Introduction to IBM Integration Bus on z/OS

Session 17887

10th August 2015

Geza Geleji (gezagel@uk.ibm.com)

Staff Software Engineer, IBM Integration Bus Development

David Coles (dcoles@uk.ibm.com)

Technical Lead, IBM Integration Bus Level 3 Service

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence.**

Important Disclaimer

Agenda

- Introduction
- Inside IBM Integration Bus
- Development
- Administration
- Developer Edition
- Industry Solutions
- Key Usage Scenarios
- Questions?

Introduction: what do we mean by *integration*?

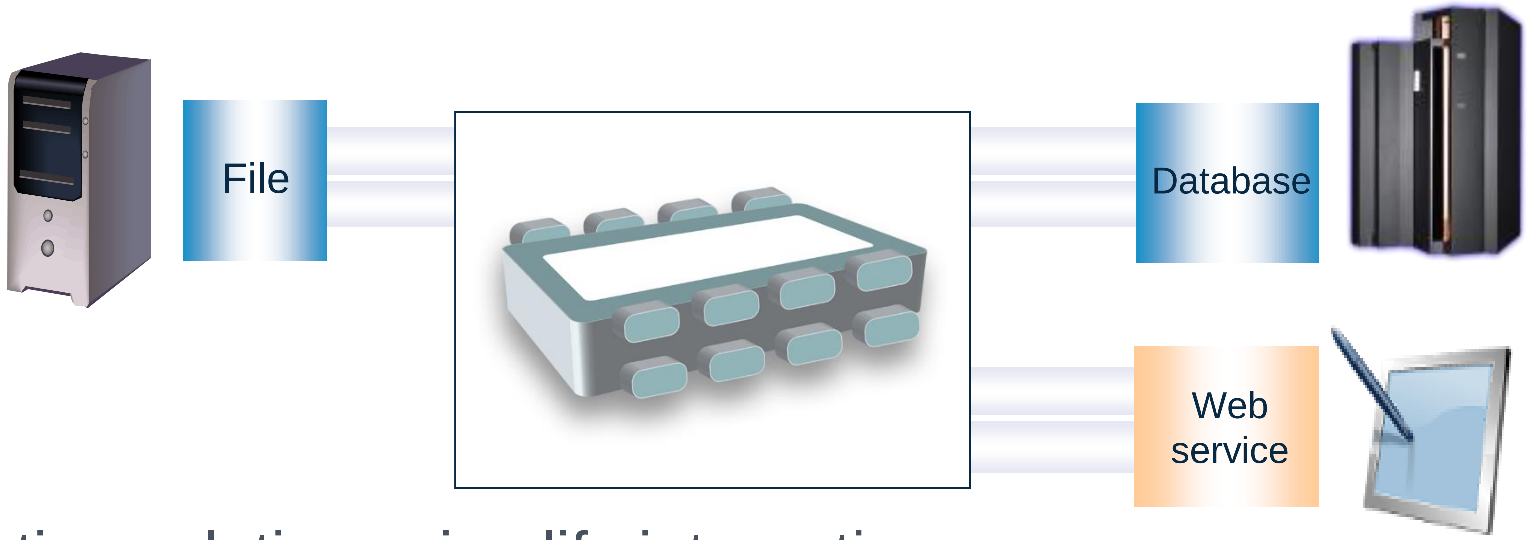
- enterprise systems consist of many logical **endpoints**
  - off-the-shelf applications, services, web apps, devices, appliances, custom built software
- endpoints expose a set of inputs and outputs, which comprise
  - protocols: MQ, TCP/IP, HTTP, file system, FTP, SMTP, POP3, etc.
  - message formats: binary (C / COBOL), XML, industry (SWIFT, EDI, HL7), user-defined
- integration is about connecting these endpoints together in meaningful ways
  - Route, Transform, Enrich, Filter, Monitor, Distribute, Decompose, Correlate, Fire and Forget, Request/Reply, Publish/Subscribe, Aggregation, Fan-in, Complex Event Processing, etc.

Notes for slide 4

Three strands are involved in connecting applications together.

1. Applications need to talk with each other over a communications protocol. Typical protocols in use today include TCP/IP, and higher level protocols such as FTP, SMTP and HTTP.
2. Over the communications protocol applications exchange data, typically in discrete structures known as messages. The format of these messages can be defined from C structures or COBOL copybooks (for example), or simply use a standard format such as XML.
3. In order to connect applications together so that their protocols and message formats interoperate, mediation patterns need to be applied to one or both systems you're trying to connect. These mediation patterns can be relatively straightforward, e.g. routing messages from one place to another, or the transformation of one message format into another… to relatively complex patterns such as aggregating multiple outputs from an application into a single message for a target system.

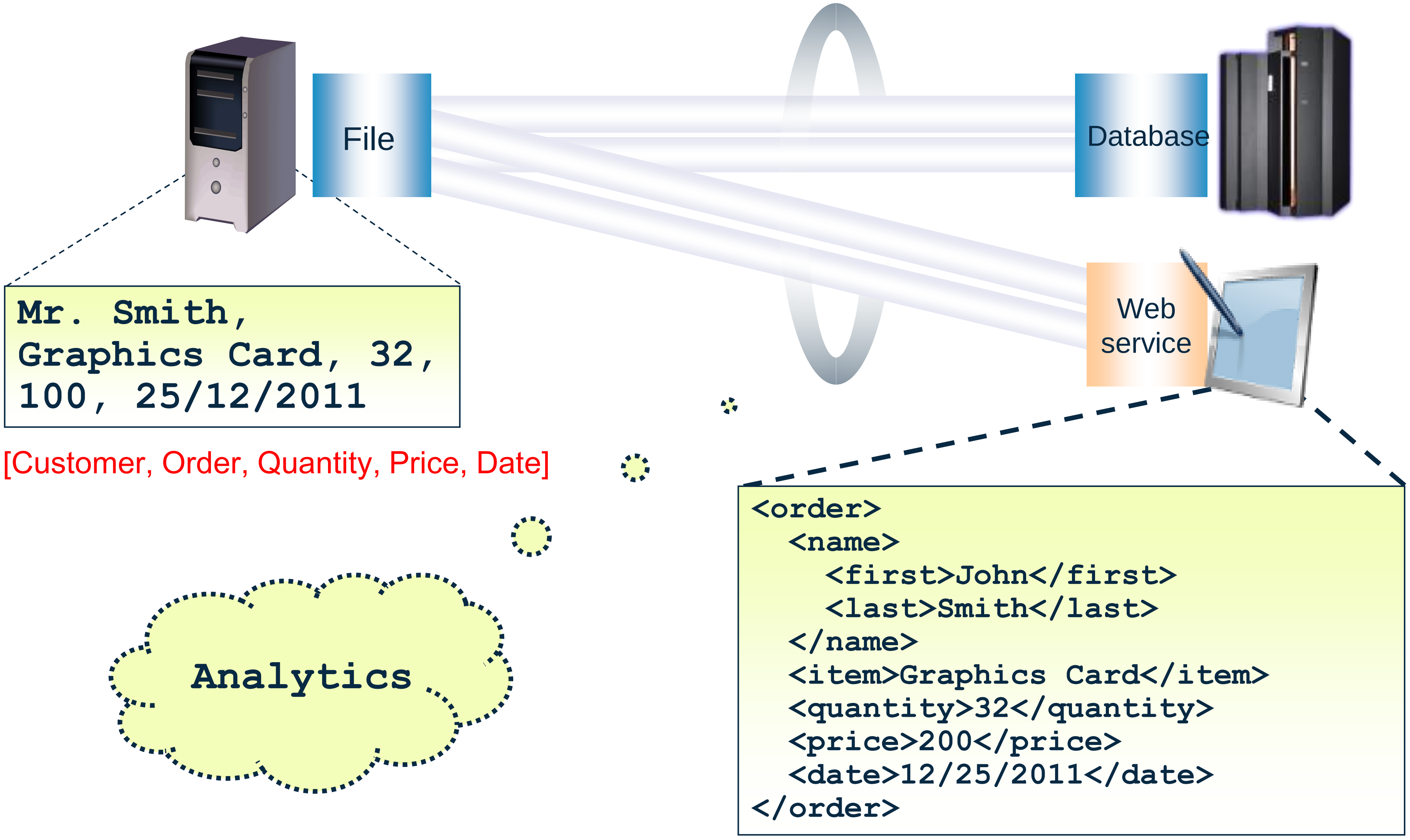
Integration solutions: reducing cost



Integration solutions simplify integration

- avoid rewrites in response to new integration requirements
- simplify maintenance by reducing expensive coupling
- flexibility adding anonymity between producers and consumers of data
- add insight into applications and business value they bring

