eBusiness Architectures and Standards

Anil K. Nori
Software Architect
Microsoft
USA
anilnori@microsoft.com

Acknowledgements

• Asera Inc.
  – eBusiness functionality, architecture
• http://www.w3.org/XML/Schema
• http://www.w3.org/TR/soap
• http://www.w3.org/TR/wsdI
• http://www.uddi.org
• http://www.ebXML.org
• http://www.rosettanet.org
• Microsoft
  – SOAP, UDDI material
Agenda

- Evolution of eBusiness
- eBusiness Applications
- eBusiness Architecture
- eBusiness Standards
- Discussion

Fundamental eBusiness Problems

- eBusiness is all about automating business processes of an enterprise.
  - Within the enterprise: across divisions, employees
  - Across the enterprise: suppliers, partners, customers
- Most eBusiness interactions are complex, labor and information intensive
  - complex, collaborative manual processes
  - timely and accurate information exchange is a challenge
- Commerce is fragmented by geography
  - creates inefficient markets, uninformed buyers and sellers
- Supply chains are bloated with excess inventory
  - inability to see and plan right products and quantities
Demand for Market Visibility

- Price Visibility
- Availability Visibility
- Supplier Visibility
- Product Visibility

Need for Collaboration

- Within and across enterprise
- Human and machine collaboration
- Partner centric
- Support exception and event-driven alerts
- Support response management
Opportunities for Optimization

- Analyze and optimize enterprise business processes
- Synchronize plans and forecasts across multiple parties
- Optimize integrations
- Dynamic reconfiguration of processes

Agile Enterprise

- Enterprise business processes are complex and continue to evolve
- Customization is critical
- Personalization is essential
  - Any geography
  - Any device
  - Any person
- Efficient manageability is important
eBusiness: *Past*

- Using FAX, Paper and Telephone
  - inefficient, error-prone, expensive
- EDI networks
  - standardized, reduced errors
  - batch oriented, expensive
  - proprietary VANs
  - point-to-point connections; lacks market Visibility
  - expensive; cut small suppliers out

BUT, EDI still lives!

eBusiness: *First Generation*

- Basic Electronic Commerce (E-Commerce)
  - Internet based sales channel
  - No intermediary
  - Mostly brochure-ware
  - Catalogs, marketing collateral
  - Sales promotions
**eBusiness: Next Generation**

- Collaborative and communities of Commerce
  - Create market Visibility
  - Collaboration across geographies
  - Reflection of complex business workflow between demand and supply chains
  - Complex and dynamic trade, e.g. auctions and exchanges
  - Real-time supply chain management
  - Supports transactional and value added services

**Enterprise Business Processes**

- Forecast
- Purchase Order
- Manufacturing
- Inventory
- Accounts Receivable
- Sales Order
- Shipping
- Finished Goods
- Accounts Payable
- Receiving
The Bigger Commerce Chain Picture

Agenda

- Evolution of eBusiness
- eBusiness Applications
- eBusiness Architecture
- eBusiness Standards
- Discussion
A Typical Commerce Process Is Not Simple

Customer Records Not Synchronized
Disconnected Web Channels
Manual Order Tracking Process
Processes Not Integrated through Order Fulfillment

Resulting In A Chaotic Customer Experience….
Streamlined Processes

Typical Commerce Solutions
Some fundamentals of Supply Chain

How do you define Supply Chain Management?
Organization of the overall business processes to enable the profitable transformation of raw materials or products into finished goods and their timely distribution to meet customer demand.

What is the Goal of Supply Chain Management?
Improving customer service through reliable and on-time delivery while reducing the cost of operations such as inventory, transportations cost, resource allocation and asset utilization.

Major elements of the Supply Chain?
Design – Plan – Buy – Make – Move (Store / Distribute) – Sell

Supply Chain is Complex and Expansive

- Multiple Systems
- Applications Built For Inside the Four Walls –No Visibility
- Multiple Infrastructures
Supply Chain Domains and Targeted Values

**Targeted Value**
- Maximized planning accuracy
- Maximized capacity utilization
- Maximized inventory turns

**Enterprise Planning**
- Collaborative forecasting
- Allocation planning
- Proactive sales

**Planning**
- Supplier inv.
- Cap visibility (ATP)
- Warehouse inventory
- Visibility

**Enterprise Execution**
- Replenishment
- Returns
- Procurement
- Pools Sched
- Compliance
- VMI

**Execution**
- ATP
- Service parts
- Inventory

**Demand Planning**
- New Prod intro
- Roll to Prod
- RCOs
- Product retire

**Procurement**
- Sourcing
- eRFI
- Procurement
- eProcurement

**Management**
- ATP
- Sales Activity
- Mgmt
- Service parts
- Inventory

**Selling**
- Ship status
- Status
- Visibility
- Connectivity
- Cost Service
- CRM

**Sell Side**
- Marketing
- Sales
- Service
- Support
- Customer feedback

New Supply Chain Models

<table>
<thead>
<tr>
<th>Offering Area</th>
<th>Traditional Focus</th>
<th>New Business Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fulfillment</td>
<td>Distribution, Warehouse and Transportation Operations and Systems</td>
<td>eFulfillment</td>
</tr>
<tr>
<td></td>
<td>Order Fulfillment</td>
<td>3PL, 4PL</td>
</tr>
<tr>
<td>Supply Chain Planning, Strategy and Synchronization</td>
<td>Network Optimization</td>
<td>eSynchronized Supply Chain</td>
</tr>
<tr>
<td></td>
<td>Demand Planning Solutions</td>
<td>eSCVA Diagnostic</td>
</tr>
<tr>
<td></td>
<td>Merger Integration</td>
<td>eCollaborative Planning</td>
</tr>
<tr>
<td></td>
<td>Supply Chain Strategy</td>
<td>eIntegrated Manufacturing</td>
</tr>
<tr>
<td>Integrated Manufacturing</td>
<td>Manufacturing Strategy</td>
<td>Collaborative Manufacturing</td>
</tr>
<tr>
<td></td>
<td>Manufacturing Operations and Systems</td>
<td>eCollaborative Capacity Management</td>
</tr>
<tr>
<td></td>
<td>Integrated Product Design</td>
<td>eProduct Design</td>
</tr>
<tr>
<td>Procurement and B2B eMarketplaces</td>
<td>Strategic Sourcing</td>
<td>eProcurement</td>
</tr>
<tr>
<td></td>
<td>Procurement Effectiveness</td>
<td>B2B Exchanges/Collaborative Hubs</td>
</tr>
<tr>
<td></td>
<td>MRO and Service Parts</td>
<td>Operator and Participant Strategy and Implementation</td>
</tr>
<tr>
<td></td>
<td>Changes in channel and business processes</td>
<td>Flexible Marketplace</td>
</tr>
<tr>
<td></td>
<td>Changes in channel and market conditions</td>
<td>Auction Management Services</td>
</tr>
</tbody>
</table>
Key eBusiness Requirements: Summary

