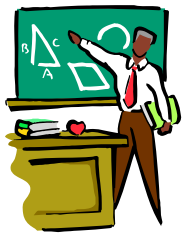
Introduction to Enterprise Service Bus and WebSphere Message Broker

WebSphere software

Agenda

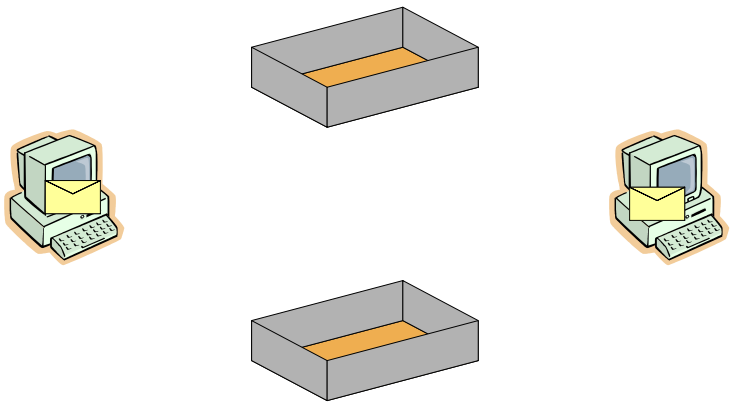
- Enterprise Service Bus Concept
- Message Broker
  - Message Flows
  - Message Nodes
  - The Logical Message Model
- Message Broker Components



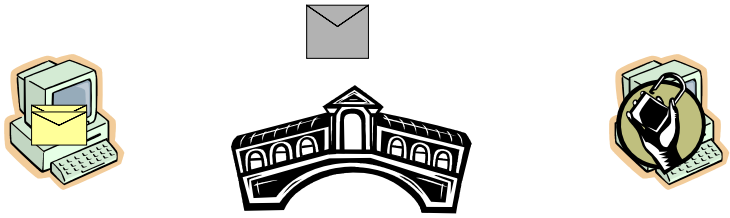
Objectives

- Have an overall understanding of the role of an Enterprise Service Bus (ESB)
- Become familiar with the WebSphere Message Broker's (Advanced ESB) primary functions
- Understand the overall steps used to build and deploy a WMB « flow »

Messaging

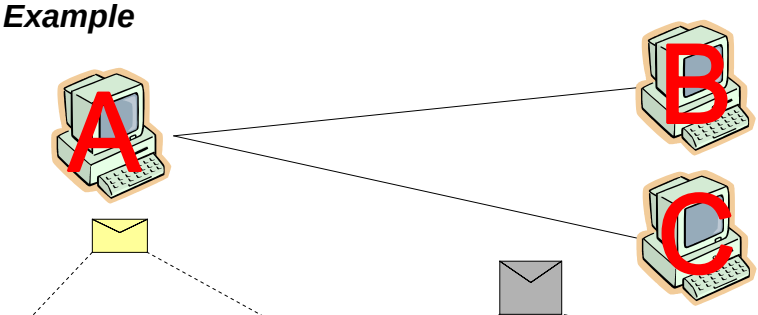


Bridges



- Data Formats
- Protocols
- Transports

An Example

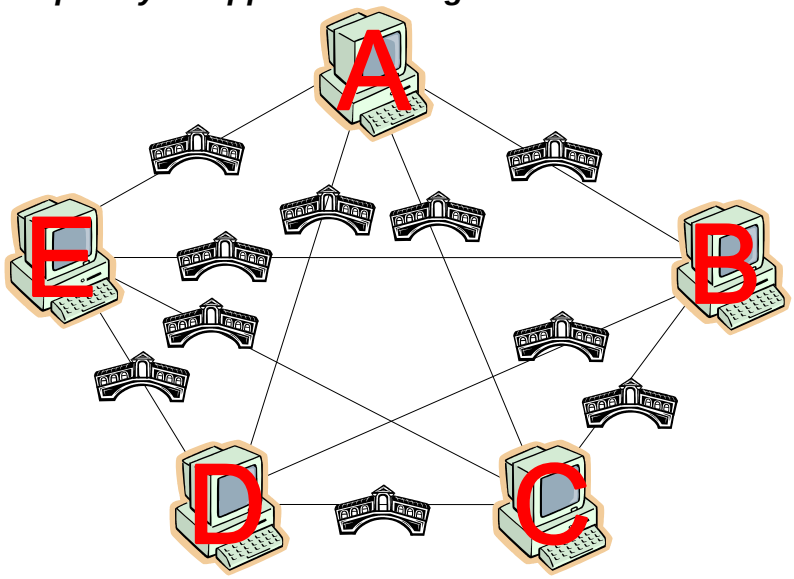



```
Fred Smith,
Graphics Card, 32,
1.50, 07/11/06
```

