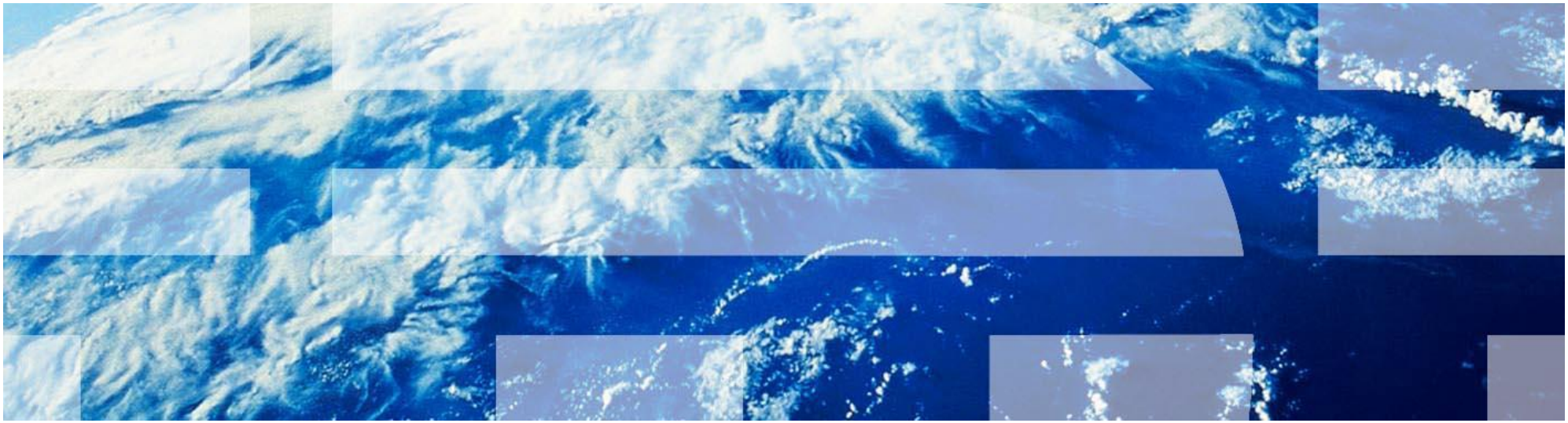
An IBM Proof of Technology

WebSphere® software

WebSphere ESB Conversion:

Comparing WESB and IIB Development



Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

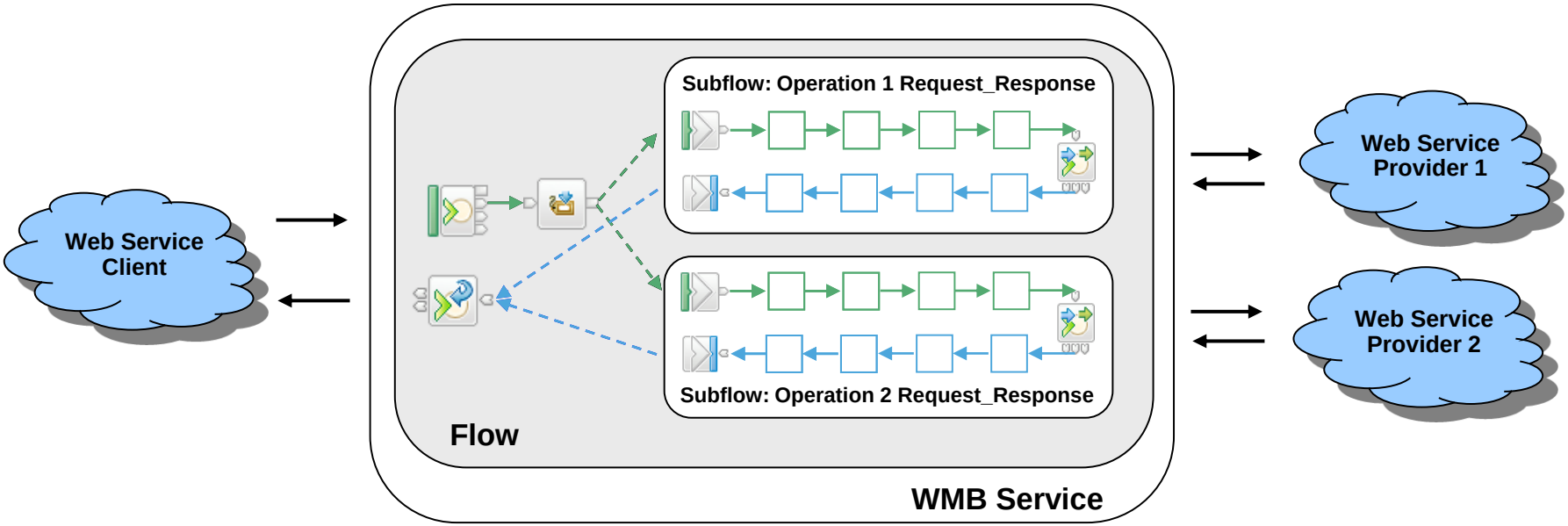
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

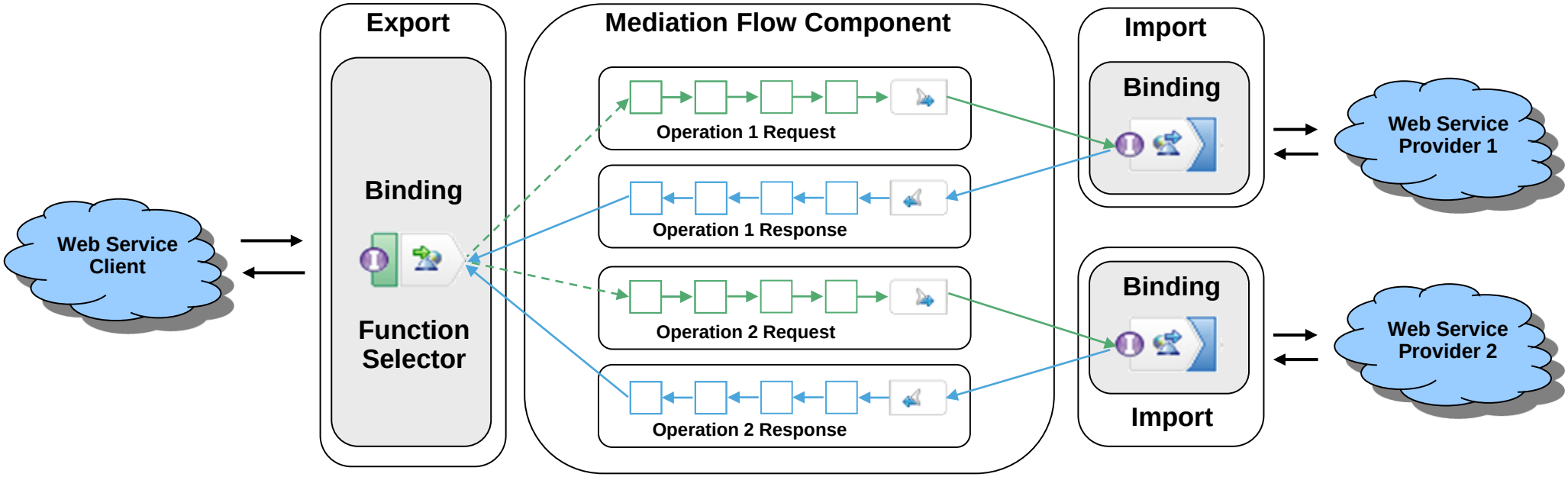
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