• Information visibility
  – Integration
  – Real-time access

• Collaboration
  – Human and Machine interactions
  – Within and across enterprises
  – Synchronous and asynchronous

• Business Process Intelligence
  – Discover, Monitor, Optimize
  – Dynamic

• Agility
  – Design, Develop, Deploy, Execute, Manage
  – Business rules, dynamic processes
  – Customize, Configure, Upgrade

• Personalization
  – Targeted content
  – Devices
  – Globalization
  – Entitlement

Agenda

• Evolution of eBusiness
• eBusiness Applications
• eBusiness Architecture
• eBusiness Standards
• Discussion
Traditional Approaches Require Extensive Code

What About:
- Globalization?
- Business Process Management?
- Device Independence?
- Monitoring?
- Tracking and Reporting?
- Adding a New App?
- Changing the Workflow?
- Adding New Users?
- Changing Rules?

eBusiness Applications Approach

Compose new business processes... as applications
- Leverage existing systems — data and logic
- Create new processes tailored to your enterprise
- Configure and personalize the user experience
- Human interactions

Using a software platform
- Business process system
- Pre-built, common services
- Flexible integration framework

EAI

Traditional Systems
## Properties of eBusiness Applications vs. Web-ified Apps

<table>
<thead>
<tr>
<th>eBusiness Applications</th>
<th>Web-ified Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable business processes</td>
<td>Pre-defined business processes</td>
</tr>
<tr>
<td>Designed for collaboration (many-to-many)</td>
<td>Designed for a single enterprise</td>
</tr>
<tr>
<td>Web Services-based</td>
<td>Data-base driven</td>
</tr>
<tr>
<td>Process based Applications</td>
<td>Monolithic Applications</td>
</tr>
<tr>
<td>Customer-centric</td>
<td>Enterprise-centric</td>
</tr>
<tr>
<td>Web-centric</td>
<td>Web-enabled</td>
</tr>
<tr>
<td>Configured by business analyst changing</td>
<td>Customized by developers modifying code</td>
</tr>
<tr>
<td>business processes</td>
<td></td>
</tr>
<tr>
<td>Open, interoperable</td>
<td>Self-contained</td>
</tr>
<tr>
<td>Flexible, adaptable</td>
<td>Rigid</td>
</tr>
</tbody>
</table>

## Need A Streamlined, Modular Approach to Building eBusiness Applications

![Diagram showing eBusiness Platform and Processes](image-url)
Architectural Foundations for Development of eBusiness Applications

- Unified Architecture
- Process based Applications
- Open & Extensible Framework
- Web Services Framework
- Integrated Development & Deployment
- Security, Scalability, Availability and Manageability
- Support Heterogeneous Environments

eBusiness Platform
Agenda

- Evolution of eBusiness
- eBusiness Applications
- eBusiness Architecture
- eBusiness Standards
- Discussion

eBusiness Standards

- Common Business Objects
  - XML Schema
- Business Processes
  - BPML, XLANG, WSFL
  - WSCI
- Services descriptions, Messaging
  - SOAP, WSDL
- Services directory, discovery
  - UDDI
- B2B Interactions, Protocols
  - ebXML
  - RosettaNet
- Security
  - WS-Security, SAML
- Presentation
  - JSP, ASP

- Wire stack
  - XML Schema
  - SOAP
  - HTTP, SMTP
- Description stack
  - XML Schema
  - WSDL
  - WSFL, WSCI, XLANG
- Discovery stack
  - UDDI
- B2B Protocols
  - ebXML
  - RosettaNet
What is a Web Service?

• A web service is *programmatically available application logic (component) exposed over the Internet*
  – Available to a variety of clients (platform independent)
  – E.g. stock quote, weather, and authentication services
  – Makes building distributed applications less difficult

• Most common metaphor for accessing information is through a web browser
  – Traditional web applications don't expose any application logic
What is a Web Service?

A web service is *programmable application logic accessible via standard Web protocols*.

![Web Service Diagram](image)

Service Oriented Architecture

![Service Oriented Architecture Diagram](image)
Web services interoperability stack

Wire Stack
- SOAP Headers
  - Envelope Extensions
  - XML Messaging
  - Data Encoding
  - Network Protocol
- Security
- Quality of Service
- Manageability

Description Stack
- WSFL
- XLANG
- Agreements
- Process
- WSDL
- Service Interface
- WSDL
- Service Implementation
- XML Schema
- XML

Discovery Stack
- UDDI
  - Directory
- ADS/DISCO
  - Inspection

WSDL – Web Services description Language
UDDI – Universal Description, Discovery and Interaction

XML Schema
What are Schemas?

- Generically, a document that describes what a correct document may contain
- Specifically, a W3C Recommendation for an XML-document syntax that describes the permissible contents of XML documents
- Created by W3C XML Schema Working Group based on many different submissions
- No known patent, trademark, or other IP restrictions
- XML Schema Part 1: Structures: [http://www.w3.org/TR/xmlschema-1/](http://www.w3.org/TR/xmlschema-1/)

What's Wrong with DTDs?

- Unusual, non-XML like syntax
- No data typing, especially for element content
- Limited extensibility
- Only marginally compatible with namespaces
- Cannot use mixed content and enforce order and number of child elements
- Cannot enforce number of child elements without also enforcing order. (i.e. no & operator from SGML)
Example Schema

Product instance:

```xml
<product effDate="2002-08-22">
  <number>55555</number>
  <size>10</size>
</product>
```