Example integration



Mr. Smith,
Graphics Card, 32,
100, 25/12/2011

[Customer, Order, Quantity, Price, Date]

```
<order>
  <name>
    <first>John</first>
    <last>Smith</last>
  </name>
  <item>Graphics Card</item>
  <quantity>32</quantity>
  <price>200</price>
  <date>12/25/2011</date>
</order>
```

[Customer, Order, Quantity, Price, Date]

Notes for slide 7

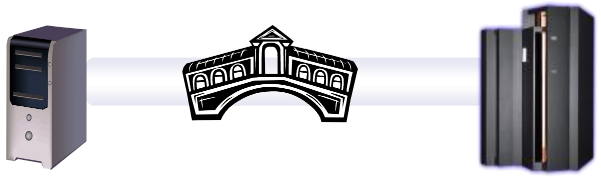
This chart describes an application integration scenario.

1. Application A sends some data to application B. At design time, the two applications agreed on the format of the data as the ordered set {Customer, Order, Quantity, Price, Date}. Further, the date is in UK format, the price in UK pounds sterling, and all fields are represented by character strings in code page 500. Finally, the data is delimited using commas.
2. Some time later, Application C is introduced. It needs the same data, but because it is a packaged application from a vendor or may be an application that already existed, it expects data to arrive in a different format. The date is in US format, the price is in dollars and the data is in XML.

So, we now have an integration choice to make. Either application C must be enhanced to support the data format between A and B, or application A must be enhanced to support application C's data format. (This is an interesting use of the word “enhanced”, but you'll probably want to use it to justify the expenditure!)

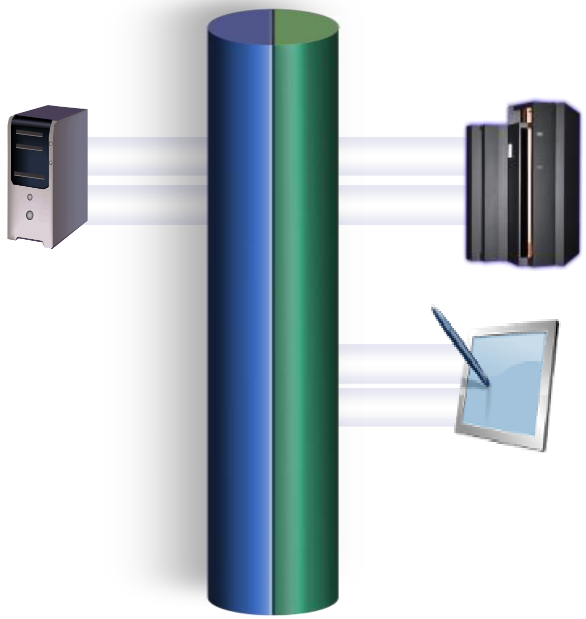
- By introducing a solution that can mediate between these applications, you can integrate them without spending time and money modifying and retesting the existing applications. IBM Integration Bus is one such solution.
- In addition, a solution like IBM Integration Bus will also allow you to intercept or record the data as it is processed, allowing you to satisfy audit requirements or for further analysis for use cases like fraud prevention.

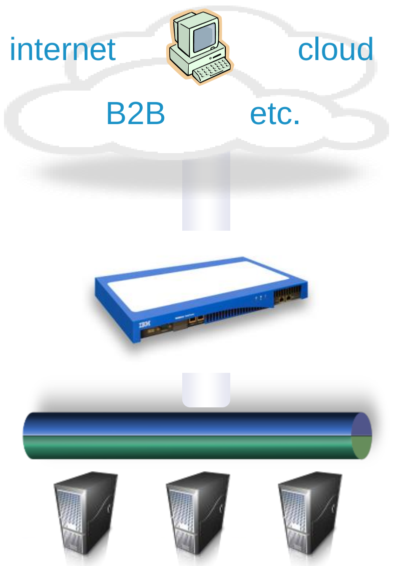
Some examples of integration topologies



Bridges

- often used for single point-to-point connections
- usually cheap and quick to configure
- more difficult to scale to larger numbers of endpoints





Gateways

- provides connectivity to third parties or to a specific class of endpoint
  - e.g.: internet, cloud, security, DMZ, B2B
- simplicity of configuration
- commonly on-ramp to back-end ESB

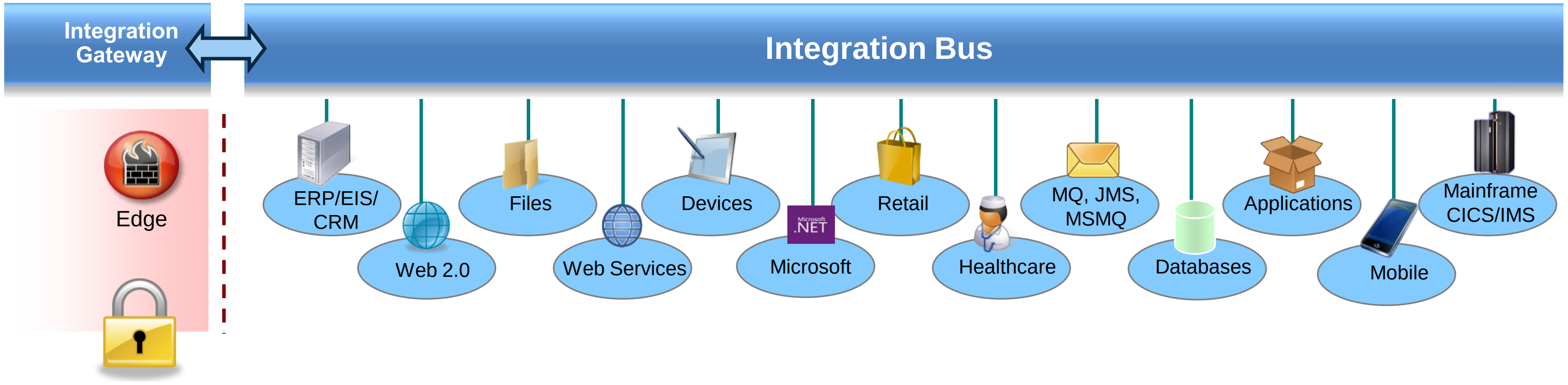
Enterprise Service Bus (ESB)

- logical construct that combines messaging and enrichment
- scales very well; can integrate small and large numbers of endpoints, and can be easily distributed
- often applied as a backbone for a Service Oriented Architecture (SOA)
- solutions can usually also be applied to hub and spoke style architectures

Introducing IBM Integration Bus

IBM's Strategic Integration Technology

- single engineered product for .NET, Java and fully heterogeneous integration scenarios
- DataPower continues to evolve for integration gateway use cases



IBM Integration Bus is the new name for WebSphere Message Broker

- technology progression over 15 years, installed at 2500+ customers worldwide across all industries
- fully supported worldwide by IBM global support network, standard 5 + 3 years support policy
- version to version migration is key design consideration
- global skills availability - SME's available globally via IBM and partners
- close interaction with growing and loyal customer base: beta and lab advocacy programs
- also incorporates WebSphere ESB use-cases

IBM Integration Bus

- provides endpoints and the ability to connect to other endpoints
  - off-the-shelf applications, services, web apps, devices, appliances, custom built software
- protocols and message formats
  - MQ, TCP/IP, HTTP, file system, FTP, SMTP, POP3 etc.
  - binary (C/COBOL), XML, industry (SWIFT, EDI, HL7), user-defined
- mediation patterns
  - Route, Transform, Enrich, Filter, Monitor, Distribute, Decompose, Correlate, Fire and Forget, Request/Reply, Publish/Subscribe, Aggregation, Fan-in, Complex Event Processing

Notes for slide 11

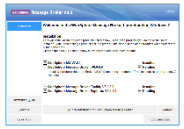
We can now revisit this earlier slide in the context of IBM Integration Bus.