- **Application Development Concepts and Tools**
- Capability Comparison
- ESB Topology Discussion
  - WESB Physical Topology
  - IIB Physical Topology

Architectural Concepts





Tooling Environment & Development Runtime

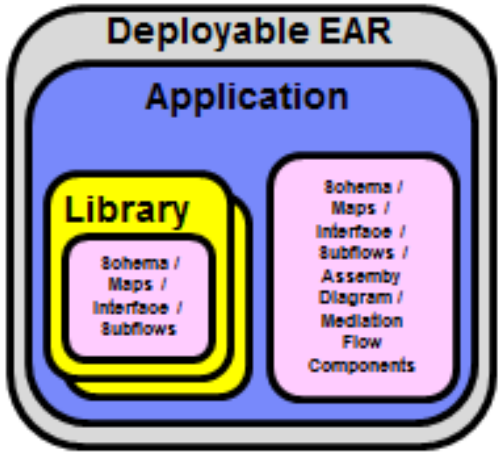
- **IBM Integration Designer (WID) and IBM Integration Toolkit**
  - Eclipse based includes the majority of the same perspectives
  - Dedicated perspective for Integration development
  - Extensive debugging capabilities for node and flow development
  - Built-in testing capabilities
  - Provides pluggable framework for code repositories
  - Windows and Linux support provided
  - **IBM Integration Toolkit is provided free of charge: http://www-01.ibm.com/support/docview.wss?uid=swg21644591**
- **Development Runtime**
  - Fully functional product provided free of charge for development and unit testing
  - Deployment directly from development environment
  - Runtime can be either local or remote

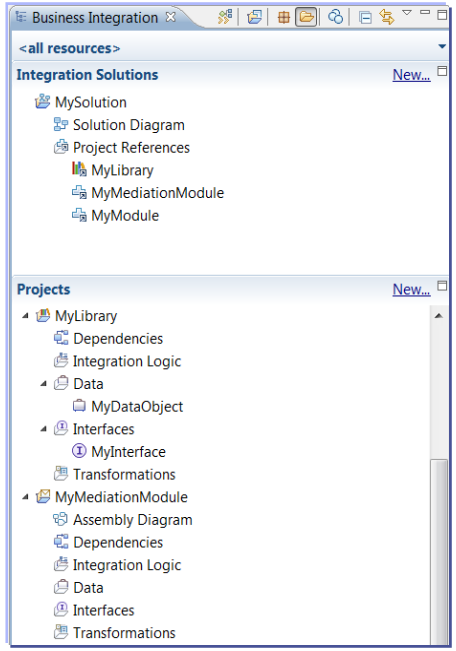
Tooling Project Types

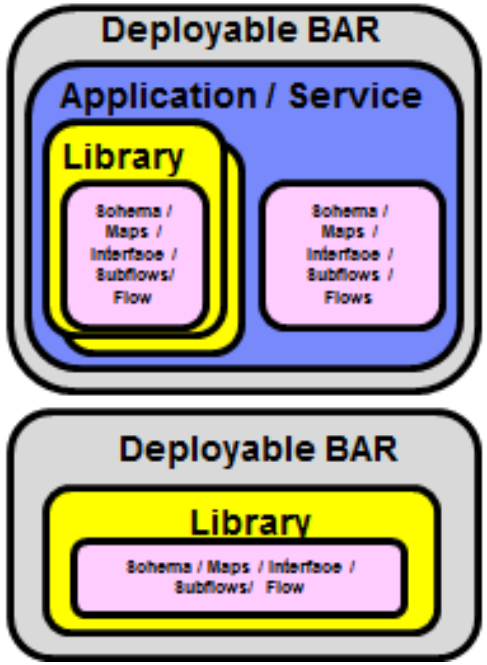
- **IBM Integration Designer (WID)**
  - Applications provide a container for integration logic
  - Libraries are associated with applications to provide common logic
  - Deployment is always at the Application level and normally includes a copy of the library within each application (shared by copy)
  - Integration Solutions can provide a overall dependency visualization across applications
  - The runtime provides application isolation ensuring that one deployed application can not conflict with another

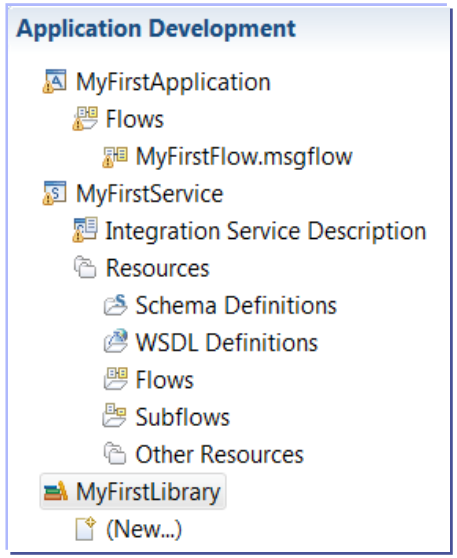
IBM Integration Toolkit

- Applications & Services both provide containers for integration logic with Services focused on exposure of HTTP/SOAP Web Services
- Libraries can be associated with applications and services to provide common logic
- Deployment can occur at the Application, Service or Library level.
- Application and Services always include a copy of the library (shared by copy) and can not access any library deployed independently
- The runtime provides application isolation for all artefacts except for Java and .NET.









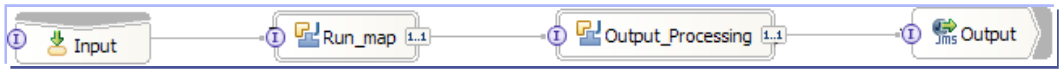
Tooling Project Type considerations

- To isolate Java code within IBM Integration Bus a separate Integration Server may need to be created.
- Although deployment uses a different mechanism, global schemas are supported in WebSphere ESB. In IBM Integration Bus this is also possible but **ALL** integration logic (e.g. flows) using the global schemas need to be included within libraries instead of applications or services.
- Services are a logical landing point for conversion activities however they currently only support SOAP/HTTP and therefore are not always appropriate.

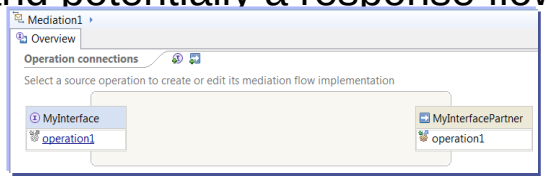
Integration construction

IBM Integration Designer (WID)

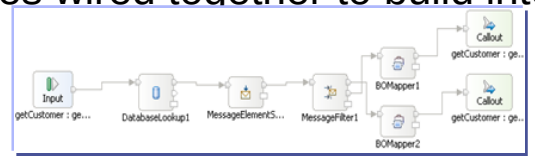
- Integration logic is constructed using four concepts:
  - **Assembly diagram**: provides an overall view of the integration logic showing the interfaces, protocols, and integration logic separated into logical components.



  - **Mediation Flow Component**: is a container for all the flows required for one or more interfaces. For each operation in each interface a request, error and potentially a response flow will exist.



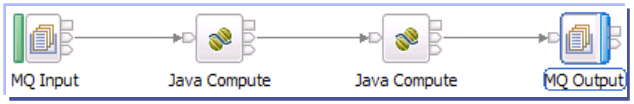
  - **Flows**: contain any number of mediation primitives wired together to build integrations.



  - **Subflows**: are similar to flows but have to be contained within a flow to be run. They provide a useful mechanism to create reuse.