Example Schema

Product Schema

```xml
<xsd:schema xmlns:xsd = "http://www.w3.org/2002/XMLSchema">
  <xsd:element name="product" type="ProductType"/>
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element name="number" type="xsd:integer"/>
      <xsd:element name="size" type="SizeType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="SizeType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```
Structure of XML Schema

- Components of XML Schema
  - Declarations vs. Definitions
- Elements and Attributes
- Data types
- Simple Types
- Complex Types
- Namespaces
- Instances and Schemas

Data Types

- Complex vs. Simple Types
  - Simple types cannot have children or attributes
    - <size>10</size>
    - <comment>This is an example.</comment>
    - <availableSizes>10 large 2</availableSizes>
  - Complex types can have child elements and attributes
    - <size system="US-DRESS">10</size>
    - <availableSizes>
      <size>10</size>
      <size>2</size>
    - Attributes have simple types not complex types
Data Types

• Named vs. Anonymous types
  – Named types are global (at the top level), uniquely named
  – Anonymous types are unnamed and defined within an element or attribute definition
    <xsd:element name="size">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="2" />
          <xsd:maxInclusive value="18" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>

• Type hierarchy
  – Simple types and complex types can be derived (inherited) from other types
  – Derived types can substitute for ancestors in instances

Simple Types

• Built-in types
  – Boolean
  – String
  – URIs
  – Numeric types
  – Time types
  – XML types
  – No money types. However, these can be derived

• Restricting simple types
  – Facets  • length  • maxInclusive
    • minLength  • maxExclusive
    • maxLength  • minInclusive
    • pattern  • minExclusive
    • enumeration  • totalDigits
    • whiteSpace  • fractionDigits

Not all facets apply to all types.
Simple Types

• List and Union types
  – An xsd:list child element derives a type as a white space separated list of base type instances
  – An xsd:union child element derives by combining legal values from multiple base types

Complex Types

• Content types
  – Contents of an element are the character data and child elements between its tags
  – Four types:
    • Simple (character data)
      <size system="US-DRESS"> 10 </size>
    • element (child element)
      <product>
        <number>55555</number>
        <size>10</size>
      </product>
    • Mixed (character data and child element)
      <letter>Dear <custName>John Doe</custName> … </letter>
    • Empty (no content)
      <color value="blue" />
Complex Types

- Content Models
  - Order and Structure of child elements
    - sequence
      - sequence requires each child element it specifies to appear in the specified order
      - The sequence can have minOccurs and maxOccurs attributes that repeat each sequence zero to any given number of times
    - choice
      - choice requires exactly one of a group of specified elements to appear
      - The choice can have minOccurs and maxOccurs attributes that adjust this from zero to any given number
    - all
      - All requires all the child elements to appear 0 or 1 times, in any order

```xml
<xsd:complexType name="ProductType">
  <xsd:sequence>
    <xsd:element name="number" type="xsd:integer" />
    <xsd:choice minOccurs="0" maxOccurs="3">
      <xsd:element name="size" type="SizeType" />
      <xsd:element name="color" type="ColorType" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="effDate" type="xsd:date" />
</xsd:complexType>
```
**Complex Types: Derived types**

- **Restriction**
  
  ```xml
  <xsd:simpleType name="SizeType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="2"/>
      <xsd:maxInclusive value="18"/>
    </xsd:restriction>
  </xsd:simpleType>
  
  **Extension**
  
  ```xml
  <xsd:complexType name="ShirtType">
    <xsd:complexContent>
      <xsd:extension base="ProductType">
        <xsd:sequence>
          <xsd:element name="color" type="ColorType"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  ```

**Schemas and Namespaces**

- Elements and attributes that are in namespaces are called qualified
- All unprefixed attributes are unqualified
- All prefixed elements are qualified
- Unprefixed elements may or may not be qualified. They are qualified if they are in a default namespace.
- Each schema has a target namespace
- Each schema can define elements in attributes in its target namespace
- A schema can also define unqualified attributes of elements in its target namespace.
- A schema can also define unqualified child elements of elements in its target namespace. Unqualified child elements are called local elements. This is a very bad idea!
- A schema may not define elements and attributes in namespaces other than the target namespace; i.e., for each namespace there must be at least one schema.
- Schemas can reference global elements and attributes defined in other schemas by importing the schema with `xsd:import` and referencing the global elements and attributes defined therein.
Target Namespace

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2002/XMLSchema"
    targetNamespace="http://example.org/prod"
    xmlns:prod = "http://example.org/prod" >
    <xsd:element name="product" type="prod:ProductType"/>
    <xsd:complexType name="ProductType">
        <xsd:sequence>
            <xsd:element name="number" type="xsd:integer" />  
            <xsd:element name="size" type="prod:SizeType" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="SizeType">
        <xsd:restriction base="xsd:integer"> 
            <xsd:minInclusive value="2" />  
            <xsd:maxInclusive value="18" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Default Namespace

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2002/XMLSchema"
    targetNamespace="http://example.org/prod"
    xmlns = "http://example.org/prod" >
    <xsd:element name="product" type="ProductType"/>
    <xsd:complexType name="ProductType">
        <xsd:sequence>
            <xsd:element name="number" type="xsd:integer" />
            <xsd:element name="size" type="SizeType" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="SizeType">
        <xsd:restriction base="xsd:integer"> 
            <xsd:minInclusive value="2" />  
            <xsd:maxInclusive value="18" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```
Instances

- Instance is a document conforming to a schema

  <product>
  <number>55555</number>
  <size>10</size>
  </product>

Annotations

- Describe the structure of XML instances
- `xsd:documentation` specifies human-readable information
- `xsd:appinfo` specifies application information

```xml
  <xsd:schema xmlns:xsd = "http://www.w3.org/2002/XMLSchema"
               xmlns:doc = "http://example.org/doc">
  <xsd:element name="product" type="prod:ProductType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        source=http://example.org/prod.html#product>
      <doc:description>These elements represent a product.</doc:description>
    </xsd:documentation>
  </xsd:element>
  </xsd:schema>
```
Advanced Features

• Identity constraints
• Substitution groups
• Redefinition
• Reusable groups

SOAP
What is SOAP?

- SOAP is a simple, lightweight XML protocol for exchanging structured and typed information on the Web
- Overall design goal: KISS
  - Can be implemented in a weekend
  - Stick to absolutely minimum of functionality
- Make it Modular and Extensible
  - No application semantics and no transport semantics
  - Think “Web based datagram”

SOAP's Four Parts:

- An extensible envelope expressing *(mandatory)*
  - what features and services are represented in a message;
  - who should deal with them,
  - whether they are optional or mandatory.
- A set of encoding rules for data *(optional)*
  - Exchange instances of application-defined data types and directed graphs
  - Uniform model for serializing non-syntactic data models
- A Convention for representation RPC *(optional)*
  - How to make calls and responses
- A protocol binding to HTTP *(optional)*
SOAP Example in HTTP

```
POST /Accounts/Henrik HTTP/1.1
Host: www.webservicebank.com
Content-Length: nnnn
Content-Type: text/xml; charset=UTF-8
SOAPAction: "Some-URI"