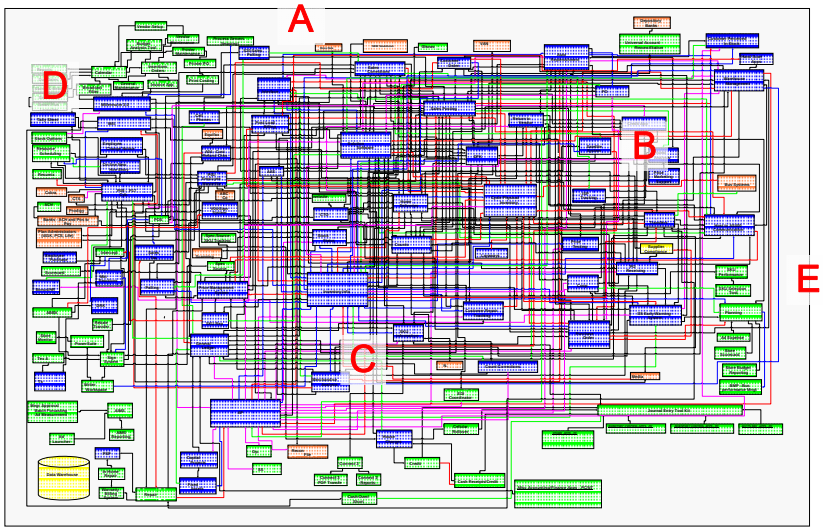
[Customer, Order, Quantity, Price, Date]

```
<order>
  <name>Mr. Smith</name>
  <item>Graphics Card</item>
  <quantity>32</quantity>
  <price>1.05</price>
  <date>11/07/06</date>
</order>
```

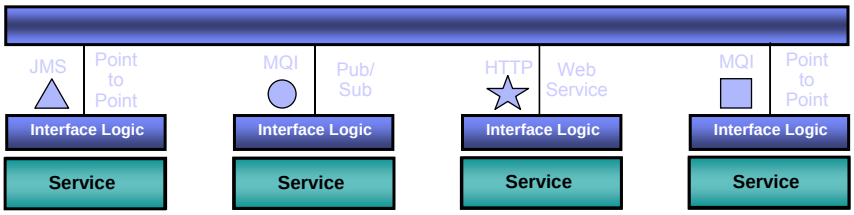
The Complexity of Application Integration



The Complexity of Application Integration (x1000)



What needs to be done?



- ✓ **Decouple** interface logic from applications
- ✓ **Detangle** point-to-point connections so that any application can talk to any other
- ✓ **Enable** applications to communicate with each other regardless of
  - ✓ Programming languages
  - ✓ System platforms
  - ✓ Programming models
  - ✓ Connection protocols
  - ✓ Data formats
- ✓ **Facilitate** application reuse

The solution: SOA and the Enterprise Service Bus

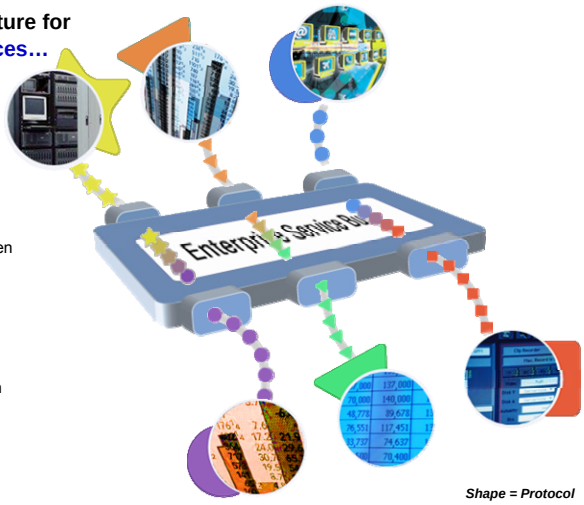
What is an Enterprise Service Bus (ESB)?