IBM Integration Toolkit

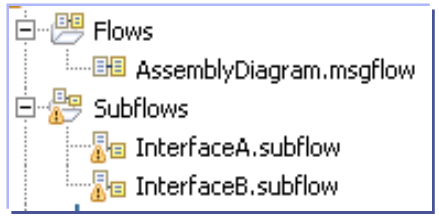
- Integration logic is constructed using two concepts:
  - **Flows**: contain any number of nodes wired together to build integrations. This includes both integration logic and exposed protocols.



  - **Subflows**: are similar to flows but have to be contained within a flow to be run. They provide a useful mechanism to create reuse

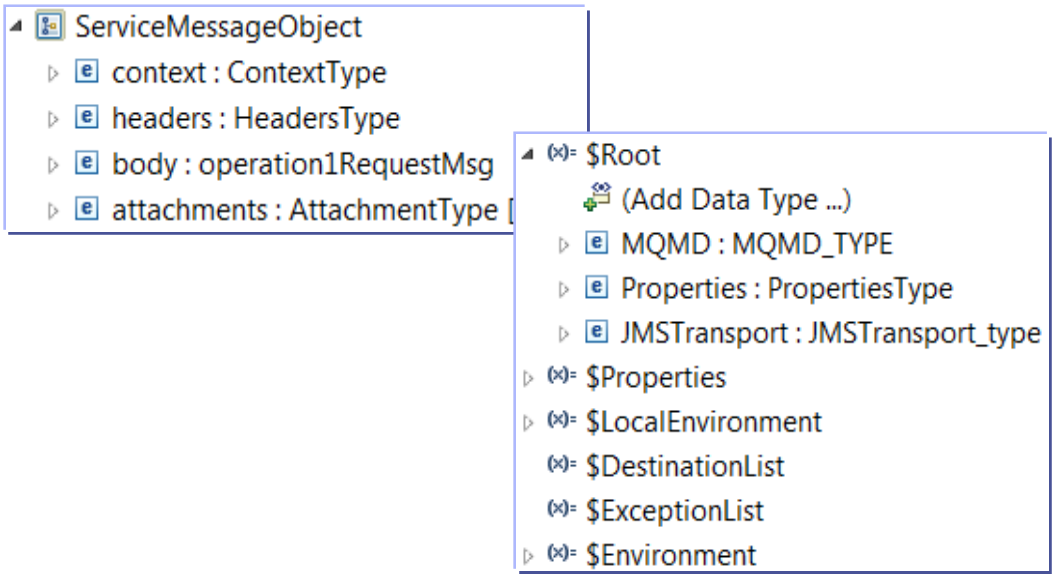
Integration construction considerations

- IBM Integration Toolkit provides the concept of a service which maps almost to a mediation flow component with the exception that a single interface can only be used. Where possible, consider using services as a landing point for conversions.
- Although IBM Integration Toolkit does not provide an assembly diagram it is possible via naming convention to create a similar high level interaction in a flow with the integration logic contained within subflows.



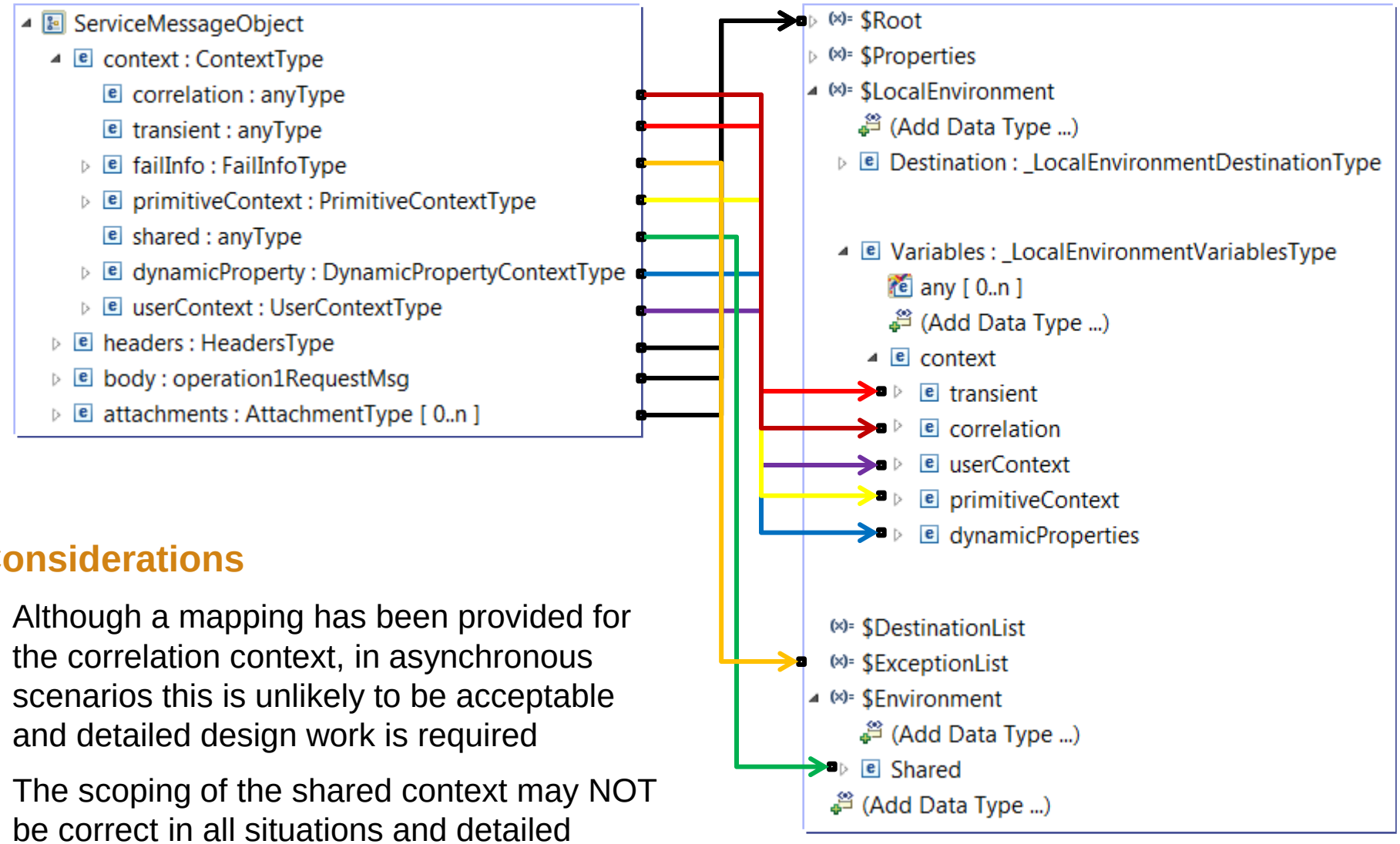
Data Model

- Service Message Object
  - Context: data other than the message payload and headers
  - Header: information associated with the message. For example, Java Message Service (JMS) headers if a message has been conveyed using the JMS API, or MQ headers if the messages has come from WebSphere MQ.
  - Body: the message payload, is the application data exchanged between service endpoints
  - Attachments: used for SOAP Attachments



- Message Tree
  - Root: contains the protocol header and payload of the message
  - Properties: contains a set of protocol independent properties for the message being processed. For instance the codepage of the message.
  - Local Environment: provides a message level structure for built-in and user defined data. For instance the outbound connectivity information may be stored here for a downstream node.
  - DestinationList: used by the RouteToLabel capability
  - ExceptionList: location where the message flow writes information about exceptions that occur when a message is processed.
  - Environment: provides a message flow level structure for built-in and user defined