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP:Header>
    <t:Transaction xmlns:t="some-URI" SOAP:mustUnderstand="1">
      5
    </t:Transaction>
  </SOAP:Header>
  <SOAP:Body>
    <m:Deposit xmlns:m="Some-URI">
      <m:amount>200</m:amount>
    </m:Deposit>
  </SOAP:Body>
</SOAP:Envelope>
```

SOAP-HHTTP Binding
HTTP Request
SOAP Body
SOAP Header
SOAP Envelope

SOAP is a Protocol!

- What does this mean?
  - It is not a distributed object system
  - It is not an RPC system
  - It is not even a Web application

- Your application decides what your application is!
  - You can build a tightly coupled system
  - …or…
  - You can build a loosely coupled system

- Why does this matter?
  - It means that you have to think about how you design your application
Myths about SOAP

• SOAP is for RPC only! No…
  – SOAP doesn't define or imply a programming model
  – It can be used for messaging, RPC, Distributed object systems etc.
• SOAP is for HTTP only! No…
  – SOAP can be used in combination with any protocol that can carry a
    SOAP envelope
  – SOAP currently defines bindings to HTTP and HTTP Extension
    Framework - others can be added
• SOAP is request/response! No…
  – SOAP doesn't define a message exchange pattern
  – Can be defined in SOAP or inherited from protocol binding

The *Amtrak* Message Model

• A train is a SOAP message
  – It starts in the departure city, stops at a set of intermediate cities,
    and stops at the destination city
• A city is a SOAP processor
  – Ensures that passengers get on and off
  – Passenger tickets are verified
• A passenger is a SOAP feature or service
  – A passenger can get on an off at any stop
  – Passengers can mix in arbitrary ways
• Message model can be put together to form arbitrary
  message graphs
The SOAP Envelope

• A SOAP envelope defines a SOAP message
  – Basic unit of exchange between SOAP processors
  – Highly flexible

• SOAP messages are one-way transmissions
  – From sender through intermediaries to receiver
  – Often combined to implement patterns such as request/response

• Messages are routed along a "message path"
  – Allows for processing at one or more intermediate nodes in addition
to the ultimate destination node.
  – A node is a SOAP processor and is identified by a URI

SOAP Headers

• Allows for modular addition of features and services
  – Open-ended set of headers
    • Authentication, privacy, security etc. etc.
  – Can address any SOAP processor using the "actor" attribute
  – Can be optional/mandatory using the "mustUnderstand" attribute
Semantics of Headers

- Contract between sender and recipient
  - Recipient is described by "actor" attribute
- Allows for different types of negotiation:
  - Take it or leave it
  - Let's talk about it
- And for different settings:
  - Server dictated
  - Peer-to-peer
  - Dictated by third party

actor Attribute

- The "Actor" attribute is a generalization of the HTTP Connection header field
  - Instead of hop-by-hop vs. end-to-end, the actor attribute can address any SOAP processor because it is a URI
  - Special cases:
    - "next hop" has a special URI assigned to it
    - "end" is the default destination for a header
mustUnderstand Attribute

- The "mustUnderstand" is the same as the "mandatory" in the HTTP Extension Framework
  - Requires that the receiving SOAP processor must accept, understand and obey semantics of header or fail
  - This allows old applications to gracefully fail on services that they do not understand

SOAP Body

- Special case of header
  - Default contract between sender and ultimate recipient
  - Defined as a header with attributes set to:
    - Implicit mustUnderstand attribute is always "yes"
    - Implicit actor attribute is always "the end"
SOAP Fault Example

• A SOAP message containing an authentication service:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
<SOAP:Header>
  <m:Authentication xmlns:m="http://www.auth.org/simple">
    <m:credentials>Henrik</m:credentials>
  </m:Authentication>
</SOAP:Header>
<SOAP:Body>
  ... body goes here ...
</SOAP:Body>
</SOAP:Envelope>
```

SOAP Fault Example… 2

• …results in a fault because the credentials were bad:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
<SOAP:Header>
  <m:Authentication xmlns:m="http://www.auth.org/simple">
    <m:realm>Magic Kingdom</m:realm>
  </m:Authentication>
</SOAP:Header>
<SOAP:Body>
  <SOAP:Fault>
    <SOAP:faultcode>SOAP:Client</SOAP:faultcode>
    <SOAP:faultstring>Client Error</SOAP:faultstring>
  </SOAP:Fault>
</SOAP:Body>
</SOAP:Envelope>
```
WSDL

- An XML-based grammar for describing the capabilities of Web Services
- Extensible
- Jointly developed by Microsoft and IBM
  - Convergence of SDL/SCL and NASSL
- Similar in concept to IDL, but it’s not IDL
  - IDL is platform dependent
  - WSDL is platform independent
Service Definition Elements

- **types**, which provides data type definitions used to describe the messages exchanged.
- **message**, which represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
- **portType**, which is a set of abstract operations. Each operation refers to an input message and output messages.
- **binding**, which specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
- **port**, which specifies an address for a binding, thus defining a single communication endpoint.
- **service**, which is used to aggregate a set of related ports.

WSDL Document Example