A flexible connectivity infrastructure for integrating applications as services…

……which reduces the number, size, and complexity of interfaces.

An ESB:

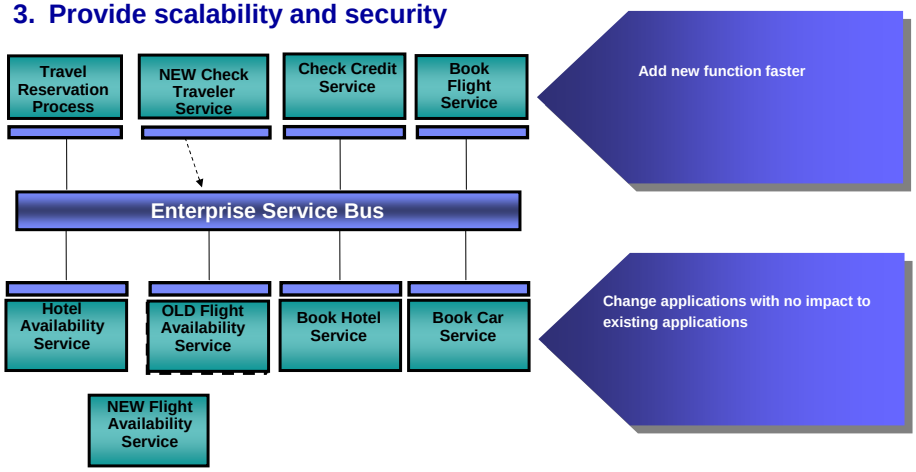
- MATCHES & ROUTES messages between services
- CONVERTS transport protocols between requestor and service
- TRANSFORMS message format between requestor and service
- DISTRIBUTES business events from/to disparate sources.



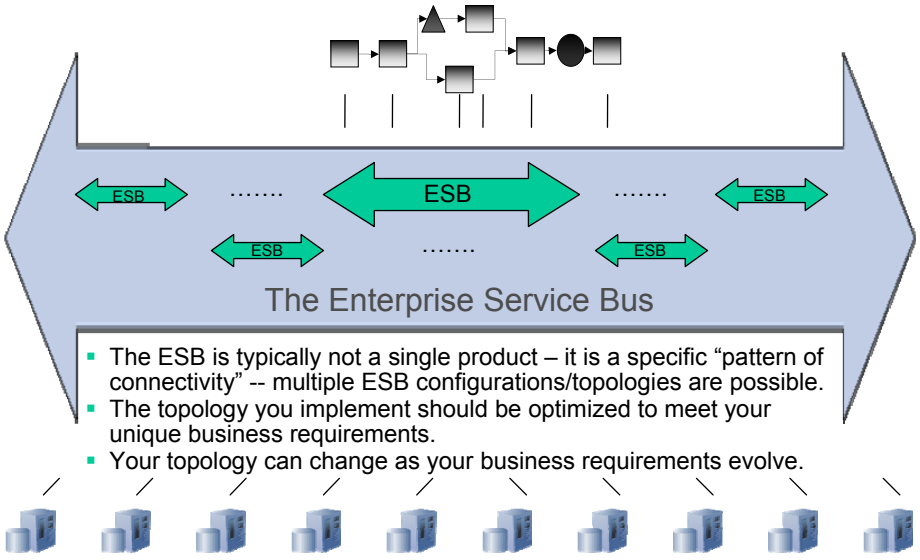
Shape = Protocol
Color = Data type

Value of an ESB

1. **Improve business flexibility**
2. **Reuse services for multiple purposes**
3. **Provide scalability and security**