- IBM Integration Bus enables “universal connectivity” by integrating protocols, message formats and mediation patterns.
- IIB provides the ability to be an endpoint and to connect to other endpoints. It can do this over a variety of protocols and using a variety of message formats, sometimes with more than one in use at a time.
- IIB supports a wide range of mediation patterns, helping to support the use of the various message formats and protocols in many ways.
- As we go through the rest of this presentation we will see how IBM Integration Bus supports all of these.

Broad Platform and Environment support

Broad range of operating system and hardware platforms supported

- AIX, Windows, z/OS, HP-UX, Linux on xSeries, pSeries, zSeries, Solaris (x86-64 & SPARC), Ubuntu
- optimized 64-bit support on all platforms; 32-bit option available for Windows and x/Linux
- support for Windows 8 and Windows Server 2012; .NET CLR V4.5 included on Windows
- Express, Standard and Advanced editions make IIB applicable for all solutions and budgets


Traditional OS

IBM Workload Deployer

SOFTLAYER®
Public Cloud

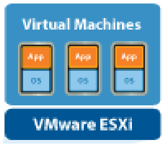
IBM Pure

Virtual images for efficient utilization & simple provisioning

- extensive support for virtualized environments, e.g. VMWare, AIX Hypervisor… any!
- support for public and private clouds: SoftLayer, Pure, non-IBM, RYO etc.
- chef scripts for automated building of flexible IIB images (see GitHub)
- pre-built images (Hypervisor editions) available on xLinux and AIX

Includes access to full range of industry standard databases and ERP systems

- DB2, Oracle, Sybase, SQL Server, Informix, solidDB
- Open Driver Manager support enables new ODBC databases to be accessed
- JDBC Type 4 for popular databases
- SAP, Siebel, PeopleSoft, JDEdwards

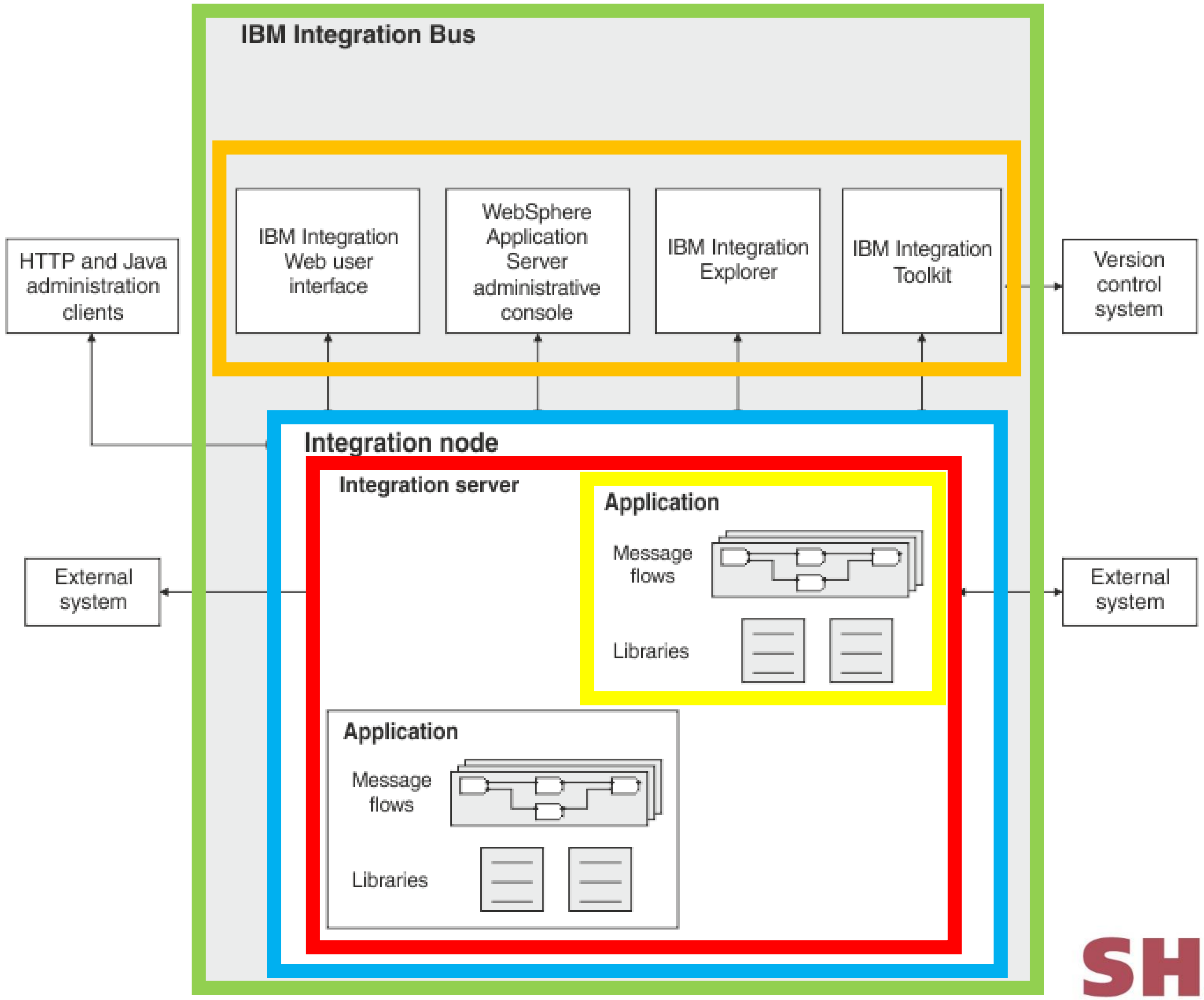


Private Cloud

Technology components and pre-requisites (@v9)

- Java 7 on all platforms
- MQ 7.1 prerequisite

What's inside?

