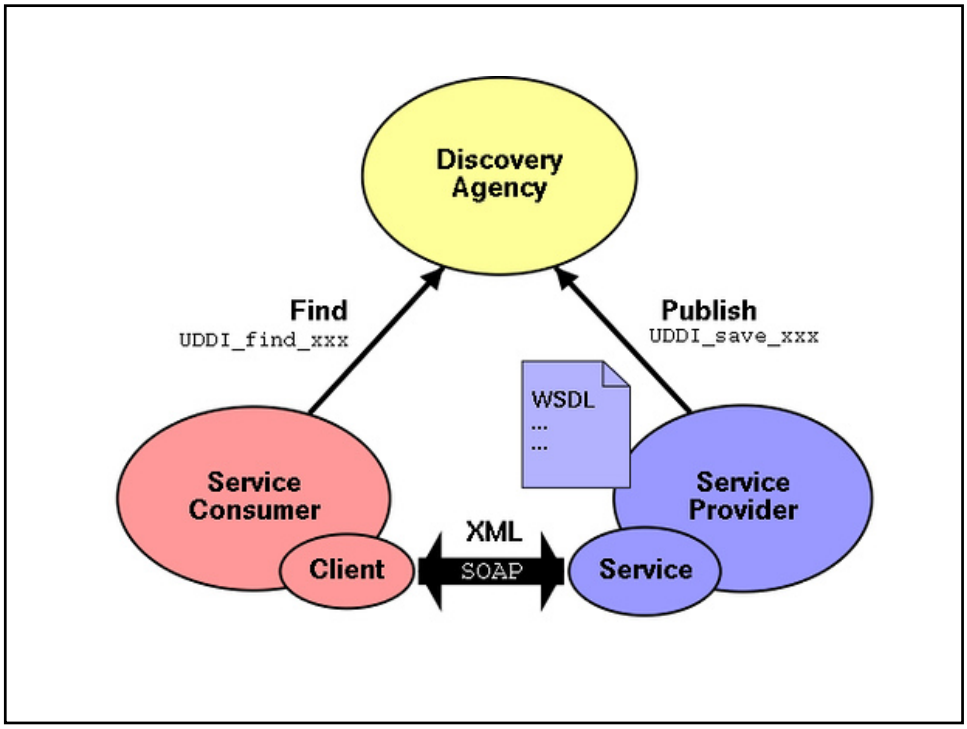
Web Services - Basics

SOAP – WSDL - UDDI





SOAP

Simple Object Access Protocol

SOAP

- SOAP is a simple XML-based protocol to let applications exchange information over HTTP.
- Protocol for accessing a Web Service.
- SOAP stands for Simple Object Access Protocol
  - a communication protocol
  - for communication between applications
  - a format for sending messages
  - communicates via Internet
  - platform independent
  - language independent
  - based on XML
  - simple and extensible
  - SOAP is a W3C recommendation

SOAP - Need

- It is important for application development to allow Internet communication between programs.
- Traditionally applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.
- A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Building Blocks

- A SOAP message is an ordinary XML document containing the following elements:
  - An *Envelope element* that identifies the XML document as a SOAP message
  - A *Header element* that contains header information
  - A *Body element* that contains call and response information
  - A *Fault element* containing errors and status information
- All the elements above are declared in the default namespace for the SOAP envelope:
  http://www.w3.org/2001/12/soap-envelope
- Thehe default namespace for SOAP encoding and data types is:
  http://www.w3.org/2001/12/soap-encoding

SOAP Syntax Rules

- A SOAP message MUST
  - be encoded using XML
  - use the SOAP Envelope namespace
  - use the SOAP Encoding namespace
- A SOAP message must NOT contain
  - a DTD reference
  - XML Processing Instructions

SOAP Message Skeleton

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

SOAP Envelope Element

SOAP Envelope element is the root element of a SOAP message.

SOAP Envelope - Example

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-
envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soa
p-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

WSDL

WSDL (*Web Services Description Language*) is an XML-based language for describing Web services and how to access them.