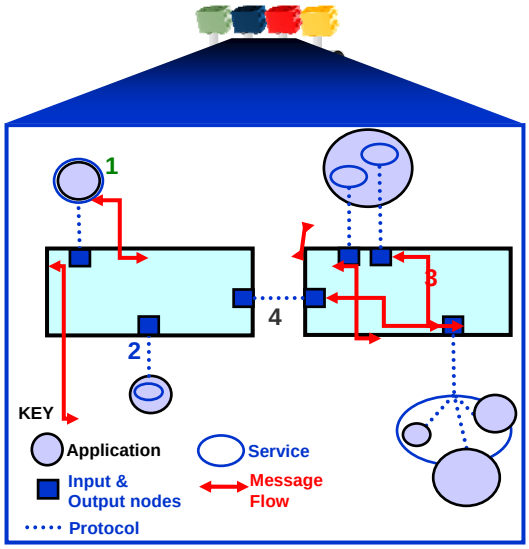
The ESB is not one thing; it is a distributed architecture



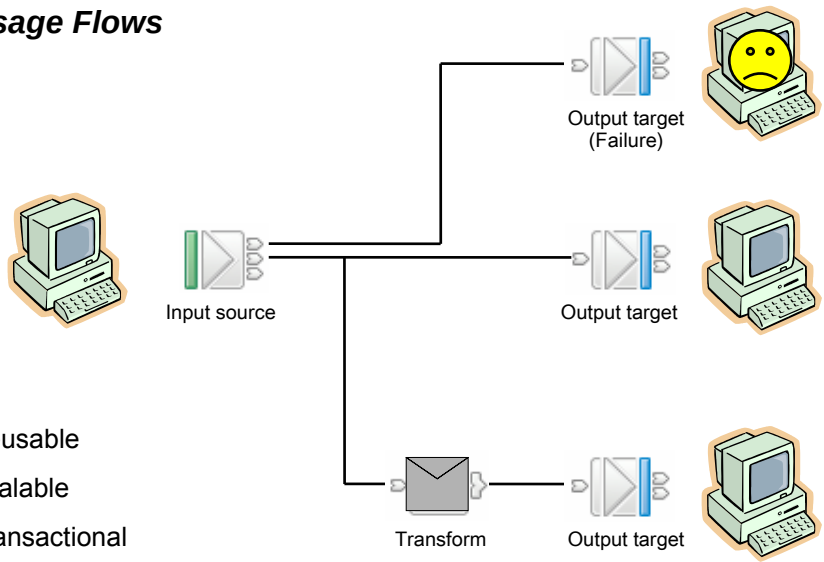
- The ESB is typically not a single product – it is a specific “pattern of connectivity” -- multiple ESB configurations/topologies are possible.
- The topology you implement should be optimized to meet your unique business requirements.
- Your topology can change as your business requirements evolve.

Message Broker as an Advanced ESB

- WebSphere Message Broker
  - creates services from applications.
  - provides service interactions for a broad range of protocol endpoints, including MQ, JMS, HTTP(S) and several others.
  - provides flexible and dynamic connectivity between service endpoints.
  - combines with other brokers to form a scaleable ESB backbone.