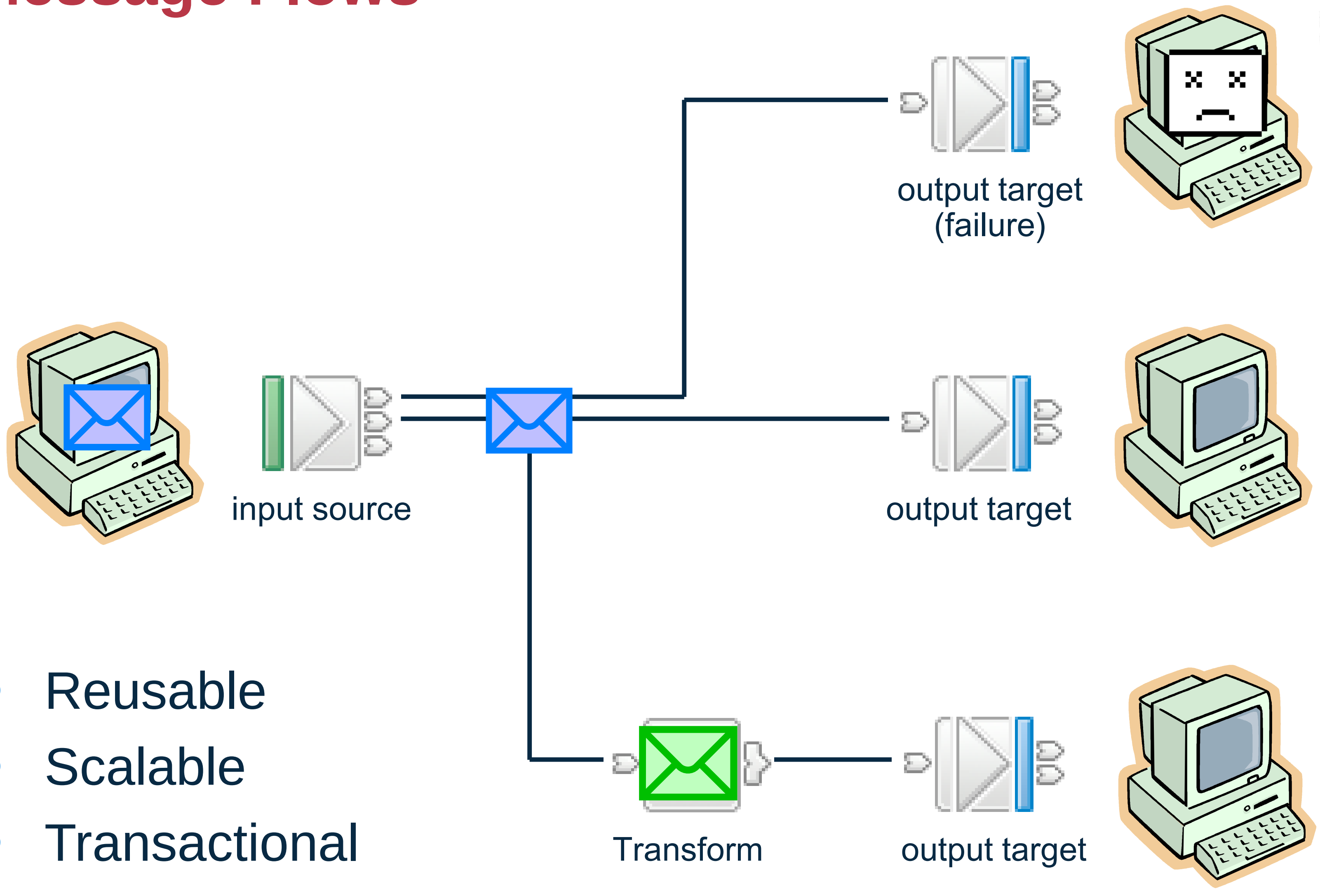


Notes for slide 14

We’ll start off by taking a look at the architectural components of IBM Integration Bus. We’ll then see how these elements are used in more detail.

- The Integration Toolkit is the development environment. Based on the Eclipse platform, all the objects required to perform application integration using IIB are developed, deployed and tested here. It provides standard ways to build integration applications, perform version control and provide for the development of custom plug-ins, such as resource editors to allow users to create project resources easily. Examples are custom editors to aid flow creation, ESQL editing and syntax checking, message set modelling, and a raft of other activities. It includes a unit test broker environment.
- The integration node (or broker) is the container that hosts integration servers (or execution groups). Each server is an operating system process that contains a pool of threads responsible for running the integration logic that is deployed to it. The integration servers directly interact with the endpoints that are being integrated.
- There is also a web user interface that provides administration capability, including monitoring of deployed objects and the ability to start, stop, delete, deploy, manage workloads etc. The APIs that the web UI uses can be used by custom administration applications, either through Java, REST or command line scripts.

Message Flows



- Reusable
- Scalable
- Transactional

Notes for slide 16 (1 of 3)

Message flows provide the processing sequence required to connect applications together.

- A message flow contains the set of operations required to take a message from an originating application and deliver copies of it, some possibly transformed, to any number of connected applications for processing.
- As a message passes through a message flow, it is transformed and routed according to the nodes it encounters, and the processing decisions made within those nodes. Later we'll see how the nodes can modify the values within, or transform the structure of, a message to provide the data transformations necessary to drive back-end server applications.
- For a given application scenario, the message flow describes all possible outcomes when processing a message. For example, if the message has a high monetary value, a copy of it might have to be routed to an audit application. Or if the message is not well-formed (maybe it's not encrypted in the right format), it might be routed to a security application to raise an alert.
- Equally important is the visualization of the application integration within then organization. Very often, for any particular application scenario, the application connectivity requirements (*business!*) are held within the heads of domain experts. Being able to view the integration structure brings benefits in scenario understanding, reuse potential, and application architecture/standards conformance.
- After a message has been processed by a message flow, the flow does not maintain any state. It is possible to maintain such state in an external database, or within the message by using an extensible header such as the MQRFH2 or message properties.

Notes for slide 16 (2 of 3)

Message flows are general purpose, reusable integration applications.

- If you were designing a general purpose integration application, linking client and server applications, the logic would comprise separate routines, each performing a well-defined function. The input routine would wait for a message, and after receiving it and checking its integrity (well formed etc.), it would transfer to the next routines to continue processing.
- After performing their processing, (e.g. enriching / reformatting / routing), these routines would pass control on through to the lowest functional levels, where output processing would occur. Here, messages would be written to devices, subsequently read by connected applications. At any level of processing an exception could be raised for subsequent processing.
- After the last output routine had completed, control would return back up through the levels to the input routine. Once here, all the changes would be committed and the input routine would wait for more input.

Notes for slide 16 (3 of 3)

Message flows are transactional.

- Message flows provide vital processing and data manipulation and are therefore fully transactional. A message flow either completes all or none of its processing successfully.
- However, if required, individual nodes can elect to perform operations outside of the message flow transaction. (e.g. audit)

Message flows are multithreaded.

- A given message passing through a series of nodes will execute on a single thread. To allow increased message throughput, message flows can be defined with many additional threads assigned to them. Peak workloads use additional threads, which are pooled during inactivity. We'll see more implementation details later. This means application scaling can be an operational rather than design time decision.

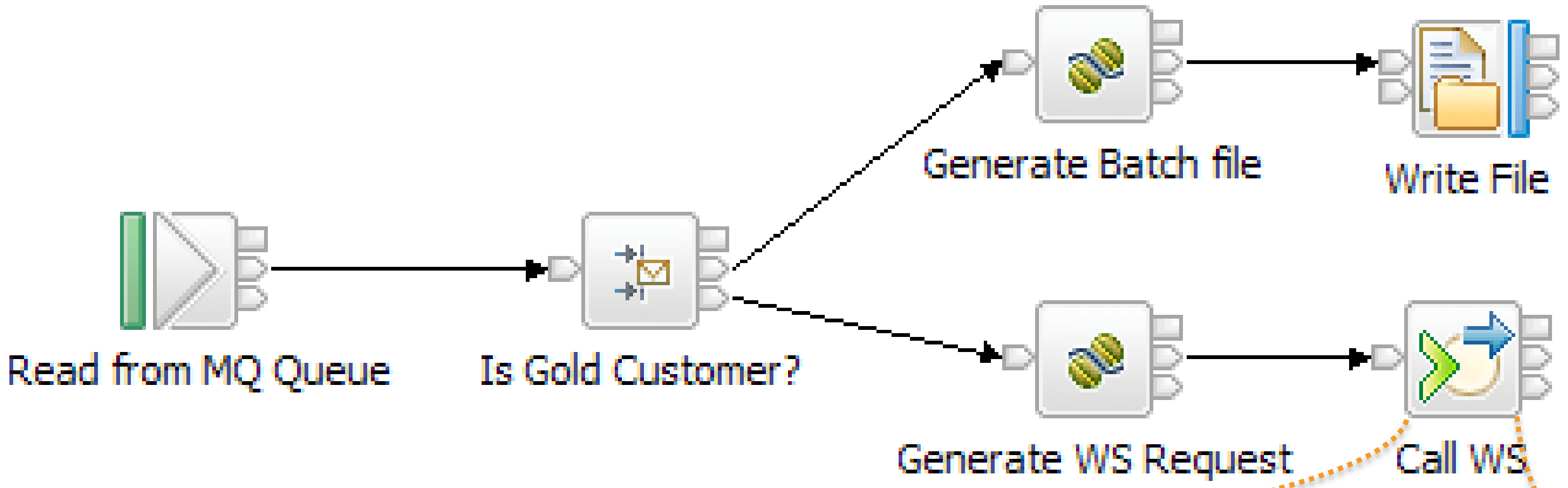
Message flow nesting and chaining allow construction of enhanced capabilities.