WSDL

- Web Services Description Language
- Written in XML - an XML document
- To describe Web services
- To locate Web services
- WSDL is a W3C recommendation
- It specifies the location of the service and the operations (or methods) the service exposes.

WSDL Document Structure

- A WSDL document describes a web service using these major elements:

Element	*Defines*
<types>	Data types used by the web service
<message>	Messages used by the web service
<portType>	Operations performed by the web service
<binding>	Communication protocols used by the web service

WSDL Ports

- The **<portType>** element is the most important WSDL element.
- It *describes* a web service, the *operations* that can be performed, and the *messages* that are involved.
- The <portType> element can be compared to a function library (or a module, or a class) in a traditional programming language.

WSDL Messages

- The **<message>** element defines the data elements of an operation.
- Each message can consist of one or more *parts*. The *parts* can be compared to the *parameters* of a function call in a traditional programming language.

WSDL Types

- The **<types>** element defines the data types that are used by the web service.
- For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL Bindings

The **<binding>** element defines the message format and protocol details for each port.

WSDL - Example

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

- The **<portType>** element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.
- The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".
- The **<message>** elements define the **parts** of each message and the associated data types.
- Compared to traditional programming, *glossaryTerms* is a function library, "getTerm" is a function with "getTermRequest" as the input parameter, and “getTermResponse” as the return parameter.

WSDL Ports

A WSDL port describes the interfaces (legal operations) exposed by a web service.

- The **<portType>** element is the most important WSDL element.
- It defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.
- The port defines the connection point to a web service. It can be compared to a function library (or a module, or a class) in a traditional programming language. Each operation can be compared to a function in a traditional programming language.

Operation Types

Type	Definition
One-way	Operation can receive a message but will not return a response
Request-response	Operation can receive a request and will return a response
Solicit-response	Operation can send a request and will wait for a response
Notification	Operation can send a message but will not wait for a response

WSDL Ports (contd)

One-Way Operation

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

- The port "glossaryTerms" defines a one-way operation called "setTerm".
- The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value". However, no output is defined for the operation.

WSDL Ports (contd)

Request-Response Operation

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

- The port "glossaryTerms" defines a request-response operation called "getTerm".
- The "getTerm" operation requires an input message called "getTermRequest" with a parameter called "term", and will return an output message called "getTermResponse" with a parameter called "value".

Binding to SOAP

```
<message name="getTermRequest">
 <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
 <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
  <input message="getTermRequest"/>
  <output message="getTermResponse"/>
 </operation>
</portType>

<binding type="glossaryTerms" name="b1">
  <soap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
   <soap:operation soapAction="http://example.com/getTerm"/>
   <input><soap:body use="literal"/></input>
   <output><soap:body use="literal"/></output>
 </operation>
</binding>
```

- The **binding** element has two attributes - name and type.
- The name attribute defines the name of the binding, and the type attribute points to the port for the binding, in this case the "glossaryTerms" port.
- The **soap:binding** element has two attributes - style and transport.
- The style attribute can be "rpc" or "document". In this case we use document. The transport attribute defines the SOAP protocol to use. In this case we use HTTP.
- The **operation** element defines each operation that the port exposes.
- For each operation the corresponding SOAP action has to be defined. You must also specify how the input and output are encoded. In this case we use "literal".