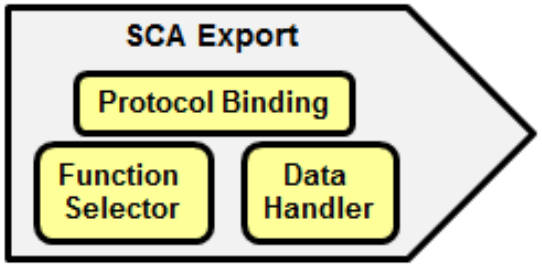
High-level mapping between the two structures



- **Considerations**
  - Although a mapping has been provided for the correlation context, in asynchronous scenarios this is unlikely to be acceptable and detailed design work is required
  - The scoping of the shared context may NOT be correct in all situations and detailed design is required

Protocol handling

- **IBM Integration Designer (WID)**
  - SCA Export/Imports are used to represent data entering and leaving the system. These are then bound to a protocol such as MQ, JMS, HTTP, WebServices etc.
  - Protocol neutral data handlers are then associated that transform the wire bytes into the Service Message Object (indirectly).
  - Several built-in data handlers exists for instance: XML, JSON, Fixed Width and Delimited
  - Custom data handlers can be created using a Java API
  - Functional selector are used to select the operation for the interface associated with the SCA Export



- **IBM Integration Toolkit**
  - Individual input and output nodes are provided for each protocol supported.
  - Protocol neutral parsers are associated with the input nodes to transform the wire format data into the message tree. The parser used to complete the processing is stored within the message tree and used for serialization.
  - Changing the input and output parser is completed by either reparsing the input data or manually changing the shape of the message tree.
  - Built-in parsers include: XML, JSON and DFDL
  - DFDL allows custom message formats to be created using either a graphical or XML editor.
  - Custom parsers can also be created

Protocol Handling considerations

- The DFDL support provides a far more straight forward mechanism to handle custom data formats and moving forward this will speed up the development time.
- Conversion of data handler code is encouraged towards DFDL instead of custom parsers
- Once parsed the SMO was wire format independent, however the message tree structure is not and if the wire format needs to be modified (e.g. JSON → XML) then the message tree structure needs to be modified

Flow and node/primitive processing

- **IBM Integration Designer (WID)**
  - A inbound message is processed using a single thread
  - The flow engine is stack based
  - Dynamic Properties can be used to override the behaviour of the flow
  - Subflows are deployed by copy
  - Automatic correlation between asynchronous request and responses is provided
  - Data integrity is enforced by the flow engine
  - Firing an output terminal is buffered until the processing of that primitive has completed. Then the wired primitive(s) are invoked.
  - Flow ordering requires explicit mediation primitives
  - Fail terminals only fire on exceptions

- **IBM Integration Toolkit**
  - A inbound message is processed using a single thread
  - The flow engine is stack based
  - A combination of the local environment and configurable services provide dynamic behaviour
  - Subflows can be deployed by reference
  - Correlation between asynchronous messages is provided via a number of customizable mechanisms
  - Data integrity is the responsibility of the node developer
  - Firing an output terminal causes the wired node to be called immediately
  - Flow ordering requires explicit nodes
  - Fail terminals can be explicitly fired by node developers

Flow and node/primitive processing consideration

- Application developers need to be aware that automatic correlation of asynchronous request/responses does not occur
  - Isolate state within each Integration Node/Server
  - Use the Global Cache feature of IBM Integration Bus
  - Use an external Database
  - Configure high availability failover of IBM WebSphere MQ and designate a queue manager as a state repository
- Custom mediation primitives conversions will need to evaluate the impact of:
  - Immediate firing of terminals
  - Data integrity

Agenda

- Application Development Concepts and Tools
- **Capability Comparison**
- ESB Topology Discussion
  - WESB Physical Topology
  - IIB Physical Topology

Protocol comparison

- **WebSphere Enterprise Service Bus**
  - Supports a number of industry standards:
    - Web Services: SOAP 1.1/1.2 HTTP & SOAP 1.1 / JMS
    - JMS Support for WAS/MQ/3rd Party
    - **MQ**
    - HTTP
    - **EJB**
    - **SCA**
  - Support for adapters (included in license):
    - Email
    - Flat File
    - FTP
    - IMS
    - CICS
    - **Lotus Domino**
    - JDBC
    - **Enterprise Content Management**
    - **iSeries**
  - Support for application adapters (additional license required)
    - SAP
    - Siebel
    - PeopleSoft
    - JD Edwards EnterpriseOne
    - **Oracle E-Business**

- **IBM Integration Bus**
  - Supports a number of industry standards:
    - Web Services: SOAP 1.1/1.2 HTTP & **JMS**
    - **Asynchronous Web Service support SOAP 1.1/1.2 HTTP & JMS**
    - JMS Support for MQ and other JMS providers such as SIB
    - Local MQ Support
    - TCP
    - CORBA
  - Support for technology and applications (included in Advanced license, separate license for application adapters for Scale customers)
    - Email
    - Flat File
    - FTP
    - IMS
    - CICS
    - JDBC
    - SAP
    - Siebel
    - PeopleSoft
    - JD Edwards EnterpriseOne

Protocol consideration

- IIB provides significantly improved capabilities in many areas including asynchronous web service support
- If the MQ binding is used it may be required to use the JMS binding for equivalent support in IIB as the MQ Input/Output nodes only support connectivity to the local queue manager
- Internal SCA bindings are not supported in IIB, if these are required to communicate with WPS/BPM Advanced functionality then external protocols will need to be added to the WPS/BPM Advanced applications
- Only outbound non-transactional support is provided for EJBs using the Java Compute node if additional support is required WebSphere Application Server is required as a fronting entry point.
- Support Pac IAM7 is available for iSeries support
- Access to Oracle resources is possible using direct connectivity instead of utilizing an adapter

Web Services standards comparison

- **WebSphere Enterprise Service Bus**
  - WS-Addressing 1.0
  - WS-AT 1.0/1.1/1.2
  - Message Transmission Optimization Mechanism (MTOM)
  - XML-binary Optimized Packaging (XOP)
  - WS-RM 1.0 / 1.1
  - WS-COOR 1.0/1.1/1.2
  - WS-Policy
  - Security Standards
    - LTPA
    - Canonical XML 1.0
    - Decryption Transform for XML Signature
    - Exclusive XML Canonicalization
    - WS-Security 1.0 & 1.1
    - Kerberos Token Profile 1.1
    - SAML 1.1 & 2.0 assertions
    - Username Token 1.0 & 1.1
    - X 509 Token 1.0 & 1.1
    - WS-Secure Conversation 1.0 (draft) & 1.3
    - WS-Trust 1.1 & 1.3
    - XML Signature Syntax and Processing
    - XML Encryption Syntax and Processing

- **IBM Integration Bus**
  - WS-Addressing
  - WS-RM certain parts of the standard
  - Message Transmission Optimization Mechanism (MTOM)
  - XML-binary Optimized Packaging (XOP)
  - SOAP with Attachments
  - Security Standards (certain aspects)
    - LTPA
    - Kerberos Tokens
    - SAML assertions
    - Username Token 1.1
    - X509 Tokens 1.1
    - XML Signature Syntax and Processing
    - XML Encryption Syntax and Processing