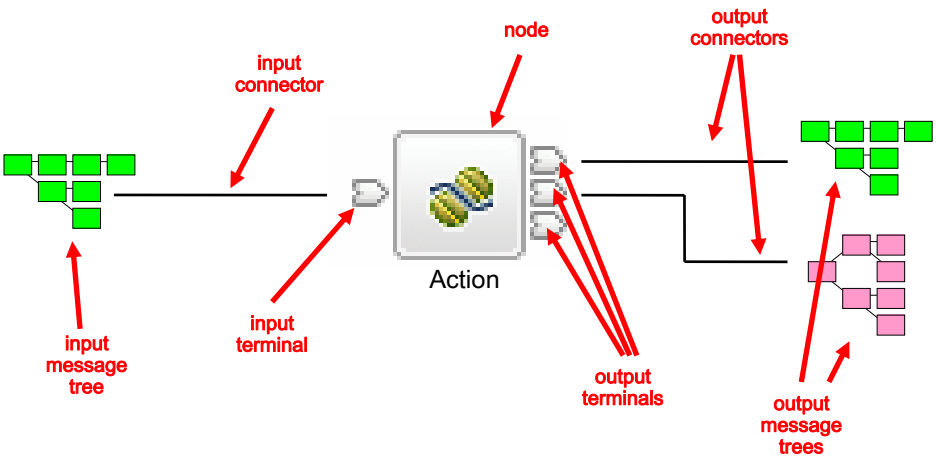


Message Flows



- Reusable
- Scalable
- Transactional

Message Nodes

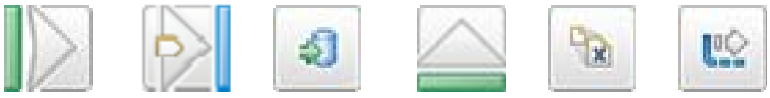


Node Types

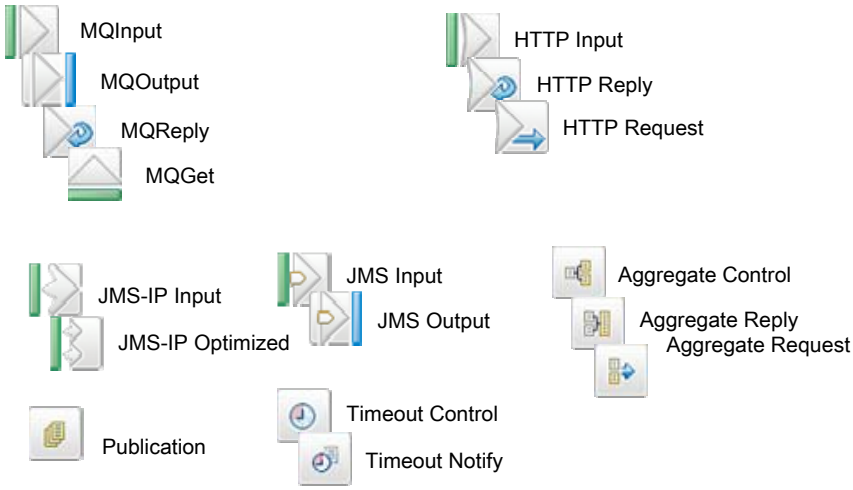
- Input
- Output
- Processing

- Protocol-specific
- Transformation
- Logical construct

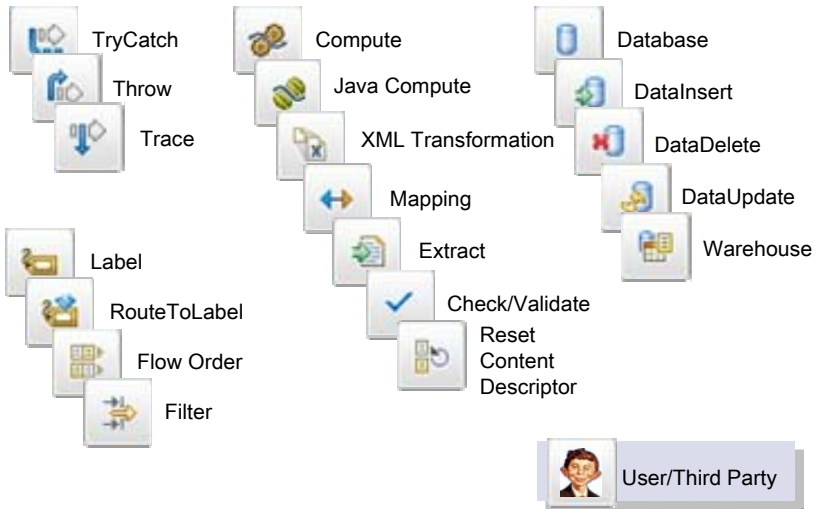
Examples:



Built-in Nodes (1/2)



Built-In Nodes (2/2)



Key SupportPacs and Offerings

- CICS Request Node (z only)
- VSAM Nodes (z only)
- QSAM Nodes (z only)
- WebSphere Transformation Extender (WTX)

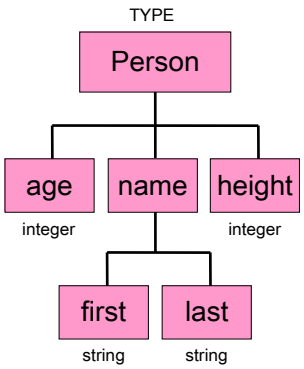
The Logical Message Model – Message Sets

```
<Person age='32' height='172'>
  <name>
    <first>Fred</first>
    <last>Smith</last>
  </name>
</Person>
```

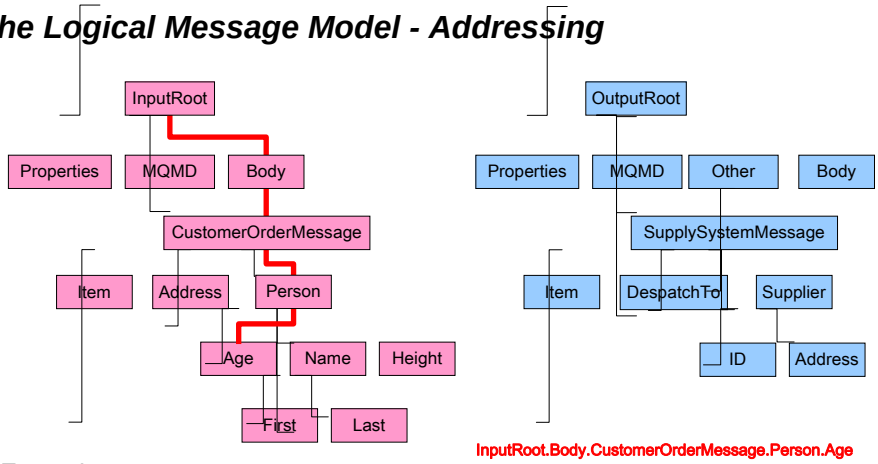
```
struct {
  int height;
  int age;
  char firstname[24];
  char lastname[24];
} Person;
```

172	32	Fred	Smith

```
PER + 172 + 32 + Fred Smith
```



The Logical Message Model - Addressing



Examples:

- Update NAMESDB with the value of *InputRoot.Body.CustomerOrderMessage.Person.Name.Last*
- Set the output message *OutputRoot.Body.SupplySystemMessage.DespatchTo* field to be the value of the input message *InputRoot.Body.CustomerOrderMessage.Address*

Message Processing Examples

DataInsert

```
IF Body.Person.Height > 183 THEN

 INSERT INTO Database.TallPeople
        (Name,Height,Age)
 VALUES (Body.Person.Name,
         Body.Person.Height,
         Body.Person.Age);
ENDIF;
```

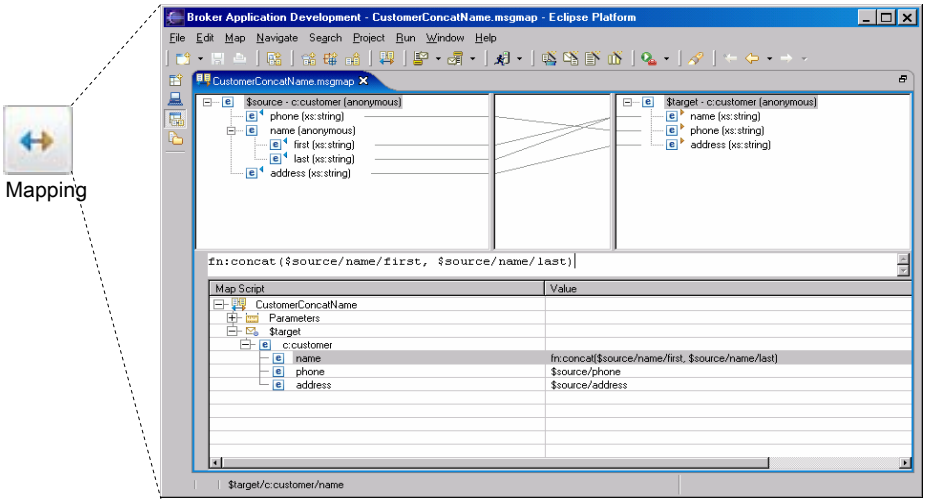
Compute

```
IF (XML format required) THEN
 OutputRoot.Properties.MessageFormat = 'XML';
ELSE IF (custom format)
 OutputRoot.Properties.MessageFormat = 'CWF';
ELSE IF (SWIFT format)
 OutputRoot.Properties.MessageFormat = 'TDS';
ENDIF;
```