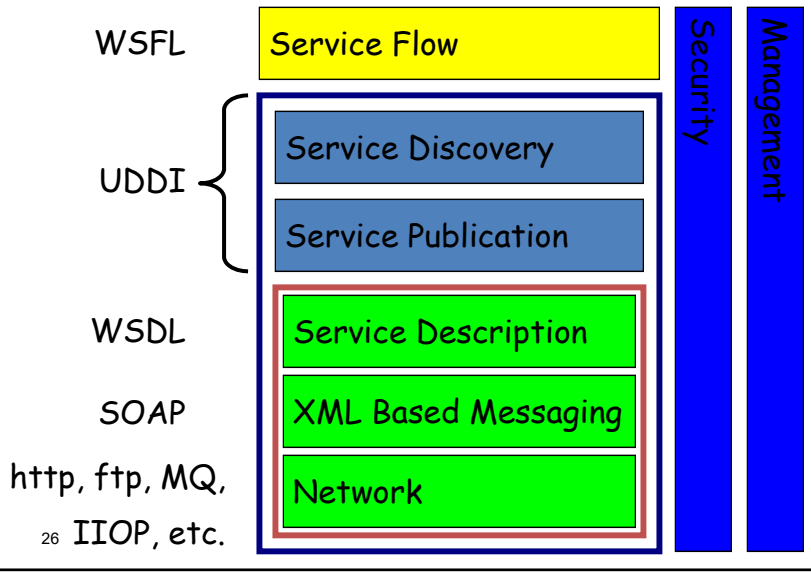
WSDL and UDDI

- Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web services.
- UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.
  - A directory for storing information about web services
  - A directory of web service interfaces described by WSDL
  - UDDI communicates via SOAP
- UDDI uses WSDL to describe interfaces to web services
- Additionally, cross platform programming features are addressed by adopting SOAP

UDDI - Overview

25

The Web Services Stack



26

Basics: SOAP

- An XML based communication protocol that allows programs to exchange information via the Internet in a standardized way – regardless of the underlying programming model
- An XML-based protocol for calling remote functions or objects over the Internet
- Designed by Microsoft, DevelopMentor and UserLand Software
- Subsequently vendors, like IBM, have jointly worked on the specification and submitted version 1.1 to W3C (World Wide Web Consortium)
- SOAP plays an important role in Internet communication because
  - it is simple
  - Internet friendly
  - based on XML
  - implementation-neutral

27

Basics: WSDL (Web Services Definition Language)

- The Problem: SOAP provides a standard way of transporting messages for use by Web Services, but it doesn't provide any way of describing what the format of the message should be for any given Web Service.
- The Caller had no way of knowing how to create the SOAP message that would represent the call.
- The Solution: WSDL is used to define and describe Web Services. Just as header files or type libraries are used to describe binary libraries.
- Result is self-describing Web Services

28

Basics: UDDI

- **Universal Description, Discovery, & Integration**
- **A project to encourage interoperability and adoption of web services**
  - Standards-based specifications for service description and discovery
- **A set of internet-based implementations**
  - UDDI business registry
  - Interoperating to share registrations
- **Partnership among industry & business leaders**
  - Initiated by Ariba, IBM, and Microsoft

29

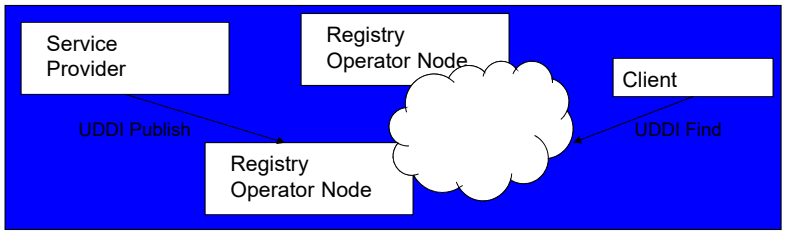
Basics: UDDI (Universal Description, Discovery & Integration)

- Specification to Publish & Discover Web Services
- Shared Distributed Registry on the Web
- Analogy: Universal Phone Book
- UDDI enables three basic functions:
  - Publish function – how a Web Service registers
  - Find function – how a client finds a Web Service
  - Bind function – how the client connects and interacts with a Web Service

30

Basics: UDDI

- UDDI Concepts:
  - Global registry hosted by UDDI Operators
  - UDDI Operators (IBM, Microsoft)
  - Defined process for managing information
  - Business publish their information for free
  - "Registered once, Published everywhere principle"
  - Information is replicated = Service Cloud