Gateway comparison

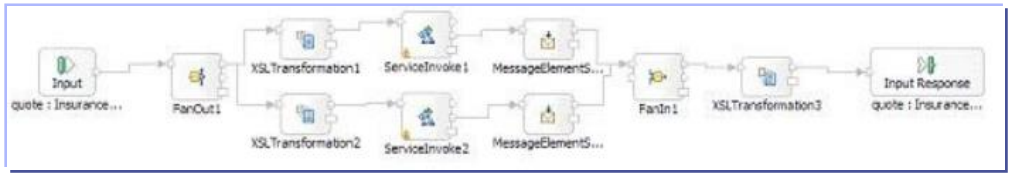
- **WebSphere Enterprise Service Bus**
  - Proxy gateway provides an out-of-the-box, end-to-end, Web services-based service gateway that enables new services to be added without stopping and starting the gateway application.
  - Gateway resolved WSDLs (end service interface + gateway binding) can be returned using the proxy gateway.
  - If access to the payload is required as a business object then the schema files need to be deployed.
  - Dynamic and Static Gateways provide supported for JMS/MQ/HTTP and Web Services
  - For SOAP/HTTP extensive support for Web Service standards including: WS-AT, WS-RM, WS-Security, WS-Addressing
  - Support for endpoint resolution based on built-in configuration store, WebSphere Service Registry Repository (Action based, SLA or custom) or database
- **IBM Integration Bus**
  - Weakly typed programming model provides native support for many Gateway scenarios
  - Dedicated support added to Web Service input/output nodes to allow WSDL-less mode
  - Support provided for several Web Service standards: WS-RM, WS-Security and WS-Addressing
  - Support for independently deployed schemas
  - Support for endpoint resolution based on database or custom integration

Gateway consideration

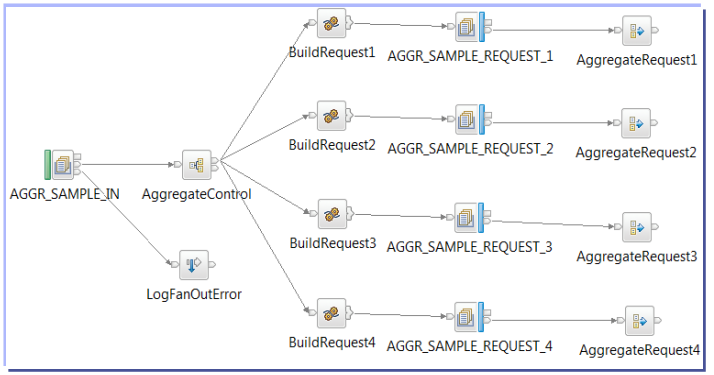
- In most cases the Web Service standards provided by IIB will be adequate, however careful review is recommended
- IIB provides improved support for independent deployment of schema files. In addition it also allows parsing of the data without any schema information into the corresponding message tree (validation disabled).
- Proxy Gateway capabilities are NOT provided in IIB and therefore resolved WSDLs and built-in endpoint resolution is not possible.
- Similar WSRR endpoint resolution is possible via custom integration using public APIs

Aggregation comparison

- **WebSphere Enterprise Service Bus**
  - FanOut/FanIn nodes provide aggregation support for a number of common patterns:
    - Aggregation of data from multiple sources
    - Batch processing with message enrichment
  - FanOut allows for two modes of operation:
    - Iterative/Batch processing
    - Branching requests
  - FanIn allows for three modes of collection:
    - Number of message received
    - XPath expression
    - All messages have been processed



- **IBM Integration Bus**
  - The Aggregation nodes provide support for separation of a message into multiple output messages and fan-in of the replies
  - Aggregate Control node marks the start of the aggregation
  - Aggregate Request node records that a request message has been sent saving relevant state data for the Aggregate Reply node
  - Aggregate Reply node collects the reply messages and combines them together



Aggregation consideration

- No requirement for a shared context as the AggregateReply node handles this internally.
- The AggregateReply node only fires the output terminal on completion or timeout, if the other FanIn completion options are required either custom implementations or the use of the collector node is suggested **(the collector node is not included in Scale mode).**
- Aggregate nodes are limited to Request/Reply invocations and therefore synchronous invocations need to occur downstream
- No built in FanOut iterative mode, however a compute node can be used to iterate over the array after a aggregateControl
- The output from a Collector node needs to be processed using a compute node as graphical mapping is not currently supported.

Event Sequencing comparison

- **WebSphere Enterprise Service Bus**
  - Event sequence qualifiers allow operation level control of sequential processing of messages.
  - This is normally enforced based on one or more keys within the message to allow parallel processing of unrelated data.
  - Supports messages arriving from Messaging and EIS bindings
- **IBM Integration Bus**
  - Event sequencing is provided in two locations:
    - The MQ input node provides the ability to sequence messages. “Sequence groups” are provided so parallel processing of unrelated data can occur
    - Sequence/resequence nodes are provided to enforce the ordering of messages. “Sequence groups” processing is also available.

Event Sequencing consideration

- Although the event sequencing capabilities are exposed differently in IIB Advanced, IIB provides improved flexibility for several use cases.
- Currently sequence/re-sequence nodes are not available in scale mode
- With scale mode you are limited to MQ event sequencing and therefore redesigning aspects of the application is likely.

Resource Comparisons

WESB Terminology	WMB (IIB) Terminology	Description
Mediation Module	Broker Archive File	The unit of deployment mediation capability
Mediation Primitive	Message Flow Primitive Node	An individual discrete piece of function applied to a message.
Export component	**Message Flow ("concrete" input node in main flow)**	**Provides inbound receipt of requests and sending responses**
Import component	**Message Flow ("concrete" output node in main flow)**	**Provides outbound sending of requests and receipt of responses**
Flow component	Message Flow	Contains request, response, fault and error flows
Reference	No equivalent	Point of exit from a component, can be wired to services of other components
Service	No equivalent	Point of entry into a component
Interface	**Interface (WSDL – but no strong typing on individual nodes)**	**Collection of operations with defined input, output and fault messages**
Operation	Data held in LocalEnvironment tree	An individual operation with input, optional output and optional fault messages
Input message	Message	The (schema-defined) request message usually received at an export
Output message	Message	The (schema-defined) response message usually received at an import
Fault message	Message	The (schema-defined) fault message usually received at an import
Qualifier	No equivalent	Quality of service indicator associated with a component reference or interface.
Binding	**Input / output node**	**Protocol / transport specific configuration associated with an import or export**
Function Selector	**Route node**	**Pluggable logic which identifies the operation from a wire format message**
Fault Selector	Route node	Pluggable logic which identifies the fault name from a wire format message
Business Object	Payload	Memory representation of an input / output / fault message
Service Message Object	Message tree	Memory representation of a Business Object plus additional contextual information (transport headers, correlation information, user context)

Primitive / Node Comparison

WESB Primitive	**WMB Node** (IIB Message Flow Nodes)
Service Invoke	Request Nodes
MessageFilter	Route Node
TypeFilter	Compute Node
Endpoint Lookup	Endpoint Lookup
FanOut/FanIn	Aggregate/Collector
Policy Resolution	Registry lookup and compute node
Flow Order	Flow Order
UDDI	Compute Node and SOAP request node
Gateway EPL	Compute and HTTP Request node
SLA Endpoint	Registry Lookup, Compute
SLA Check	Registry Lookup, Compute
XSLT	XSLTransform
BO Mapper	Mapping Node
Message Element Setter	Mapping Node

