Microsoft BizTalk Enterprise Service Bus Toolkit

March 26, 2014

Author : Mallesh Bonla
Agenda

- Application Integration Challenges
- ESB Toolkit Frameworks and Architecture
- Components Drilldown
  - Itinerary Terminology
  - Orchestration
  - Itinerary On-Ramp / Off-Ramp
  - Pipelines
  - Messaging
  - Resolver
  - Adapter Provider
- Itineraries Deployment
- Patterns & Extensibility
- BizTalk ESB Toolkit Benefits
- Summary
Connecting Application in BizTalk with out Service

- **Point-to-Point**
  - Complex Interface
  - Redundant Logic
  - Doesn’t Scale
  - Lacks Visibility

- **Business Impact**
  - Delays Response to changing business needs
Agility thru Service Composition

Enterprise Service Bus

- Service Consumer
- Service Consumer
- Service Consumer

Location & Version Transparency
- Invocation & Orchestration

Transport Protocol Conversion
- Error Handling & Repair

Data Format Transformation
- Message Interactions Support

Service Provider
- Service Provider
- Service Provider
1. Transform my message
2. Determine which endpoint I need
3. Route my message
4. Route the response to a second service
5. Return the final result to me

ESB Toolkit - New Abstraction Layer on Top of BizTalk

Declarative, Meta-data, Policy and Configuration – Driven.

Microsoft BizTalk Enterprise Service Bus Toolkit

Mallesh Bonla
BizTalk ESB Toolkit Components

ESB Toolkit
- On/Off-Ramps
- Management Portal
- Core Web Services

ESB Toolkit Core
- Resolvers
- Adapter Providers
- Itinerary Services
- Exception Management

BizTalk Components
- Adapters
- Dynamic Ports
- Host Environment
- UDDI 3.0
- BAM
- Transformation Engine
- Business Rules Engine
- Orchestration Engine
- Pub Sub Engine
BizTalk ESB Toolkit Architecture

Core Web Services
- Transformation Web Service
- Resolver Web Service
- Exception Web Service
- Operations Web Service
- UDDI Web Service

BizTalk Receive Ports
- On-Ramps
  - Generic SOAP Receive
  - Generic WCF Receive
  - Generic JMS Receive
  - Custom Receive
- Pipeline

BizTalk Send Ports
- Off-Ramps
  - Generic SOAP Send
  - Generic WCF Send
  - Generic JMS Send
  - Custom Send
- Pipeline

ESB Toolkit Core
- Transform Service
- Route Service
- Custom Service
- Resolvers (...)
- Adapter Providers (...)
- Exception Logger
- Fault Processor
- Exception Handler

BizTalk Pub/Sub Engine

Microsoft BizTalk Enterprise Service Bus Toolkit
Mallesh Bonla
The Core ESB Toolkit Components

- Itinerary Services: Message Processor
- Resolvers: Context Finder
- Adapter Providers: Adapter Properties
- Itinerary Policy
# Itinerary Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
</table>
| Itinerary             | • Processing instructions expressed as XML  
                         • Defines the message processing steps and order  
                         • Enables the message to control its own order of execution                                                                 |
| Itinerary Service     | • Individual processing "step" represented in the itinerary  
                         • Each step can be orchestration-based or messaging-based  
                         • Resolvers are used to determine where to route messages  
                         • Adapter Providers promote properties required for delivery                                                                 |
| Itinerary On-Ramp     | • Receive Port that uses specialized ESB pipelines  
                         • ESB Toolkit includes on-ramps, or you can create your own                                                                                 |
| Itinerary Off-Ramp    | • Dynamic Send Port that automatically determines its configuration using ESB-specific promoted properties  
                         • Uses specialized ESB pipelines to send messages  
                         • One-way and two-way off-ramps are provided, or create your own                                                                          |
Itinerary Services: Orchestration

Invoked via orchestration subscription

- Executed by BizTalk runtime via orchestration subscriptions
Reviewing Itinerary Message Flow

Two-Way ESB On-Ramp

1. Request
2. Pipeline resolves itinerary and retrieves it from the repository
3. MessageBox
4. Pipeline caches the itinerary to reattach it to the response
5. Request
6. Pipeline reattaches the cached itinerary to the response
7. Response
8. Pipeline removes itinerary before sending final response message
9. Response

Itinerary Store

Itinerary Cache
Message flow through an ESB-based application

1. The message is send from a service consumer into BizTalk through a two-way on-ramp.

2. The receive pipeline uses the itinerary resolver component to retrieve the appropriate itinerary from the itinerary store database and then attaches the itinerary to the message.