31

Basics: UDDI

- How it works:
  - Standards bodies populate standard service descriptions
  - Service providers add services they provide
  - UDDI business registry assigns unique identifiers to each service
  - Online marketplaces, search engines, etc. query the registry
  - Data returned is used to invoke methods on remote Web Services

Name	
Software AG	
Key	
30d0c75c-f423-4ac8-90c5-a00928577b71	
Discovery URL	**Usage**
http://uddi.microsoft.com/discovery?businessKey=30D0C75C-F423-4AC8-90C5-A00928577B71	businessEntity
Description	
System Software Vendor for Transactional Software, EAI and XML	
Contact Name	**Role**
Rainer Glaap	*empty*
Locator Name	**Value**
Information Services and Data Processing Services	514

Back

32

Basics: UDDI

```
<?xml version="1.0" encoding="utf-8" ?>
- <businessEntity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    businessKey="30D0C75C-F423-4AC8-90C5-A00928577B71" operator="Microsoft Corporation" authorizedName="Rainer F. Glaap"
    xmlns="urn:uddi-org:api">
  - <discoveryURLs>
      <discoveryURL useType="businessEntity">http://uddi.microsoft.com/discovery?businessKey=30D0C75C-F423-4AC8-90C5-
        A00928577B71</discoveryURL>
    </discoveryURLs>
    <name>Software AG</name>
    <description xml:lang="en">System Software Vendor for Transactional Software, EAI and XML</description>
  - <contacts>
    - <contact useType="">
        <description xml:lang="en">Director Product Marketing</description>
        <personName>Rainer Glaap</personName>
        <phone useType="">+49-6151-92-0</phone>
        <email useType="">rainer.glaap@softwareag.com</email>
      </contact>
    </contacts>
  - <businessServices>
    - <businessService serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" businessKey="30d0c75c-f423-4ac8-90c5-
        a00928577b71">
        <name>Tamino Demo Web Service</name>
        <description xml:lang="en">Real Estate Simulation</description>
      - <bindingTemplates>
        - <bindingTemplate serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" bindingKey="ad090451-9851-4687-82d3-
            73cb49a4d15d">
            <description xml:lang="en">software ag internal use only at the moment</description>
            <accessPoint URLType="http">http://localhost/demoWebSvc/rpcrouter</accessPoint>
            <tModelInstanceDetails />
          </bindingTemplate>
        </bindingTemplates>
      + <categoryBag>
      </businessService>
    </businessServices>
  + <categoryBag>
  </businessEntity>
```

Basics: UDDI

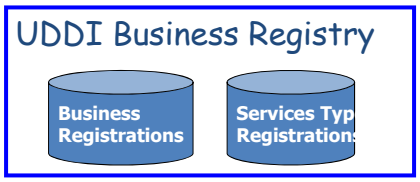
- Kinds of Data:
  - White Pages
    - Name of Business
    - Contact Information
    - Description of the Company
  - Yellow Pages
    - Business Classification Information
    - Based on NAICS, UNSPSC, or Geographical Index
    - List of Products (multiple entries)
  - Green Pages
    - Technical Information about services
    - Example: business processes, binding info, etc.

34

How UDDI Works

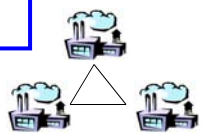
1) Software companies, standards bodies and programmers populate the registry with descriptions of different tModels

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



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

3) UDDI assigns a programmatically unique identifier (UUID) to each tModel and business registration and stores them in an Internet registry



5) Businesses use this data to facilitate easier integration with each other over the Web

35

Basics: UDDI

- UDDI Information Model:
  - Business Information
  - Business Service Information
  - Binding Information
  - Technical Information

tModel

- A Technical Model (t-Model) is a data structure that represents an XML Web services type (a generic representation of a registered service) in a Universal Description, Discovery, and Integration (UDDI) registry.
- Each business that is registered with UDDI categorizes all of its Web services according to a defined list of service types.
- The t-Model is an abstraction for a technical specification of a service type; it organizes the service type information and makes it accessible in the registry database.