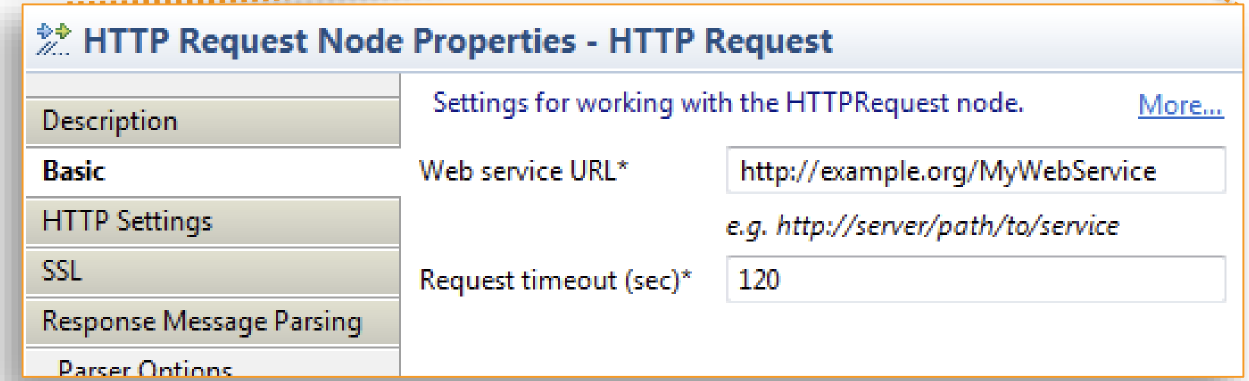
- Sophisticated flows can be rapidly constructed by linking individual flows together as well as nesting flows within each other.

References:

1. Message Flow overview at http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac00310_.htm

Message Flow example





Settings for working with the HTTPRequest node.

Web service URL* http://example.org/MyWebService

e.g. http://server/path/to/service

Request timeout (sec)* 120

Notes for slide 20

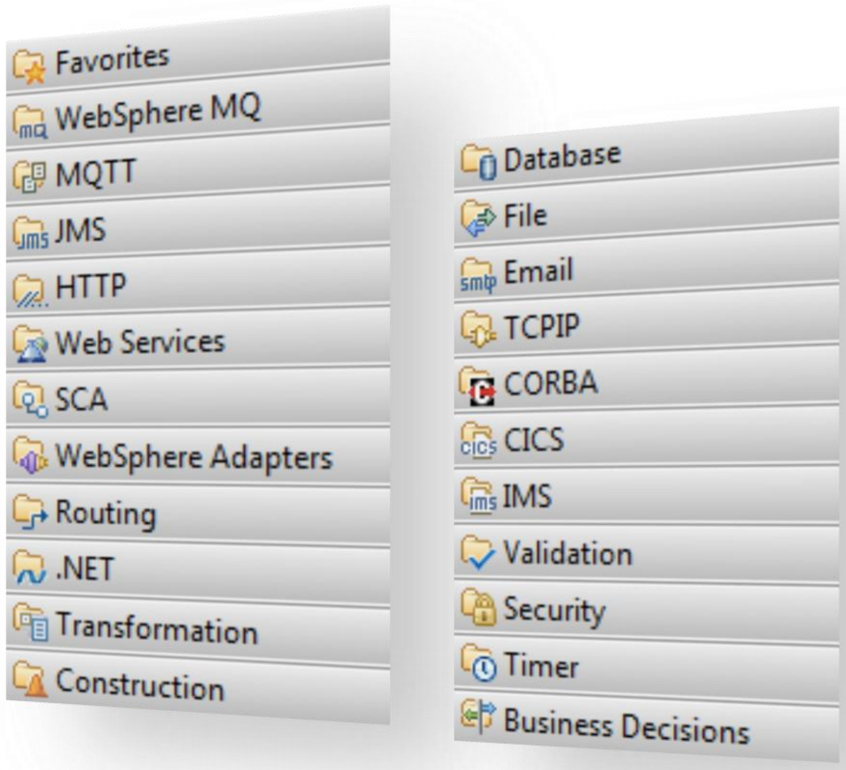
Here is an example of a message flow.

- The ‘Read from MQ Queue’ node tells IIB to take messages from an MQ queue (the name of which is embedded as a property of the node, or overridden by an administrator at deployment time).
- The message is passed onto the ‘Is Gold Customer?’ node, where a routing decision is made based on a field described in the incoming message, again which is a property on the node itself. We’ll see exactly how this condition is specified later on.
- If the described condition holds, the message is routed to the ‘Generate WS Request’ node where the message is transformed – presumably into an SOAP message that is recognisable by the web service which is invoked by the subsequent ‘Call WS’ node.
- If the described condition does not hold, the message is routed to the ‘Generate batch file’ node, which formats the message for subsequent output to a file in the ‘Write file’ node.

This flow may not tell the complete integration story between the calling application and the target Web Service/File applications. For example, there is no communication back to the calling application to say that the message has been processed (or even received). Nor is there any logic in the message flow to cope with failures – for example, if the web service is not available. This is logic that could be incorporated into the flow, but not visualised here for clarity.

Nodes

- the building blocks of message flows
- each node type performs a different (input, output or processing) action
- many different node types
  - – grouped into logical categories in the editor
  - – nearly 100 nodes available out-of-the-box



Notes for slide 22

Nodes can be grouped in several ways; for example, by where in the flow they are used:

- Input nodes do not have input terminals; processing of the message flow starts when a message is retrieved from an input device, for example WebSphere MQ.
- Output nodes do not have output terminals (or at least, they are not wired to any other node). The final stage of output processing is after a message is put using one or more output nodes, and processing control returns to the input node which commits or backs out the transaction. Recalling that a message flow is analogous to a functional decomposition, it makes sense that the top most level (i.e. the input node) controls the overall transaction.
- Processing nodes are nodes that are neither input nor output nodes. They will be connected to nodes both upstream (i.e. towards the input nodes) and downstream (i.e. towards the output nodes).