3. The message is published to the MessageBox database where the itinerary is evaluated against the available services, in this case a two-way off-ramp subscribes.

4. The send pipeline in the two-way off-ramp removes the itinerary from the message and places it in the itinerary cache.

5. Service request is made using the message.

6. Response is received from the service, the receive pipeline of the two-way off-ramp retrieves the itinerary from the itinerary cache and reattaches it to the message.

7. The message is published to the MessageBox database where the itinerary is evaluated against the available services, in this case the original two-way on-ramp subscribes to the message.

8. The send pipeline remove the itinerary.

9. Message is sent as response to the original requestor.
Exploring Itinerary (Logical On-Ramp Concept)

- SOAP header contains Itinerary Descriptor
- Select Itinerary Based on Descriptor
- Process Messaging-Based Services
- Processed message and itinerary is published

Diagram:
- WCF Client
- WCF Adapter
- Itinerary On-Ramp
- Receive Pipeline
- MessageBox
- ESBItineraryDb
Exploring Itinerary (Logical Off-Ramp Concept)

1. Request Pipeline
   - Process any messaging-based itinerary services
   - Cache itinerary (for request-response)

2. MessageBox

3. Dynamic Send PortA
   - WCF Adapter

4. WCF Service

5. Dynamic Send PortA
   - WCF Adapter

6. Response Pipeline
   - Resolve itinerary from Cache
   - Process any messaging-based itinerary services

7. Cont...

Microsoft BizTalk Enterprise Service Bus Toolkit
Mallesh Bonla
The processing of a two-way off-ramp will follow the list below

1. Message is routed from the MessageBox to the two-way dynamic send port.

2. The request (send) pipeline processes any messaging-based itinerary services and then caches the itinerary so it can be reattached to the response message.

3. The message is sent to the external service using an adapter.

4. The external service does its thing.

5. The response is received back into BizTalk via the adapter.

6. The response (receive) pipeline resolves the itinerary from the itinerary cache and reattaches it to the message, then executes any messaging-based itinerary services.

7. The message is published to the MessageBox.
The Role of ESB Pipelines

General ESB Pipeline Concepts

• The Toolkit includes components to support the ESB pattern
  ✓ Ships with several pipelines built from these components
  ✓ Duplicate and augment standard BizTalk processing functionality
• Receive pipelines attach itineraries, and optionally process services
• Send pipelines are responsible for caching itineraries before sending, and optionally process services
  ✓ For request-response, itinerary is cached and matched to response.

Core ESB Pipeline Components

• Components Implementing the Routing Slip Pattern
  ✓ Itinerary Component
  ✓ Itinerary Selector Component
  ✓ Itinerary Cache Component
  ✓ Dispatcher Component
  ✓ Dispatcher Disassemble Component
  ✓ Forwarder Component
  ✓ Identifying Bonus ESB Pipeline Components
  ✓ The Namespace Components
## Pipelines Included with the ESB Toolkit

### ESB Receive Pipelines

<table>
<thead>
<tr>
<th>Pipeline Name</th>
<th>Included Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItineraryReceive</td>
<td>• ESB Itinerary</td>
</tr>
<tr>
<td></td>
<td>• XML Disassembler</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td>ItineraryReceiveXml</td>
<td>• ESB Itinerary</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher Disassemble</td>
</tr>
<tr>
<td>ItineraryReceivePassthrough</td>
<td>• ESB Itinerary</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td>ItinerarySelectReceive</td>
<td>• ESB Itinerary Selector</td>
</tr>
<tr>
<td></td>
<td>• XML Disassembler</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td>ItinerarySelectReceiveXml</td>
<td>• XML Disassembler</td>
</tr>
<tr>
<td></td>
<td>• ESB Itinerary Selector</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td>ItinerarySelectReceivePassthrough</td>
<td>• ESB Itinerary Selector</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
</tbody>
</table>

### ESB Send Pipelines

<table>
<thead>
<tr>
<th>Pipeline Name</th>
<th>Included Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItinerarySend</td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td></td>
<td>• XML Assembler</td>
</tr>
<tr>
<td></td>
<td>• ESB Itinerary Cache</td>
</tr>
<tr>
<td>ItinerarySendPassthrough</td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td></td>
<td>• ESB Itinerary Cache</td>
</tr>
</tbody>
</table>