```xml
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wSDL"
 xmlns:tns="http://example.com/stockquote.wSDL"
 xmlns:xsd1="http://example.com/stockquote.xsd"
 xmlns:soap="http://schemas.xmlsoap.org/soap/"
 xmlns="http://schemas.xmlsoap.org/soap/">

```
Types

<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all>
          <element name="price" type="float"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>

Messages

<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>
Port Types

<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>

Port Types

• One-way. The endpoint receives a message.
  – <input message="tns:GetLastTradePriceInput"/>

• Request-response. The endpoint receives a message, and sends a correlated message.
  – <input message="tns:GetLastTradePriceInput"/>
  – <output message="tns:GetLastTradePriceOutput"/>

• Solicit-response. The endpoint sends a message, and receives a correlated message.
  – <output message="tns:GetLastTradePriceOutput"/>
  – <input message="tns:GetLastTradePriceInput"/>

• Notification. The endpoint sends a message.
  – <output message="tns:GetLastTradePriceOutput"/>
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http/">
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="encoded" namespace="http://example.com/stockquote" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded" namespace="http://example.com/stockquote" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </soap:binding>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort"
    binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>

</definitions>
UDDI

• Universal Description, Discovery and Integration
  – Developed by UDDI.org
  – Will be submitted to a standards body
• A registry for Web services
• Helps you find a Web service and its description (WSDL)
  – Search by business
  – Search by service type
• UDDI Working Group
  • Accenture
  • Ariba
  • Commerce One
  • Compaq
  • Fujitsu
  • Hewlett-Packard
  • i2
  • Intel
  • IBM
  • Microsoft
  • Oracle
  • SAP
  • Sun
  • Verisign
• UDDI Advisory Group -- Any one can join
Public Registry Operations

- Peer nodes (websites)
  - Companies register with any operator
  - Registrations replicated on a daily basis
  - Complete set of “registered” records available at all operator nodes
- Common set of SOAP APIs supported by all operators
- Compliance enforced by business contract

UDDI Private Registries

- Community-based registry
  - Within a business
  - Within a trusted community
- Extended UDDI to support
  - Privacy
  - Security
  - Data integrity
  - Reliability
  - Manageability
  - Richer query capabilities
UDDI Overview

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

4. Business uses this data to facilitate easier integration with each other over the Web

What’s in the Registry?

- Standards bodies, programmers and businesses register information about their service types, including specifications, taxonomies, etc

- Businesses register public information about themselves and the services they offer

Service Types
- a.k.a. “tModels”

- White Pages
- Yellow Pages
- Green Pages
White Pages

- Information about a business
- Business name
- Text description(s)
  - List of multi-language text strings
- Contact info
  - Names, addresses, phone numbers, fax numbers, web sites…
- Identifiers
  - List of identifiers
  - DUNS, Thomas Register, others

Yellow Pages

- Business/Service categorizations
- Three standard taxonomies in V1
  - Industry codes: NAICS (US govt.)
  - Product/services: UN/SPSC (ECMA)
  - Location: geographical taxonomy
- Implemented as name-value pairs
  - Allows any valid taxonomy identifier to be attached to the business white page
- V2 supports custom taxonomies
Green Pages

Technical information

• How to invoke services
  – References to specifications for web services
  – Support for pointers to various file and URL based discovery mechanisms if required
• Programming/platform/implementation agnostic

UDDI Data Model
tModel

- Represents a technical model
  - Service type or specification type
    - The “green pages”
  - Categorization
    - Used for “yellow pages”
  - Identification
    - Used for “white pages”
- Generated by UDDI upon <save>
  - tModelKey (UUID unique identifier)

businessEntity

- Represents a business
- “White pages” information:
  - Name, address, contact info
  - IdentifierBag (D-U-N-S, Thomas Register)
- “Yellow pages” information:
  - CategoryBag (list of categorization tModels)
- Generated by UDDI upon <save>:
  - businessKey (UUID unique identifier)
  - authorizedName (security token)
  - discoveryURL (points to businessEntity)
businessService

- Represents a service
  - Contained within businessEntity
- Description
- “Yellow pages” information:
  - CategoryBag
- Generated by UDDI upon <save>
  - serviceKey (UUID unique identifier)

bindingTemplate

- Specific service information
  - Contained within businessService
- Description
- “Green pages” information
  - Access point
  - tModel reference
  - OverviewDoc
- Generated by UDDI upon <save>
  - bindingKey (UUID unique identifier)
Putting It All Together…

- businessEntity
  - businessService
    - bindingTemplate
      - InstanceDetails
    - categoryBag
      - keyedReference
    - identifierBag
      - keyedReference
  - Rosetta-Net
    - BASDA
      - Simple.Buy
        - Schemas, Interchange specification
  - tModels
    - GEO Code
    - NAICS
    - Other...
    - DUNS Numbers
    - Thomas Registry ID

- ebXML
What is ebXML

- ebXML = Electronic Business XML
- Global Standard for electronic business
- ebXML enables anyone, anywhere to do business with anyone else over the Internet
- Out-of-the-box technical interoperability
- Unambiguous commercial interoperability
  - Explicitly specified and “executable” business processes
- Service-based business process architecture
- Enable the evolution of many new business models and patterns
- Complementary to existing B2B initiatives
  (UDDI, RosettaNet, TradeXchange, etc.)

An end-to-end B2B XML Framework

Sponsored by …

(United Nations Center For Trade Facilitation And Electronic Business) (Organization for the Advancement of Structured Information Standards)

Hundreds of participants from all over the world Businesses, governments, academia, institutions
**ebXML Vision**

- A global electronic market place where enterprises of any size, anywhere can:
  - Find each other electronically
  - And conduct business
    - Using XML messages
    - According to standard business process sequences
    - With clear business semantics
    - According to standard or mutually agreed trading partner protocol agreements
    - Using off the shelf purchased business applications

---

**B2B Collaboration**

- B2B collaboration requires more than just an XML protocol and a service registry
- You have to deal with
  - Business semantics
  - Negotiating terms and conditions
  - Interoperability
  - Security and Privacy
  - Reliability
- ebXML provides concrete specifications to enable dynamic B2B collaborations
**Usage Example**

1. Request ebXML specifications
2. ebXML specifications detail
3. Register scenarios and implementation details
4. Register company business profile
5. Confirm profile and scenarios accepted
6. Query about Company X
7. Company X’s Profile
8. Submit CPA
9. Accept CPA
10. Request Company X’s Scenario
11. Company X’s Scenario

**Company Profile**

- **Collaboration Protocol Profile**
  - Defined using ebXML Specification Schema
  - Concrete specification of your ebusiness offerings