Java Compute

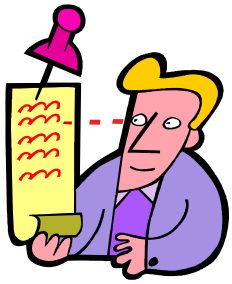
```
public class jcn extends MbJavaComputeNode {
  public void evaluate(MbMessageAssembly assembly)
               throws   MbException
  {
    …
   String personName =
(String)assembly.getMessage().evaluateXPath("/Body/Person/Age");
    …
  }
```

Message Processing Examples



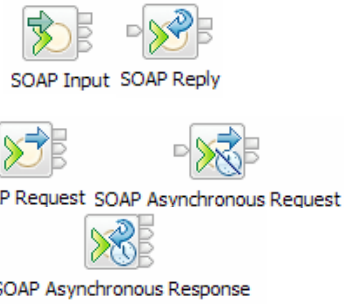
Logical Message Model

- Many message formats have been already defined
  - e.g. MQ, XML, SOAP, MIME, HTTP, SWIFT, ACORD, AL3, EDI-FACT, EDI-X.12, FIX, HL7, HIPAA
- …but you can always model your own from within the advanced Eclipse tooling
  - either from scratch or by importing C structures, XML schema, COBOL copybooks etc.
- Lazy and opaque parsing capability



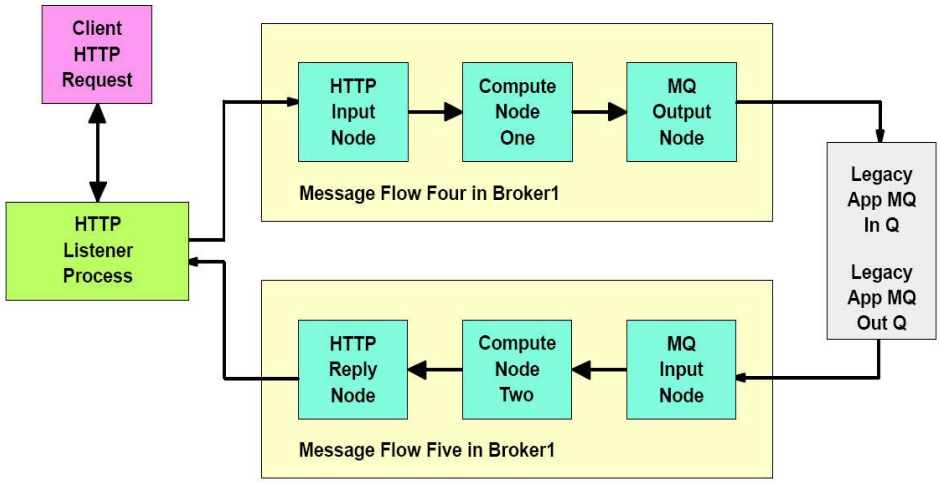
Support for Web Services

- Support WS-Security and WS-Addressing "out of the box"
  - Support for WS-Addressing Endpoint References and Message addressing properties
  - Support for WS-Security authentication, encryption and signing
    - Username password, X509 certificates for authentication
    - Comprehensive encryption and signing algorithms (from JSSE/JCE)
    - Configuration using Policy Sets
      - Eclipse based Policy Set editor to create security profiles
- Support provider and consumer scenarios
  - Provider:
    - SOAP input & SOAP reply
  - Consumer:
    - Synchronous SOAP request
    - Asynchronous SOAP request and reply
  - Can be combined to provide Web Service intermediary
  - SOAP Extract and SOAP Envelope nodes
    - Simplify processing of SOAP payload and headers