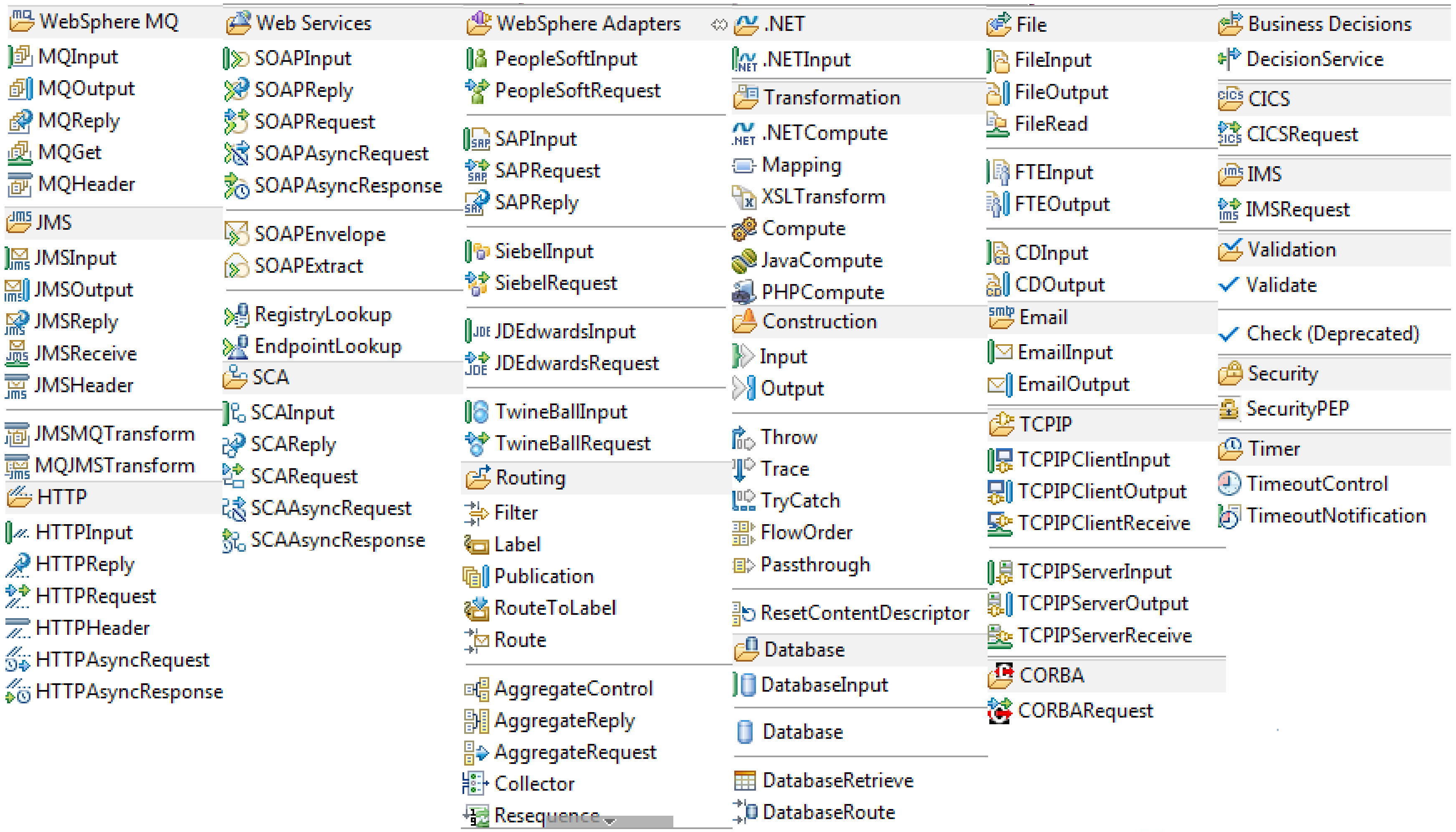
They can also be grouped by the function that they perform:

- Protocol-specific nodes give the broker the ability to interact with particular systems, such as MQ and Web Services.
- Transformation nodes will take a message in one format on the input terminal and output a converted message on the output terminal.
- Logical constructs give the message flow designer the vocabulary required to solve complex integration scenarios, for example, the ability to aggregate messages from multiple places or the ability to filter messages based on their content.

References:

- More on nodes can be found here: http://www-01.ibm.com/support/knowledgecenter/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac04550_.htm

Message Flow Nodes



WebSphere MQ
- MQInput
- MQOutput
- MQReply
- MQGet
- MQHeader

JMS
- JMSInput
- JMSOutput
- JMSReply
- JMSReceive
- JMSHeader
- JMSMQTransform
- MQJMSTransform

HTTP
- HTTPInput
- HTTPReply
- HTTPRequest
- HTTPHeader
- HTTPAsyncRequest
- HTTPAsyncResponse

Web Services
- SOAPInput
- SOAPReply
- SOAPRequest
- SOAPAsyncRequest
- SOAPAsyncResponse
- SOAPEnvelope
- SOAPExtract
- RegistryLookup
- EndpointLookup

SCA
- SCAInput
- SCAReply
- SCARequest
- SCAAsyncRequest
- SCAAsyncResponse

WebSphere Adapters
- PeopleSoftInput
- PeopleSoftRequest
- SAPInput
- SAPRequest
- SAPReply
- SiebelInput
- SiebelRequest
- JDEdwardsInput
- JDEdwardsRequest
- TwineBallInput
- TwineBallRequest

Routing
- Filter
- Label
- Publication
- RouteToLabel
- Route
- AggregateControl
- AggregateReply
- AggregateRequest
- Collector
- Resequence

.NET
- .NETInput

Transformation
- .NETCompute
- Mapping
- XSLTransform
- Compute
- JavaCompute
- PHPCompute

Construction
- Input
- Output
- Throw
- Trace
- TryCatch
- FlowOrder
- Passthrough
- ResetContentDescriptor

Database
- DatabaseInput
- Database
- DatabaseRetrieve
- DatabaseRoute

File
- FileInput
- FileOutput
- FileRead
- FTEInput
- FTEOutput
- CDInput
- CDOutput

Email
- EmailInput
- EmailOutput

TCPIP
- TCPIPClientInput
- TCPIPClientOutput
- TCPIPClientReceive
- TCPIPServerInput
- TCPIPServerOutput
- TCPIPServerReceive

CORBA
- CORBARequest

Business Decisions
- DecisionService

CICS
- CICSRequest

IMS
- IMSRequest

Validation
- Validate
- Check (Deprecated)

Security
- SecurityPEP

Timer
- TimeoutControl
- TimeoutNotification

Notes for slide 24 (1 of 2)

Here's a list of the protocol specific nodes built in to IBM Integration Bus V9. For example:

- The WebSphere MQ nodes allows Integration Broker to interact with queues on MQ Queue Managers. For example, MQInput is an input node that triggers a flow when a message arrives on a queue; MQOutput puts a message to a queue.
- The WebSphere Adapters nodes provides native support in Integration Bus for inbound and outbound communication with Enterprise Information Systems.
- Web Services nodes provide a rich environment for running as a Web Services requestor, provider and intermediary. Support for WS-Security, WS-Addressing, import and export of WSDL and validation against the WS-I Basic profile. The RegistryLookup and EndpointLookup nodes provide support for WebSphere Registry and Repository (WSRR).
- The File nodes are very sophisticated and include support for the FTP and SFTP protocols, as well as advanced processing scenarios such as record detection. These nodes are complemented by additional nodes which provide support for managed file transfer systems (IBM's MQ File Transfer Edition and Sterling Connect:Direct).
- HTTP nodes complement the Web Services capability. Support is provided for HTTP 1.0, 1.1 and HTTPS.
- JMS nodes work with any JMS 1.1 compliant provider.
- The EmailOutput node is a highly configurable node that allows e-mail messages to be sent over the SMTP protocol. EmailInput allows e-mails to be received from POP3 or IMAP servers.
- TCP/IP nodes allow the Integration Bus to communicate with any client or server talking the ubiquitous TCP/IP protocol.
- CORBA, IMS and CICS request nodes for integrating with CORBA, IMS and CICS applications respectively.