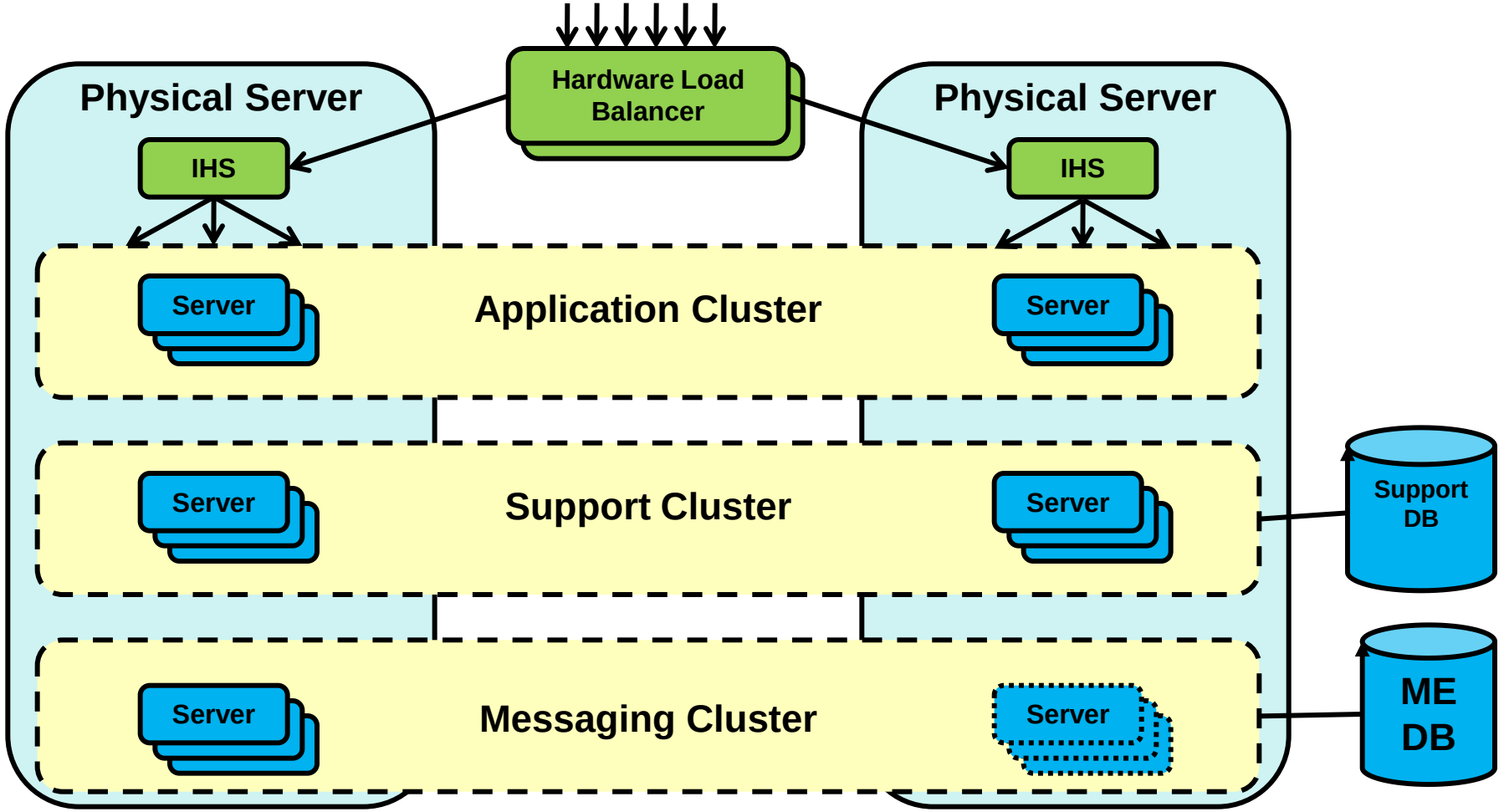
WESB Primitive	**WMB Node** (IIB Message Flow Nodes)
SetMessageType	Reset Content Descriptor
Database Lookup	Database retreive
Data Handler	Reset Content Descriptor / Compute Node
Custom Mediation	Java Compute Node
SOAP Header	Compute Node
HTTP Header	HTTP Header
JMS Header	JMS Header
MQ Header	MQ Header
Message Logger	DatabaseInsert
Event Emitter	Monitoring framework provided
Trace	Trace
Stop	PassThrough the closest
Fail	Throw
MessageValidator	Validate node
Subflow	Subflow
Synchronous Transaction Rollback	SOAPReply wiring to a Throw will achieve the same

Agenda

- Application Development Concepts
- Capability Comparison
- **ESB Topology Discussion**
  - **WESB Physical Topology**
  - IIB Physical Topology

WebSphere Enterprise Service Bus Golden Topology

- The Golden Topology is the standard production template which is customized to meet a number of quality of service requirements. The particular setup used may remove the support cluster and load balancers but most customers use the below as a template.



Considerations

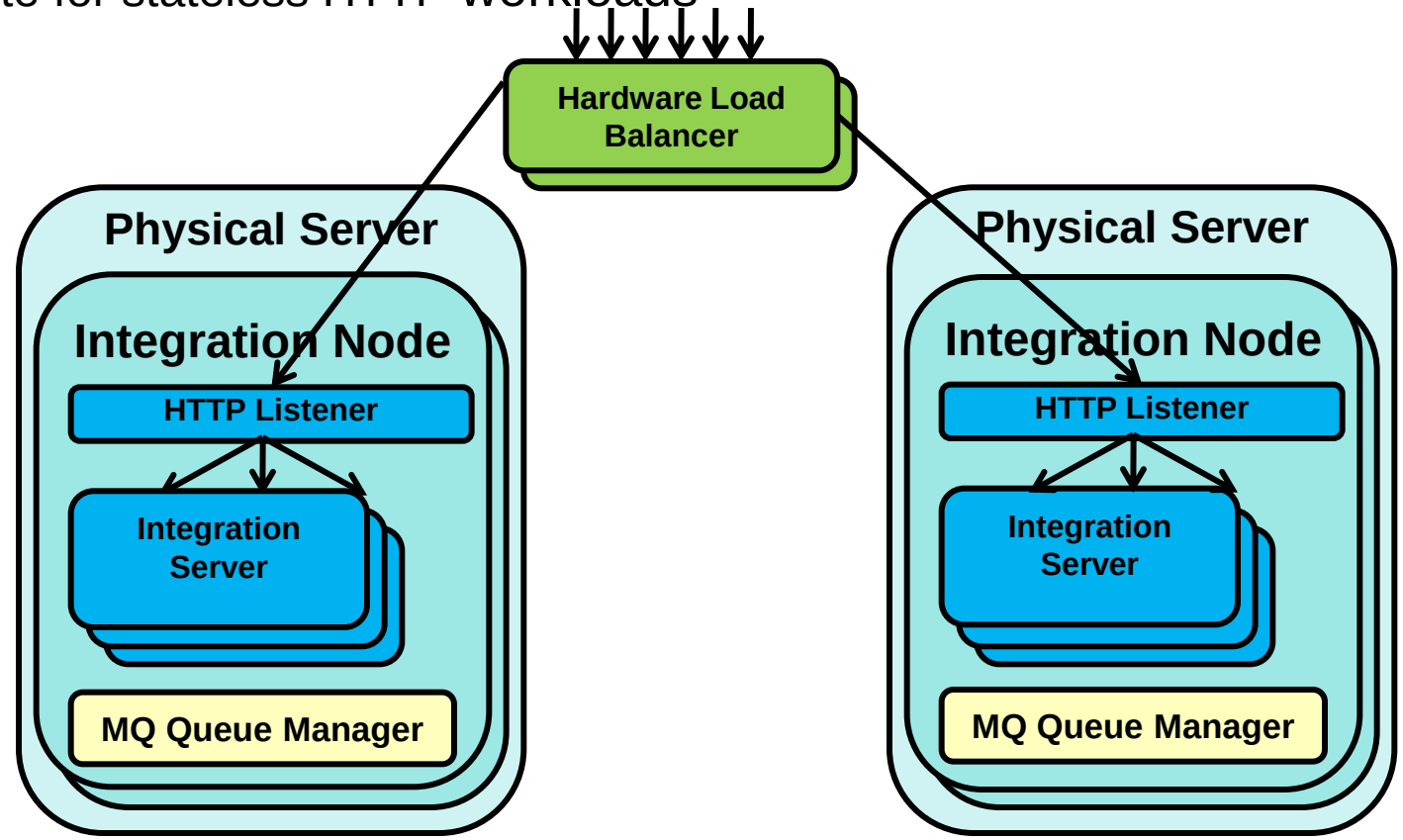
- High Availability of the solution
- Recover in-flight requests
- Asynchronous invocation styles
- Application isolation
- Application persistent requirements
- SIB JMS Bindings
- Asynchronous Bindings correlation information
- Long running integration logic
- Event Sequencing
- Store and forward