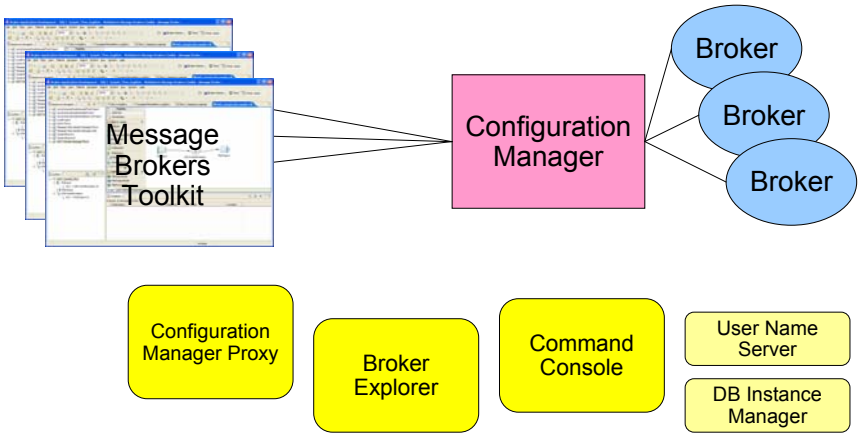
Scenarios - HTTP to MQ

HTTP transport to a MQ-enabled legacy application that does not speak HTTP



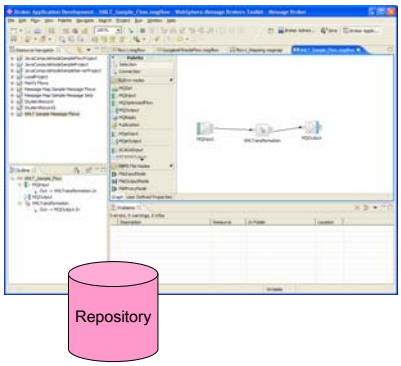
Request ID must be passed through to identify the original client in response

Components of the Message Broker



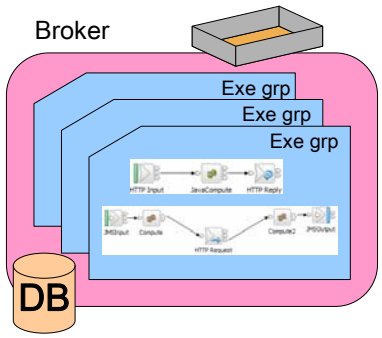
Components – Message Brokers Toolkit

- Development of broker artefacts
  - e.g. message flows and message sets
  - Stored in a local or team repository
- Administration of broker domains
  - e.g. deployment and monitoring
- Based on Rational Application Developer
  - Eclipse 3.0
  - Windows and Linux platforms
  - Included Java, XML development environment
  - Integrated debugging capability



Components – Broker Runtime

- Runs message flow processing logic
- Made up of one or more 'execution group' processes that can run multiple message flows each
  - Provides isolation and scalability
- Available on Windows, z/OS, AIX, HP, Solaris, Linux (Intel, zSeries, PPC)
- Requires a database and a local MQ Queue Manager
- Can also act as a high-performance publish/subscribe provider



Components – Configuration Manager

- Acts as the gateway between the Message Broker Toolkit runtime broker domain
  - Drives deployment to the runtime broker
  - Reduces the load on the runtime broker by providing domain information to the Message Broker Toolkit
  - Controls access to the broker domain for administration and deployment
- Contains an internal repository of domain information and ACLs
- Available on Windows, z/OS, AIX, HP, Solaris, Linux (Intel, zSeries, PPC)
- Requires a local MQ queue manager