### ESB Response Pipelines

<table>
<thead>
<tr>
<th>Pipeline Name</th>
<th>Included Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItinerarySendReceive</td>
<td>• Same functionality as the ItineraryReceive and ItinerarySelectSendReceive, but adding the Itinerary Cache component in the Decode stage</td>
</tr>
<tr>
<td>ItineraryForwarderSendReceive</td>
<td>• ESB Itinerary Cache</td>
</tr>
<tr>
<td></td>
<td>• XML Disassembler</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher</td>
</tr>
<tr>
<td></td>
<td>• Forwarder</td>
</tr>
<tr>
<td>ItineraryForwarderSendReceiveXml</td>
<td>• ESB Itinerary Cache</td>
</tr>
<tr>
<td></td>
<td>• ESB Dispatcher Disassemble</td>
</tr>
<tr>
<td></td>
<td>• Forwarder</td>
</tr>
</tbody>
</table>

---

Microsoft BizTalk Enterprise Service Bus Toolkit

Mallesh Bonla
Itinerary Services: Messaging

Invoked as by ESB interceptor pipeline component

- **Messaging Services**
  - Loaded using dependency injection
  - Can produce multiple messages if invoked from ESB disassembler
ESB Toolkit Resolver

- ESB Resolver
  - Endpoint Resolution
    - UDDI 2.0/3.0
    - XPATH
    - Static
    - Business Rules Engine (BRE)
  - Artifact Resolution
    - Itinerary (BRE)
    - Itinerary Static
ESB Adapter Provider

- FTP
- WebSphere MQ
- File
- Custom
- WCF-BasicHTTP
- WCF-WsHTTP
- WCF-Custom
  - WCF-SQL
- SMTP
Itineraries Deployment

Design Time

- XML File
- Itinerary Store
- BizTalk Deployment Tools

Test/Run Time

- Message
- OnRamp
- Itinerary
- Itinerary Selector
- Itinerary & Message
- ESB Processing

BizTalk Deployment Tools

Microsoft BizTalk Enterprise Service Bus Toolkit
Mallesh Bonla
Itinerary Modeling

Models and designer extensions

- Itinerary DSL model
- Service Definition
- Itinerary Runtime Model (XML)
- Runtime Definition
- Technology Definition
- Custom Extender
- Designer Extenders
- Service Definition
- Adapter Manifests
- Itinerary Services Configuration
Three Ways to Assign Itineraries

- **Advanced Service Client**: Client sends an itinerary in a WCF/SOAP Header request.

- **Adaptive Service Client**: Client resolves itinerary via resolver service and then sends it in a WCF/SOAP Header request.

- **Service Proxy**:
  - Itinerary is resolved on the server’s ESB On-Ramp receiving port via configurable resolver.
ESB Design Patterns

The ESB Toolkit promotes pattern-based development

- Message Routing Patterns
  - Message Router
  - Content-Based Router
  - Routing Slip
  - Scatter-Gather
  - Recipient List
  - Splitter

- Message Transformation Patterns
  - Message Translator
  - Normalizer
  - Content Enricher

- Service Mediation Patterns
  - VETO/VETRO
  - Request-Response

- Service Management Patterns
  - Repair and Resubmit
# BizTalk ESB Toolkit Extensibility

<table>
<thead>
<tr>
<th>Extensibility Point</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter Provider</td>
<td>Create new Adapter Provider component, manifest file</td>
</tr>
<tr>
<td>Orchestration Service</td>
<td>Implement orchestration, update configuration</td>
</tr>
<tr>
<td>Messaging Service</td>
<td>Implement service and designer extender</td>
</tr>
<tr>
<td>Resolver</td>
<td>Implement resolver component</td>
</tr>
<tr>
<td>Composite Resolver</td>
<td>Define Unity container, fact providers, fact translators</td>
</tr>
<tr>
<td>Cache Manager</td>
<td>Support Enterprise Library 4.1 API</td>
</tr>
<tr>
<td>Itinerary Exporter</td>
<td>Implement custom exporter component</td>
</tr>
</tbody>
</table>
BizTalk ESB Toolkit provides

Service Composition Challenges:
• Location & Version Changes
• Sequencing Changes
• Data Semantics Mismatch
• Transport Protocol Mismatch
• Interaction Model Mismatch
• Quality of protection
• Error Recovery
• Monitoring & Visibility
• Quality of service

BizTalk ESB Solutions:
➢ Dynamic Endpoint Resolution
➢ Itinerary-based Routing
➢ Dynamic Transformation
➢ Protocol Mediation
➢ Itineraries-based Processing
➢ BizTalk Security Model & SSO
➢ Exception Framework & Portal
➢ BAM Integration
➢ Exception Management Portal
BizTalk ESB Toolkit Benefits

Provides the right benefits to cope with complex and rapidly changing integration challenges

- Higher levels of service re-use
- Lower operational costs
- Faster adaptation to business changes
- Visibility business and exception metrics
- Highly extensible to introduce new functionality or encapsulate patterns