Agenda

- Application Development Concepts
- Capability Comparison
- **ESB Topology Discussion**
  - WESB Physical Topology
  - **IIB Physical Topology**

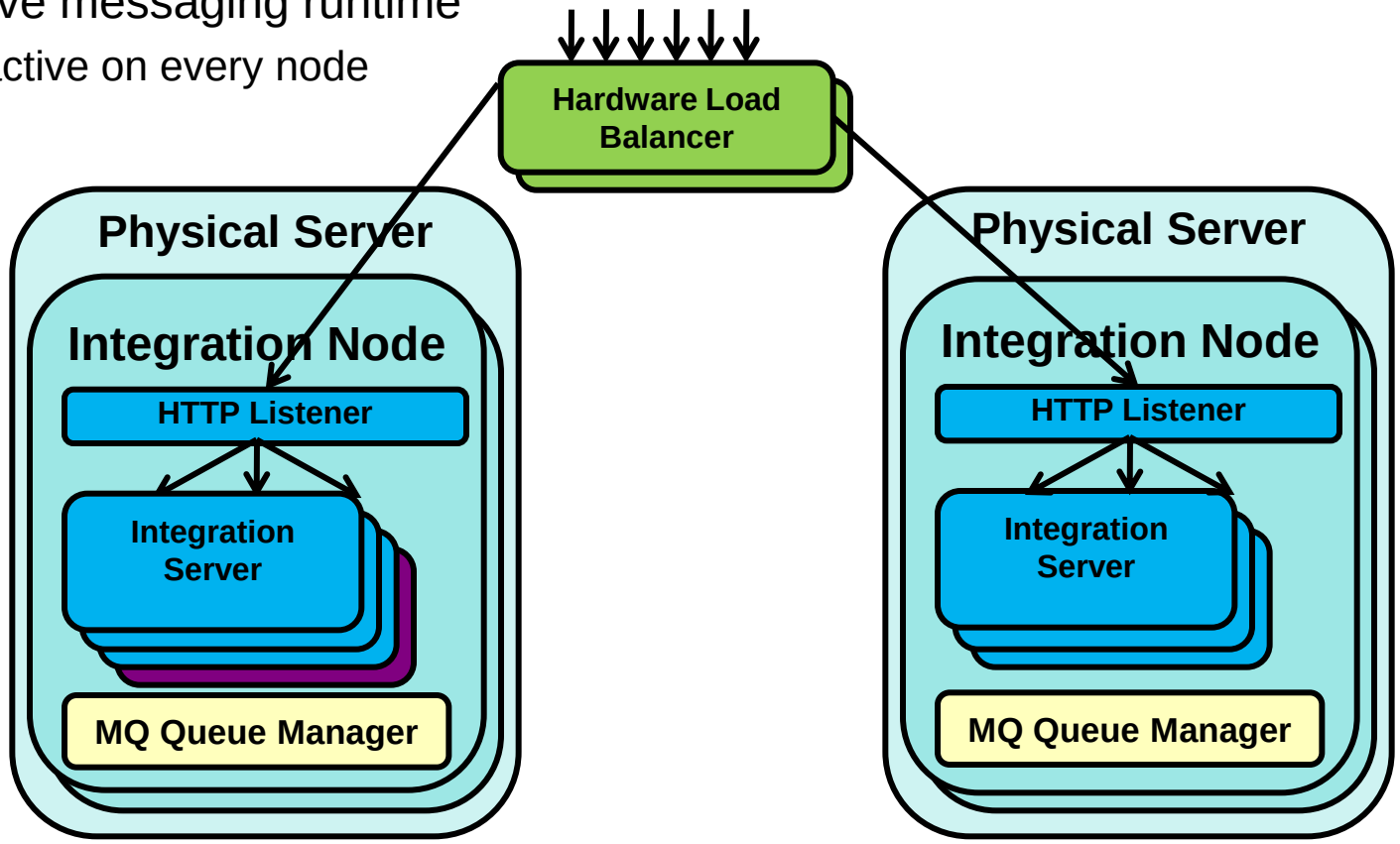
IBM Integration Bus Topology

- Build an IBM Integration Bus topology for *your* needs
  - Use the simplest topology and feature set that fits your needs
  - No single 'golden topology' recommendation
- The below represents a foundation starting point
  - Active-active infrastructure
  - Complete for stateless HTTP workloads



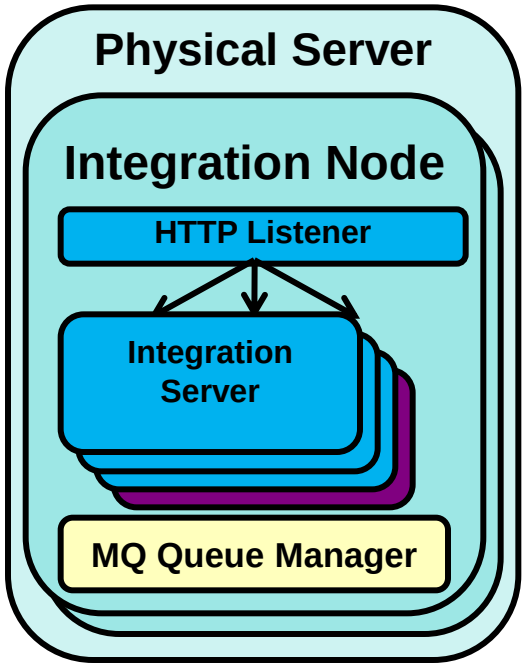
Notable differences to WESB golden topology

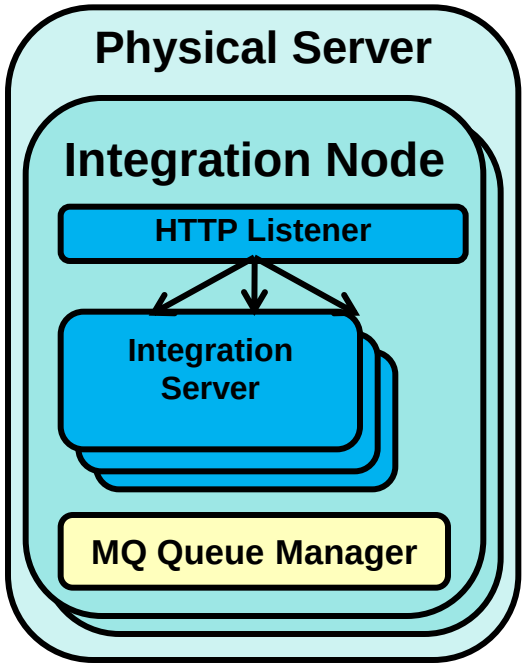
- Not all nodes are created equal
  - *Could* choose to have a different set of servers on each node
  - *Could* choose to have a different set of applications (or 'flows') on the servers of each node
  - Simple scripting interface provided for roll-out across a set of nodes
- HTTP distribution handled within the Integration Node
  - IHS can also be layered into the solution
- Active-active messaging runtime
  - MQ is active on every node