36

Basics: UDDI

```
- <businessEntity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    businessKey="30D0C75C-F423-4AC8-90C5-A00928577B71" operator="Microsoft Corporation" authorizedName="Rainer F. Glaap"
    xmlns="urn:uddi-org:api">
  + <discoveryURLs>
    <name>Software AG</name>
    <description xml:lang="en">System Software Vendor for Transactional Software, EAI and XML</description>
  + <contacts>
  + <businessServices>
  + <categoryBag>
  </businessEntity>
```

- businessEntity (Business Information, White Pages):
  - **businessKey attribute** – Unique identifier
  - **authorizedName attribute** – Individual who publishes the information
  - **operator attribute** – Name of UDDI registry operator
  - **discoveryURL element** – URLs that point to alternative discovery means
  - **name element** – Name of the organization
  - **description element** – A short business description
  - **contacts element** – Contact information for the business
  - **businessServices element** – List of services the business provides
  - **indentifierBag element** – List of alternative identifiers for the business
  - **categoryBag element** – Tag business with specific classification info

37

Basics: UDDI

```
<?xml version="1.0" encoding="utf-8" ?>
- <businessEntity xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    businessKey="30D0C75C-F423-4AC8-90C5-A00928577B71" operator="Microsoft Corporation" authorizedName="Rainer F. Glaap"
    xmlns="urn:uddi-org:api">
  - <discoveryURLs>
      <discoveryURL useType="businessEntity">http://uddi.microsoft.com/discovery?businessKey=30D0C75C-F423-4AC8-90C5-
        A00928577B71</discoveryURL>
    </discoveryURLs>
    <name>Software AG</name>
    <description xml:lang="en">System Software Vendor for Transactional Software, EAI and XML</description>
  - <contacts>
    - <contact useType="">
        <description xml:lang="en">Director Product Marketing</description>
        <personName>Rainer Glaap</personName>
        <phone useType="">+49-6151-92-0</phone>
        <email useType="">rainer.glaap@softwareag.com</email>
      </contact>
    </contacts>
  - <businessServices>
    - <businessService serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" businessKey="30d0c75c-f423-4ac8-90c5-
        a00928577b71">
        <name>Tamino Demo Web Service</name>
        <description xml:lang="en">Real Estate Simulation</description>
      - <bindingTemplates>
        - <bindingTemplate serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" bindingKey="ad090451-9851-4687-82d3-
            73cb49e4d15d">
            <description xml:lang="en">software ag internal use only at the moment</description>
            <accessPoint URLType="http">http://localhost/demoWebSvc/rpcrouter</accessPoint>
            <tModelInstanceDetails />
          </bindingTemplate>
        </bindingTemplates>
      + <categoryBag>
      </businessService>
    </businessServices>
  + <categoryBag>
  </businessEntity>
```

Basics: UDDI

```
- <businessServices>
  - <businessService serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" businessKey="30d0c75c-f423-4ac8-90c5-
      a00928577b71">
      <name>Tamino Demo Web Service</name>
      <description xml:lang="en">Real Estate Simulation</description>
    - <bindingTemplates>
      - <bindingTemplate serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" bindingKey="ad090451-9851-4687-82d3-
          73cb49a4d15d">
          <description xml:lang="en">software ag internal use only at the moment</description>
          <accessPoint URLType="http">http://localhost/demoWebSvc/rpcrouter</accessPoint>
          <tModelInstanceDetails />
        </bindingTemplate>
      </bindingTemplates>
    + <categoryBag>
    </businessService>
  </businessServices>
```

- businessServices (Business Service Information, Yellow Pages):
  - **serviceKey attribute** – Unique identifier for a particular service
  - **businessKey attribute** – Business Entity key of business entity that contains this service
  - **name element** – Name of the service family
  - **description element** – Description of service family
  - **bindingTemplates element** – Technical description
  - **categoryBag element** – Tag business with specific classification info