Notes for slide 24 (2 of 2)

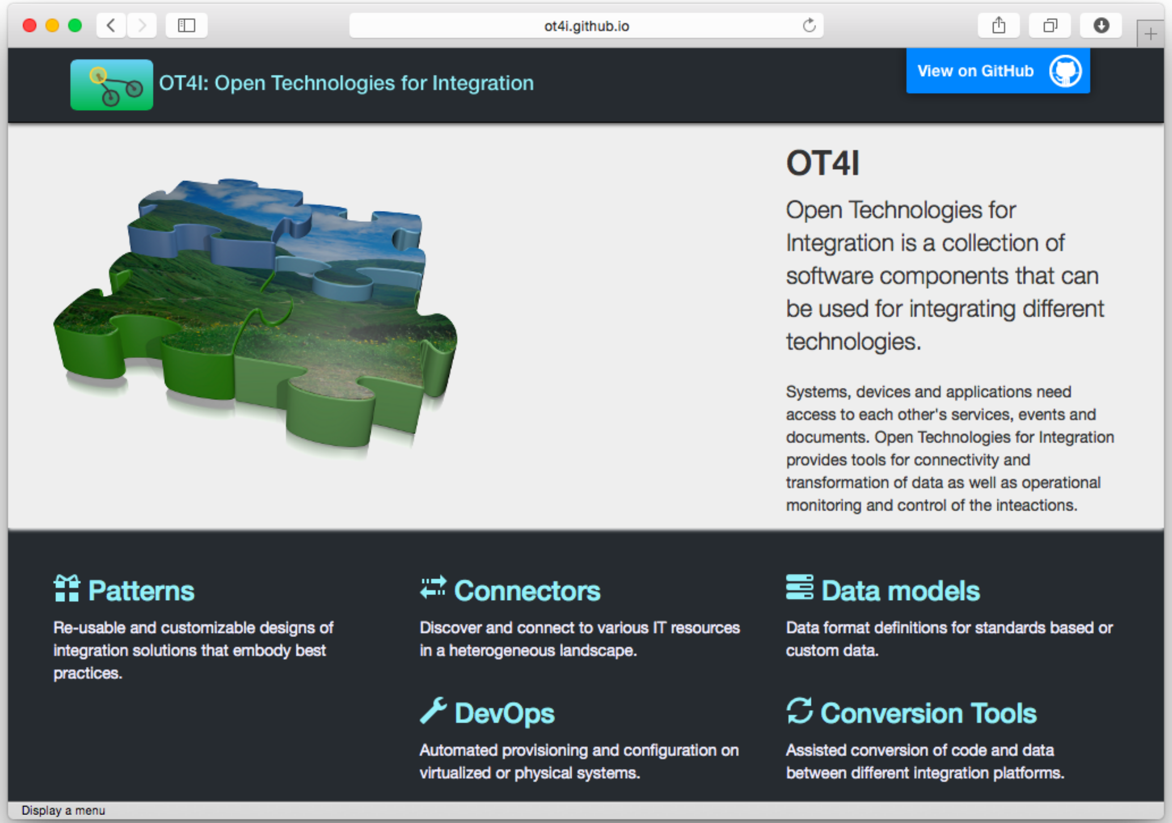
- Database nodes allows message flows to interact with many different data sources, including DB2, Oracle and Sybase.
- Timer nodes provide support for triggering message flows and certain times or intervals.
- The Routing category allows messages to easily flow around a network, and also allow multiple messages to be aggregated or propagated in the correct sequence.

The Transformation category provides Integration Bus with the capability to transform messages from one format into another. Six ways of doing this are available out-of-the-box. More on these later.

Construction nodes:

- Nodes have error handling as part of their design. If an error is detected within a primitive node (e.g. database error), the message is transferred to the failure output terminal. If the failure terminal is not connected, an exception is generated and propagated back towards the input node. There is also a specialized Throw node which allows a flow designer to generate an exception in a controlled way. Nodes can have transaction scope inside or outside of the flow.
- A TryCatch node is used to process any such exceptions. Its ‘try’ terminal is used for normal processing, but if an exception occurs along this path, the TryCatch node regains control and the original message is propagated through the ‘catch’ terminal.
- If the message reaches the input node, it is subject to "back out" processing. In this case, it will be propagated down its catch or failure terminal, returned to the input queue, put to a back out or dead letter queue, or discarded, as appropriate.

IBM and third-party extensions


ot4i.github.io

OT4I: Open Technologies for Integration

View on GitHub

OT4I

Open Technologies for Integration is a collection of software components that can be used for integrating different technologies.

Systems, devices and applications need access to each other's services, events and documents. Open Technologies for Integration provides tools for connectivity and transformation of data as well as operational monitoring and control of the inteactions.

Patterns
Re-usable and customizable designs of integration solutions that embody best practices.

Connectors
Discover and connect to various IT resources in a heterogeneous landscape.

Data models
Data format definitions for standards based or custom data.

DevOps
Automated provisioning and configuration on virtualized or physical systems.

Conversion Tools
Assisted conversion of code and data between different integration platforms.

Display a menu

V3.0.0.1	V1.0.0.0	V1.0.0.0
Aug 2014	Jun 2014	Dec 2013

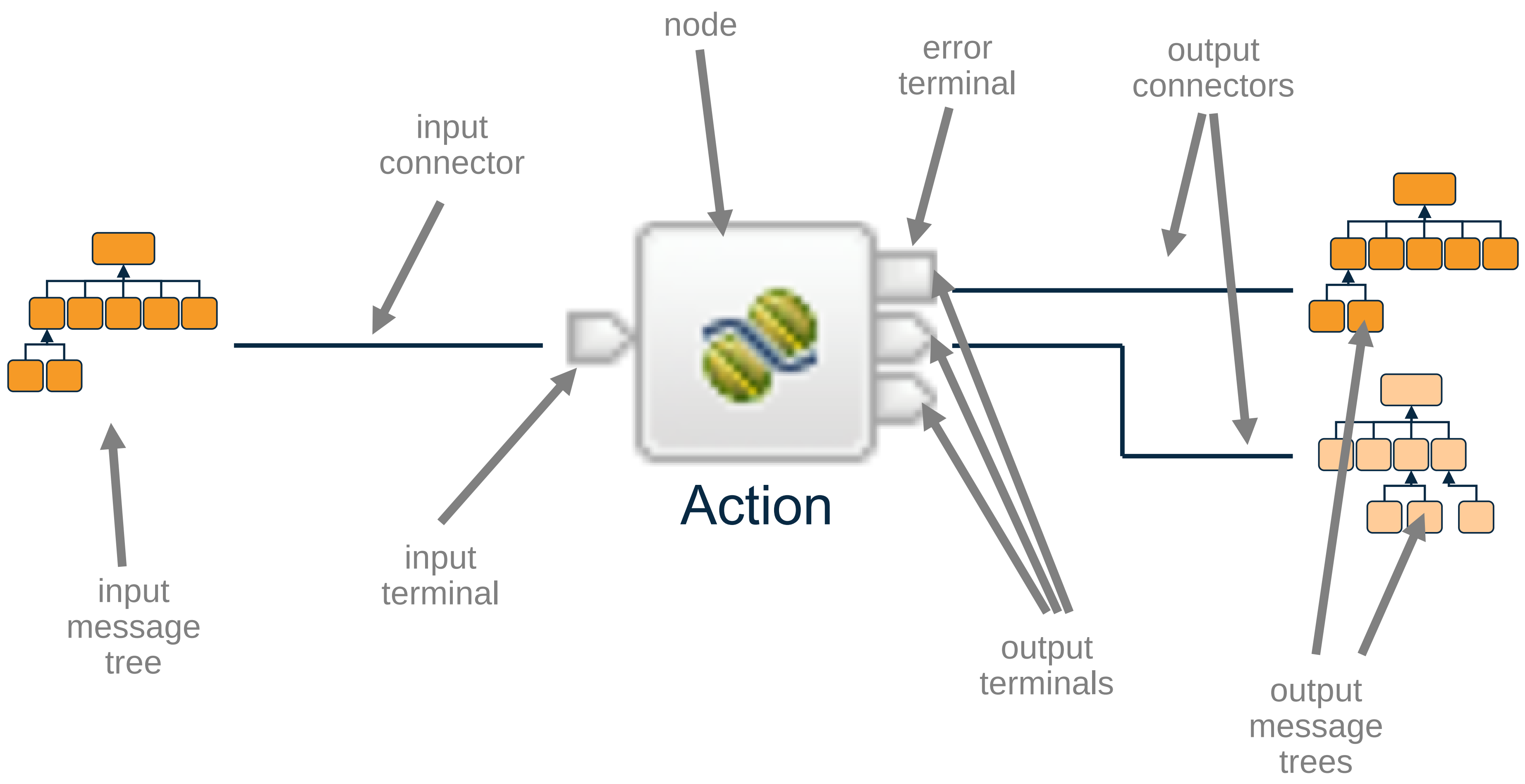
Many other nodes and features available through product extensions

- Tibco RV, VSAM, QSAM

Write your own nodes

- native node framework available in C and Java
- OT4i connector framework provides means to implement full lifecycle, including endpoint discovery

Node Terminology



Notes for slide 28 (1 of 2)

Message flow nodes provide the individual processing elements that make up a message flow.