    - Business scenarios you support
    - Service interfaces you implement
    - Document formats exchanged
    - Technical requirements/options (protocols, security, reliability)

- **Composed of**
  - Business process models
  - Information models
  - Context rules
Business Scenarios

• Often defined by Industry Groups
  – Standard business scenarios remove the need for prior agreements among trading partners

• Business Process Model
  – Interactions between parties
  – Sequencing of interactions
  – Documents exchanged in each interaction

• Information Model
  – Document definition
  – Context definition
  – Context rules

Core Components

• Reusable low-level data structures
  – e.g., party, address, phone, date, currency
  – Context-sensitive

• Single, consistent lexicon

• Used to define business process and information models

• Facilitates interoperability between disparate systems
Context Affects Process

- Industry Sector
- Product
- Business process
- Geo-political region
- Official constraints
  - Legislative
  - Standards
  - Good practice
  - Contractual
ebXML Specification Schema

Registering Your Business

- Register your business in an ebXML Registry
  - Index to all information in the repository
  - Rich query facility
- Store specifications in an ebXML Repository
  - CPP
  - Schemas
  - Process models
  - Core components
  - Classification and categorization schemes
  - Arbitrary objects and code
Negotiating an Agreement

- Find registry and search for partners
- Examine CPP
- Ascertain compatibility of business process and technical specifications
- Stipulate your “rules of engagement”
- Produce Collaboration Protocol Agreement
  - Conditions under which two partners will conduct business transactions together

Business Service Interface

- Implements the CPA, supporting dynamic integration
- Not yet specified
  - Hand-crafted for the moment
- Enables one Party to converse with the other Party using the ebXML Message Service
ebXML Message Service

- Reliable, secure XML messaging service
  - Enforces the rules of engagement in CPA
- Transport independent
- Extends SOAP Messages with Attachments (SwA)
  - Reliability framework
  - Security framework
  - Manifest, trace, and delivery options

Delivery Options

- Communications models
  - Synchronous or asynchronous
  - Request/response
  - Fire and forget
  - Multipart message delivery
- Reliability options:
  - Best effort
  - Once and only once
Security

- Identification
- Authentication
- Authorization
- Privacy
- Integrity
- Non-repudiation
- Logging

**ebXML Message Structure**

- Communication Protocol Envelope (HTTP, SMTP, etc.)
- SOAP Messages with Attachments MIME Envelope
  - MIME Part
    - SOAP-ENV:Envelope
      - SOAP-ENV:Header
        - eb:MessageHeader
        - eb:TraceHeaderList
        - Other:etc...
      - SOAP-ENV:Body
        - eb:Manifest
        - eb:etc...
        - Other:etc...
    - MIME Part
      - Payload
  - ebXML Header Information
  - ebXML Message Service Handler control data
  - Payload Container(s)
Summary of Components

- Registry and Repository
- Core Components
- ebXML Specification Schema
  - Business Process Model
  - Information Model
- CPP/CPA
- Message Service

SOAP and UDDI

- Obviously useful, but they don’t constitute an end-to-end B2B framework
- No support for business models or negotiating business agreements
- No Quality of Service facilities
- Complementary not competitive to ebXML
  - SOAP provides messaging foundation
  - UDDI helps you find ebXML services
  - ebXML Repository stores service specifications
Business Process and Workflow Specification

Business Processes

• Model
  – Functional/Business logic
  – Event driven logic
  – Collaboration logic
  – Interaction logic
  – Integration logic

• Building applications as processes gives following benefits
  – Flexibility to Change
  – Reusability of Components/Methodology
  – Integration capability with disparate application
  – Application Development Scalability
  – Application Deployment Scalability – component distribution, manageability, upgradeability

\[ \text{Process} = \text{Workflow} + \text{Rules} \]
Business Process Specification

• Capture: design, development, deployment, execution and optimization
• Currently, no clear standard
  – BPML from Business Process Management Initiative
  – XLANG from Microsoft
  – WSFL from IBM
  – WSCI for Process Interface Specification from SAP, BEA, SUN, Intalio

What is BPML?

• Abstract Model and Grammar for expressing business processes
  – It does not provide any application semantics

• Used for variety of purposes
  – Definition of Enterprise Business Processes
  – Definition of Complex Web Services
  – Definition Multi-Party Collaborations
Packages

- Package is a collection of BPML definitions and declarations
  - BPML definitions and declarations can reference definitions expressed in
    other languages. E.g. XSD type definitions, WSDL service definitions

  - `<package`
  - `name = NCName`
  - `targetNamespace = anyURI>`
  - `Content: (documentation?, import*,`
  - `(connect | correlation | locator | process |`
  - `property | selector | (extension element))+)`
  - `</package>`

- Import statements
  - To import definitions and declarations from different documents and
    namespaces

  - `<!— Import BPML definitions from this namespace/document -->`
  - `<import namespace="http://www.bpmi.org/examples/import/process"`
  - `location="http://www.bpmi.org/examples/import/process.bpml"/>`

Activities

- Activities perform a specific function within a process
  - E.g. invoking a service or another process

  - `<{activity type}`
  - `name = NCName`
  - `{other attributes}>`
  - `Content: (documentation?, {other elements}*)`
  - `</{activity type}>`

  - Atomic activity is an elementary unit of work that cannot be further
    decomposed
  - Complex activity is composed of other activities

- Activity Context
  - Activity is always executed within a context
  - Context distinguishes between multiple instances of same activity

- Activity Set
  - collection of one or more activities that execute in the same context
Activity Types

• Action is an atomic activity
  • <action>
    – name = NCName
    – portType = QName
    – operation = NCName
    – {extension attribute}>
    – Content: (documentation?, correlate*, locate?, call?, output*)
  • </action>
  – Correlate
    • Used for identifying process instance based on data passed in a message. e.g. purchase order number
  – Locate
    • Used if a service instance must be identified
  – Call
    • An action can perform set of activities by calling a process
  – Output
    • Construct information and provide it to output message

Activity Types

• All executes activities within the activity set in non-sequential order
• Assign a new value to a property in the current context
• Call instantiates a process and waits for it to complete
• Choice selects and executes one activity set in response to a triggered event
• Compensate performs compensation for all instances of the named transaction
• Delay expresses the passage of time
• Empty – used in places where no work is to be performed
• Fault – triggers a fault within the current context
Activity Types

- Foreach - performs all the activities within the activity set repeatedly, once for each item in the list
- Join - waits for instances of process to complete.
- Sequence - performs all the activities within the activity set in sequential order
- Spawn – instantiates a process