39

Basics: UDDI

```
- <bindingTemplates>
  - <bindingTemplate serviceKey="64d709ee-aac3-431a-9e9b-a94bdeb38866" bindingKey="ad090451-9851-4687-82d3-
      73cb49a4d15d">
      <description xml:lang="en">software ag internal use only at the moment</description>
      <accessPoint URLType="http">http://localhost/demoWebSvc/rpcrouter</accessPoint>
      <tModelInstanceDetails />
    </bindingTemplate>
  </bindingTemplates>
```

- bindingTemplate (Binding Information, Green Pages):
  - **bindingKey attribute** – Unique identifier for a particular binding template
  - **serviceKey attribute** – Key of the Business service key that contains the binding template
  - **description element** – Description of binding template
  - **accessPoint element** – Entry point for the service, i.e. email address, URL, etc.
  - **hostingRedirector element** – Point to another binding template
  - **tModelInstanceDetails element** – List tModel info structures, tModel acts as a fingerprint for the service

40

Basics: UDDI

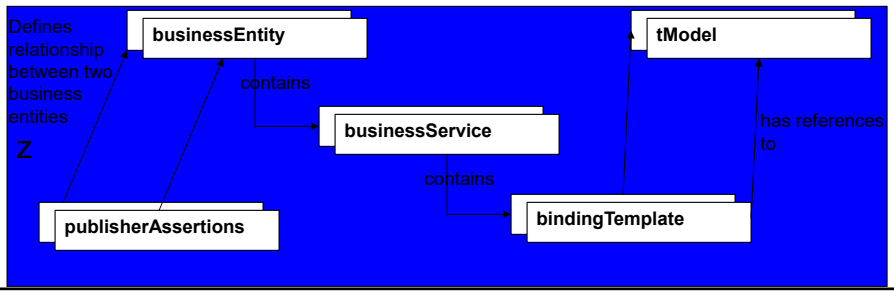
```
- <tModelDetail generic="1.0" xmlns="urn:uddi-org:api" operator="Software AG">
  - <tModel tModelKey="UUID:297AAA47-2DE3-4454-A04A-CF38E889D0C4" operator="Microsoft Corporation"
      authorizedName="Microsoft UDDI Publishing Authority">
      <name>microsoft-com:geoweb:2000</name>
      <description xml:lang="en">Microsoft Geographic Taxonomy (August 2000 Release)</description>
    - <overviewDoc>
        <description xml:lang="en">Microsoft Geographic Taxonomy</description>
        <overviewURL>http://uddi.microsoft.com/tmodels/microsoft-com.geoweb.2000/overview.htm</overviewURL>
      </overviewDoc>
    - <categoryBag>
        <keyedReference tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4" keyName=""
          keyValue="categorization" />
      </categoryBag>
    </tModel>
  </tModelDetail>
```

- tModel (Technical Information):
  - **tModelKey attribute** – Unique identifier for a particular binding tModel
  - **authorizedName attribute** – Name of individual who published this tModel information
  - **operator attribute** – Name of UDDI registry operator site
  - **name element** – Name of tModel
  - **description element** – Description of tModel
  - **overviewDoc element** – Reference to remote instructions or descriptions related to the tModel
  - **identifierBag element** – List of record id numbers for this tModel
  - **categoryBag element** – List of record classification information for this tModel

41

Basics: UDDI

- Publisher Assertion:
  - **Introduced with UDDI version 2.0**
  - **Publisher Assertion Structure contains:**
    - **fromKey element – First business entity assertion is made for**
    - **toKey element – Second business entity assertion is made for**
    - **keyedReference element – Designates the relationship type**



UDDI Version 3

- Multi-registry environment
  - Root and affiliate registries
- URI-based keys
- Digital signature support
- Complex categorization support
- Information extensibility
- Improved WSDL support
- Subscription API

43

Issues

- Security
  - UDDI Server denial-of-service attack
  - Open access – everyone can see your services
    - This is why many UDDI implementations are internal
- Support
  - Use more prevalent, but still “crossing the chasm”

44