Stateful logic – overview of differences with WESB

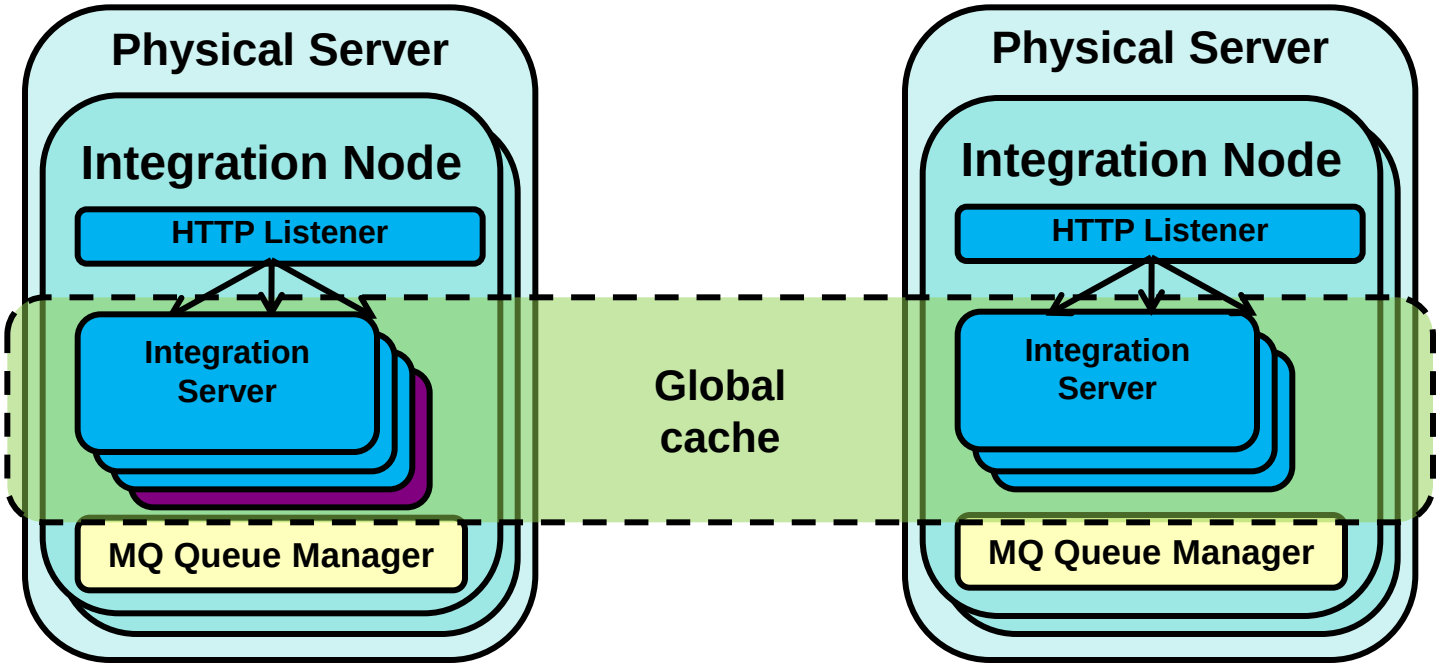
- State in WESB is commonly stored in the Messaging layer
  - A *single* highly available Messaging Engine that 'floats' across the servers (active-passive)
  - A Database used under the covers to make that state available across the cluster
- The simple starting point below does not have an equivalent
  - Each integration node has a different MQ queue manager (active-active)
- Let's take a look at the options…





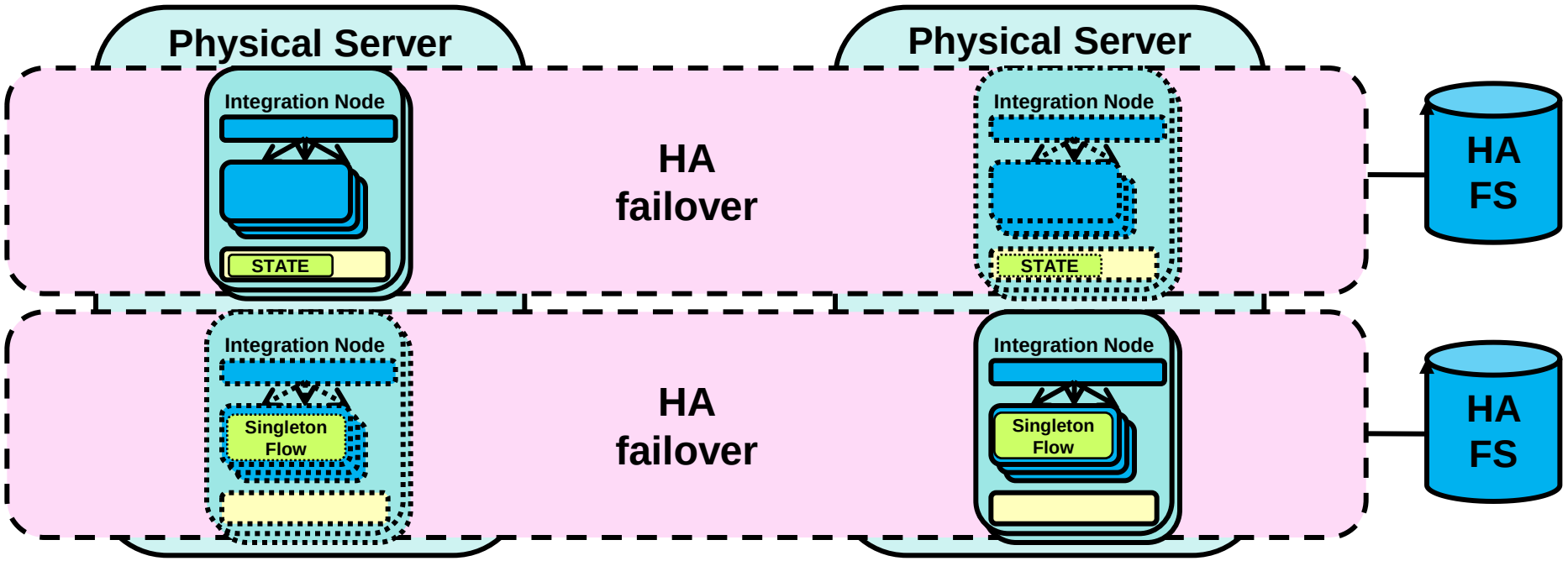
Stateful logic – option 1: Global cache

- High value feature of IBM Integration Bus
  - Uses IBM WebSphere eXtreme Scale technology
- Simple to configure as a high available, synchronously replicated, cache across all brokers
- Ideal for caching request/reply context
- Ideal for performance caching of state tables persisted to a file or DB



Stateful logic – option 2: HA failover

- Choose a HA failover technology
  - MQ multi-instance – Needs highly available network-attached storage (NAS)
  - HA clustering software (PowerHA, Veritias Cluster Server, MSCS etc.) – Direct fiber connection to SAN
- Host your MQ state queues and/or singleton flows in one active/passive integration node
  - Use JMS nodes to attach remotely to that MQ queue manager
- Solves other HA integration challenges that cannot be active/active. Examples:
  - Flows that listen for arrival files, and do not have special file locking
  - Flows with strict ordering requirements



Questions

IBM Copyrights and Trademarks Statement

Backup slides