- Switch - selects and executes one activity set based on the evaluation of one or more conditions
- Until - executes all the activities within the activity set repeatedly, one or more times, based on the truth value of a condition
- While - executes all the activities within the activity set repeatedly, zero or more times, based on the truth value of a condition

Processes

- Progressively continuing procedure consisting of a series of controlled activities
  - <process
  -    name = NCName
  -    instantiation = message | other : message
  -    scope = public | private : private>
  -    Content: (documentation?, parameters*, context?, {any activity}+)
  -  </process>

- Process can be instantiated on the receipt of messages
  - Receipt of a message
  - Receipt of all messages
  - Receipt of choice (any one) message

- Process can also be instantiated by other activities
  - Spawned or called from another process
  - Called in response to a system event
Context

- Contexts are composed hierarchically
  - Current context contains parents context and its local declarations
  - Local declarations are not available to parent or siblings
  - Local declarations hide the declarations with same name in parent context

- `<context>`
  - Content: `((connect | process | property | {extension element})*,
  exception?, transaction?, completion?)`
  - `</context>`

- Types of Local Declarations
  - Property
  - Exception
  - Process
  - Transaction
  - Connector

Properties

- Property is a name value
  - Properties are referenced by their fully qualified names
  - Values are not restricted to a particular schema, and maybe of any type

- `<property`
  - `name = QName`
  - `select = XPath>`
  - Content: `documentation?, (extension element) | value)?)`
  - `</property>`

- Example illustrating a property declaration
  - `<bpmn:property name="tns:someDate">`
  - `<bpmn:value>2001-01-29</bpmn:value>`
  - `</bpmn:property>`

- Property can be changed by direct or indirect result of an activity
Selectors

- Selectors are used to extract values from the contents of a message and assign it to a property
  
  - `<selector`
  - `property = QName`
  - `element = QName`
  - `type = QName`
  - `message = QName`
  - `part = NCName`
  - `select = XPath`
  - `{extension attribute}>`
  - `Content: (documentation?, {extension element}?)`
  
- Example Selector that calculates the tns:orderTotal property from the tns:poMessage message

  - `<bpmn:selector property="tns:orderTotal" element="tns:lineItems"`
  - `select="sum(lineItems/lineItem/(quantity * price))"/>

Connecting Services

- Global model provides a views of how processes interact
  - Processes are loosely coupled and interact through messages

  - `<model`
  - `name = NCName`
  - `processes = listOfQName>
  - `Content: (documentation?, connect+)`
  
  - </model>

- Connectors express the interaction between processes
  - Provide a link between the operations performed between processes

  - `<connect`
  - `name = NCName`
  - `type = direct | broadcast : direct>
  - `Content: (documentation?,operation[2],{extension element}?)`
  
  - </connect>

  - `<operation`
  - `portType = QName`
  - `name = NCName`
  - `{extension attribute}/>"
Exceptions

- Event handler defines the triggering event and the activity set to be performed when the event occurs
  - `<exception>`
  - Content: (onMessage | onTimeout | onFault)+
  - `</exception>`

- 3 types of events
  - onMessage event handler responds to an input message
    - `<onMessage>`
    - Content: (documentation?, action, context?, (any activity)+)
    - `</onMessage>`
  - onTimeout event handler responds to a time-out event
    - `<onTimeout property = QName type = duration | dateTime : duration
      reference = QName | QName@start | QName@end>`
    - Content: (documentation?, context?, {any activity}+)
    - `</onTimeout>`
  - onFault event handler responds to a fault.
    - `<onFault code = QName>`
    - Content: (documentation?, context?, {any activity}+)
    - `</onFault>`

Transactions

- Transactions allow multiple activities to be treated as a single unit of work, providing a guarantee of consistency and reliability
  - `<transaction`
    - `name = NCName`
    - `type = atomic | open : atomic`
    - `participation = supports | always | never : never`
    - `retries = nonNegativeInteger : 0>`
    - `Content: (compensation?)`
  - `</transaction>`

- Atomic Transactions
  - All activities performed as part of the transaction behave as a single unit of work
  - If the transaction cannot complete successfully, it will rollback to the state before the beginning of the transaction
  - In order to provide an all-or-nothing guarantee, an atomic transaction must exhibit the ACID (Atomicity, Consistency, Isolation, Durability) properties

- Open Transactions
  - These can be used for long-lived transactions that cannot complete in a short time span
  - Resources are acquired for short periods of time and then released
  - Isolation requirement is relaxed and arbitrary levels of nesting is allowed
Transactions

• Recovery
  – Backward recovery guarantees that in the event of the transaction aborting, the process will return to the consistent state that existed prior to the beginning of the transaction
  – Atomic transactions provide automatic backward recovery
  – Forward recovery guarantees that in the event of system failure, the transaction state can be restored to a consistent state and its execution can continue reliably past the point of failure
  – Forward recovery only applies to open transactions
  – Atomic activities and atomic transactions always perform backward recovery in the event of system failure

• Compensation – for backward recovery on open transaction
  – Logic for reverting the effects of a completed activity or transaction
    • <compensation>
    • Content: (documentation?, parameter*, context?, {any activity}+)
    • </compensation>

RosettaNet
The RosettaNet Vision, Mission

- **Vision:** The Leader in global e-business standards
- **Mission:** RosettaNet drives collaborative development and rapid deployment of internet-based business standards, creating a common language and open e-business processes that provide measurable benefits and are vital to the evolution of the global, high-technology trading network.

Information Exchange Standards

<table>
<thead>
<tr>
<th>Process-centric</th>
<th>Message-centric</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Real-time</td>
<td>- Batch</td>
</tr>
<tr>
<td>- 100% of B2B processes</td>
<td>- 10% of B2B processes</td>
</tr>
<tr>
<td>- Internet-enabled</td>
<td>- VAN-enabled</td>
</tr>
<tr>
<td>- XML</td>
<td>- X.12/EDIFACT/JECA LS</td>
</tr>
<tr>
<td>- Global</td>
<td>- Regional</td>
</tr>
<tr>
<td>- All businesses</td>
<td>- Large businesses</td>
</tr>
<tr>
<td>- Standard industry dictionaries</td>
<td>- Custom industry dictionaries</td>
</tr>
</tbody>
</table>
Partner-to-Partner
Electronic Business Interface

Company A
Company Specific processing
SAP
ERP
Translate from RosettaNet standards to Company A system data set.

Internet & XML

Company B
Company Specific processing
I2
APS
Translate from RosettaNet standards to Company B system data set.

Business Processes

Private process (Company-specific)
Process
Receive PO Request
Select Supplier
Generate RFQ
Send RFQ
Select RFQ Response
Send PO
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Customer
Receive PO
Receive PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

CRM
SCM
ERP

Private process (Company-specific)
Process Sales Order
Receive PO
Check Customer
Check Credit
Check Availability
Create Sales Order
Send PO
Send PO Response
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Supplier
Receive PO
Send PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

Private process (Company-specific)
Process
Receive PO Request
Select Supplier
Generate RFQ
Send RFQ
Select RFQ Response
Send PO
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Customer
Receive PO
Receive PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

Private process (Company-specific)
Process
Receive PO Request
Select Supplier
Generate RFQ
Send RFQ
Select RFQ Response
Send PO
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Supplier
Receive PO
Send PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

Private process (Company-specific)
Process
Receive PO Request
Select Supplier
Generate RFQ
Send RFQ
Select RFQ Response
Send PO
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Customer
Receive PO
Receive PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

Private process (Company-specific)
Process
Receive PO Request
Select Supplier
Generate RFQ
Send RFQ
Select RFQ Response
Send PO
Send PO Response Acknowledge
Close

Public process (Standard)
Send PO Supplier
Receive PO
Send PO Acknowledge
Send PO Response
Receive PO Response Acknowledge
Send PO Response Acknowledge
Close

CRM
SCM
ERP
RosettaNet Business Process Architecture

- Partner Interface Process™ (PIP™)
- RosettaNet Dictionaries
- RosettaNet Implementation Framework (RNIF) Core
  (Messaging Services)
**Partner Interface Process™ (PIP™)**

- Depict activities, decisions and interactions that fulfill a business transaction
- Specify structure and format of business document payloads
- Organized by clusters and segments

---

**RosettaNet Business & Technical Dictionaries**

- Ensures consistent information exchange during PIP™ execution
- Technical dictionary (form, fit, function)
  - Specifies common product properties
- Business dictionary
  - Specifies common partner properties
  - Enables partners to identify one another
- Shares common standards
  - Partner Identification
  - Product Identification
RosettaNet Implementation Framework (RNIF) Core

- Defines RosettaNet Object (RNO)
- Specifies how to transport RosettaNet Object between trading partners’ network applications
- Version 2.0 features and benefits:
  - HTTP and SMTP transfer protocols better support for e-marketplaces
  - Support for .pdf, .gif files - can send complex documents
  - Support for S/MIME v.2 - greater security, privacy and authentication

RosettaNet Process Model

RosettaNet Partner Interface Process™ Standards
### RosettaNet: Clusters Segments and PIPs

#### Cluster 0: RosettaNet Support
- **Segment 0A:** Administrative
  - **PIP 0A1:** Notification of Failure

#### Cluster 1: Partner, Product, & Service Review
- **Segment 1A:** Partner Review
  - **PIP 1A1:** Request Account Setup
  - **PIP 1A2:** Maintain Account
- **Segment 1B:** Product and Service Review
  - **PIP 1B1:** Manage Product Information Subscriptions

#### Cluster 2: Product Information
- **Segment 2A:** Preparation for Distribution
  - **PIP 2A1:** Distribute New Product Information
  - **PIP 2A2:** Query Product Information
  - **PIP 2A3:** Query Marketing Information
  - **PIP 2A4:** Query Sales Promotion & Rebate Information
  - **PIP 2A5:** Query Technical Information
  - **PIP 2A6:** Query Product Lifecycle Information
  - **PIP 2A7:** Query Product Discontinuation Information
  - **PIP 2A8:** Distribute Product Stock Keeping Unit (SKU)
  - **PIP 2A9:** Query EC Technical Information
  - **PIP 2A12:** Distribute Product Master

#### Cluster 3: Order Management
- **Segment 3A:** Quote and Order Entry
  - **PIP 3A1:** Request Quote
  - **PIP 3A2:** Query Price and Availability
  - **PIP 3A3:** Request Shopping Cart Transfer
  - **PIP 3A4:** Request Purchase Order
  - **PIP 3A5:** Query Order Status
  - **PIP 3A6:** Distribute Order Status
  - **PIP 3A7:** Notify of Purchase Order Update
  - **PIP 3A8:** Request Purchase Order Change
  - **PIP 3A9:** Request Purchase Order Cancellation
  - **PIP 3A10:** Notify of Quote Acknowledgement
  - **PIP 3A11:** Notify of Purchase Order Information
  - **PIP 3A14:** Distribute Planned Order

#### Segment 2B: Product Change Notification
- **PIP 2B1:** Change Basic Product Information
- **PIP 2B2:** Change Marketing Information
- **PIP 2B3:** Change Sales Promotion & Rebate Information
- **PIP 2B4:** Change Product Technical Information
- **PIP 2B5:** Change Product Lifecycle Information

#### Segment 2C: Product Design Information
- **PIP 2C1:** Distribute Engineering Change Status
- **PIP 2C2:** Request Engineering Change
- **PIP 2C3:** Distribute Engineering Change Response
- **PIP 2C4:** Request Engineering Change Approval
- **PIP 2C5:** Notify of Engineering Change Order
- **PIP 2C6:** Notify of Engineering Change Implementation

#### Segment 3B: Transportation and Distribution
- **PIP 3B1:** Distribute Transportation Projection
- **PIP 3B2:** Notify of Advance Shipment
- **PIP 3B3:** Distribute Shipment Status
- **PIP 3B4:** Query Shipment Status
- **PIP 3B5:** Request Shipment Change
- **PIP 3B6:** Notify of Shipments Tendered
- **PIP 3B13:** Notify of Shipment Confirmation
- **PIP 3B18:** Notify of Shipment Documentation

#### Segment 3C: Returns and Finance
- **PIP 3C1:** Return Product
- **PIP 3C2:** Request Financing Approval
- **PIP 3C3:** Notify of Invoice
- **PIP 3C4:** Notify of Invoice Reject
- **PIP 3C5:** Notify of Billing Statement
- **PIP 3C6:** Notify of Remittance Advice

#### Segment 3D: Product Configuration
- **PIP 3D8:** Distribute Work in Process
- **PIP 3D9:** Query Work in Process
RosettaNet: Clusters Segments and PIPs

Cluster 4: Inventory Management
Segment 4A: Collaborative Forecasting
- PIP 4A1: Notify of Strategic Forecast
- PIP 4A2: Notify of Embedded Release Forecast
- PIP 4A3: Notify of Threshold Release Forecast
- PIP 4A4: Notify of Planning Release Forecast
- PIP 4A5: Notify of Forecast Reply
Segment 4B: Inventory Allocation
- PIP 4B2: Notify of Shipment Receipt
Segment 4C: Inventory Reporting
- PIP 4C1: Distribute Inventory Report

Cluster 5: Marketing Information Management
Segment 5C: Design Win Management (EC)
- PIP 5C1: Distribute Product List
- PIP 5C2: Request Design Registration
- PIP 5C3: Create Design Win
- PIP 5C4: Distribute Registration Status
- PIP 5C5: Query Registration Status
Segment 5D: Ship from Stock and Debit (EC)
- PIP 5D1: Request Ship from Stock and Debit Authorization
- PIP 5D2: Notify of Blanket Ship from Stock and Debit Authorization
- PIP 5D3: Distribute Open Ship from Stock and Debit Authorization Status
- PIP 5D4: Query Ship from Stock and Debit Authorization Status
- PIP 5D5: Create Ship from Stock and Debit Claim Status
- PIP 5D6: Notify of Ship from Stock and Debit Claim Status

Cluster 6: Service and Support
Segment 6C: Technical Support and Service Management
- PIP 6C1: Query Service Entitlement
- PIP 6C2: Request Warranty Claim

Cluster 7: Manufacturing
Segment 7C: Distribute Manufacturing Information
- PIP 7C6: Distribute Product Quality Event Data

Agenda

- Evolution of eBusiness
- eBusiness Applications
- eBusiness Architecture
- eBusiness Standards
- Discussion
Discussion

- How do all these standards fit together?
- How does ebXML fit into web services?
- How does ebXML relate to RosettaNet?
- What about workflow standards?
  - WSCI, BPML, XLANG, WSFL
- How does database technology fit here?

Questions?