We've seen that a message flow is the combination of operations required to achieve application integration. We build a message flow from small units called nodes; these nodes represent the base elements required to connect messaging applications together.

Looking at a message flow, you can see several objects identifiable with this processing.

- Nodes represent functional routines encapsulating integration logic
- Terminals represent the various outcomes possible from node processing
- Connectors join the various nodes through their terminals

A message processing node defines a single logical operation on a message.

A message processing node is a stand alone procedure that receives a message, performs a specific action against it, and outputs zero or more messages as a result of the action it has taken.

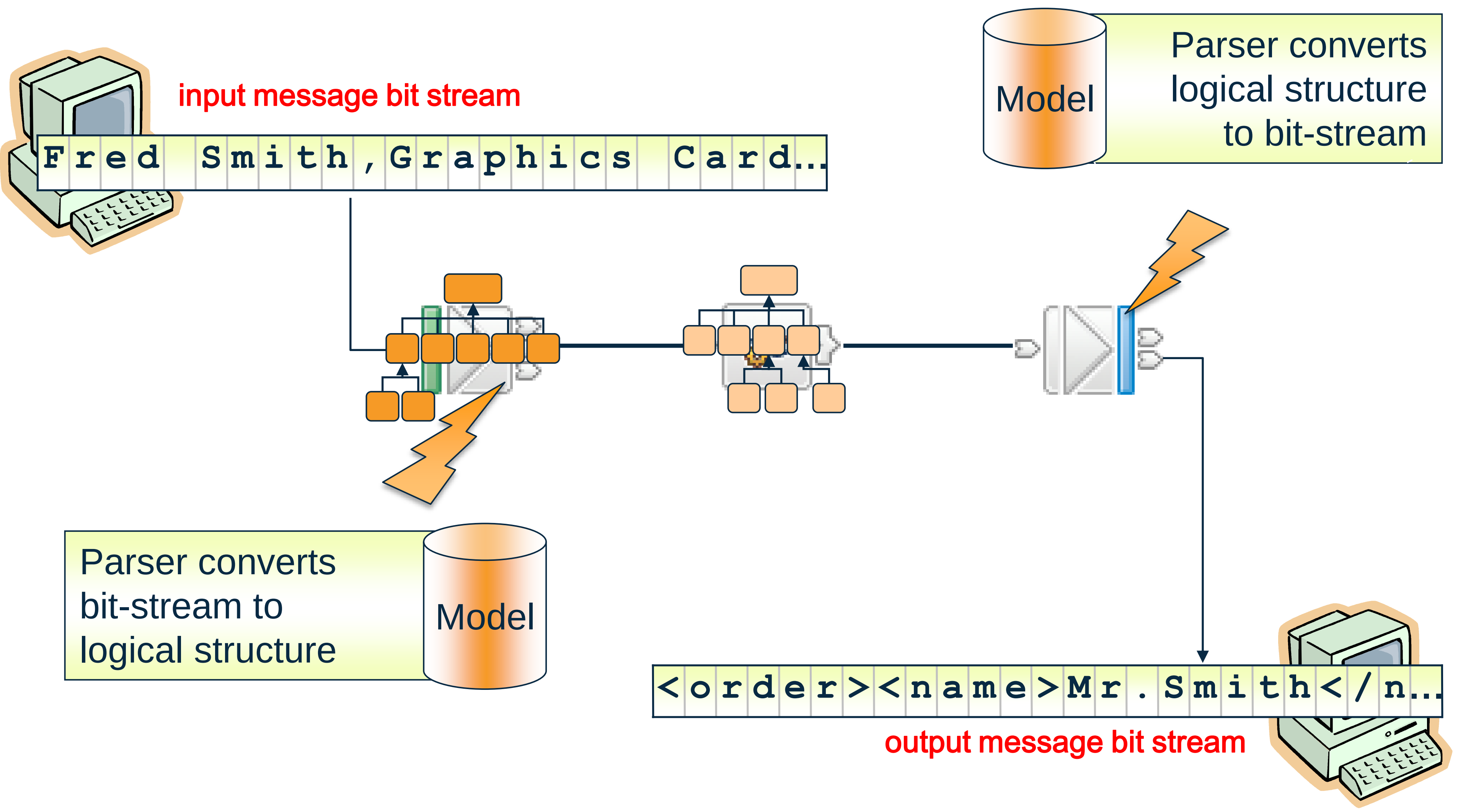
The action represented by a message processing node encapsulates a useful and reusable piece of integration logic. Nodes can be thought of as reusable components in an integration library.

Notes for slide 28 (2 of 2)

A node is joined to its neighbours in the data flow through connectors attached to its data terminals.

- Every node has a fixed number of connection points known as "input" terminals and "output" terminals. These allow it to be connected to its neighbours. Each node normally has one input terminal upon which it receives messages, and multiple output terminals for different processing results within the node. Different types of node have different numbers of terminals.
- A connector joins an output terminal of one node to an input terminal of the next node in the message flow. You can leave an output terminal unconnected, or you can connect a single output terminal to more than one target node.
- After a node has finished processing a message, the connectors defined from the node’s output terminals determine which nodes process the message next. If a node has more than one output terminal connected to a target node, it is the node (not you) that determines the order in which the different execution paths are executed. If a single output terminal has more than one connector to a target node, it is the broker (again, not you) which determines this execution order.
- A node does not always produce an output message for every output terminal: often it produces one output for a specific terminal depending on the message received. E.g. a filter node will typically send a message on either the true or false terminal, but not both.
- When the processing determined by one connector has been completed, the node reissues the message to the next connector, until all possible paths are completed. Updates to a message are never propagated to previously executed nodes, only to following nodes.
- The message flow can only start processing the next message when all paths through the message flow (that is, all connected nodes from all output terminals, as appropriate) have been completed.

Parsers


input message bit stream
Fred Smith,Graphics Card...
Model
Parser converts logical structure to bit-stream
Parser converts bit-stream to logical structure
Model
<order><name>Mr. Smith</n...
output message bit stream

Notes for slide 31

On the previous slide we saw that objects called “message trees” are sent to a node’s input terminals, and either the same or different message tree is propagated from a node’s output terminals.

- The message tree is a logical definition of a message processed by the broker. It’s described as a tree because messages are typically hierarchical in structure; a good example of this is XML. Other message formats too, are also often derived from complex structures which themselves can be derived from complex structures, and so on, which gives them a tree-like shape with leaf nodes representing simple data types.
- In IBM Integration Bus, parsers have the job of converting between physical messages (bit-streams) and logical trees. When a message arrives at the broker through an input node, the message bit-stream is converted into a tree structure by the parser, which typically uses a model to drive the form of the logical tree. Built-in parsers handle well known headers within the message (MQMD, MQRFH2 etc.). Finally the user data is parsed into the tree using the domain parser as identified in the MQRFH2 (or input node). Message Broker’s built-in parsers support multiple domains (MRM, SOAP, XMLNSC, Data Object, XMLNS, JMSMap, JMSStream, MIME, IDOC, BLOB and XML) to enable parsing of user and industry standard formats.
- As the logical tree is passed from node to node, the form of the logical tree may change depending on what the node is doing.
- When the message arrives at an output node, the parser converts the logical tree back into a physical bit-stream where it can be output to the external resource, where it can be read by the target (receiving) application. However, note that an output node need not indicate the end of a flow; it is possible to output to multiple destinations within a single invocation of a message flow. In this case, the logical tree can be passed on to other nodes and manipulated further, even after it has been converted back into a physical bit-stream for this particular output node.

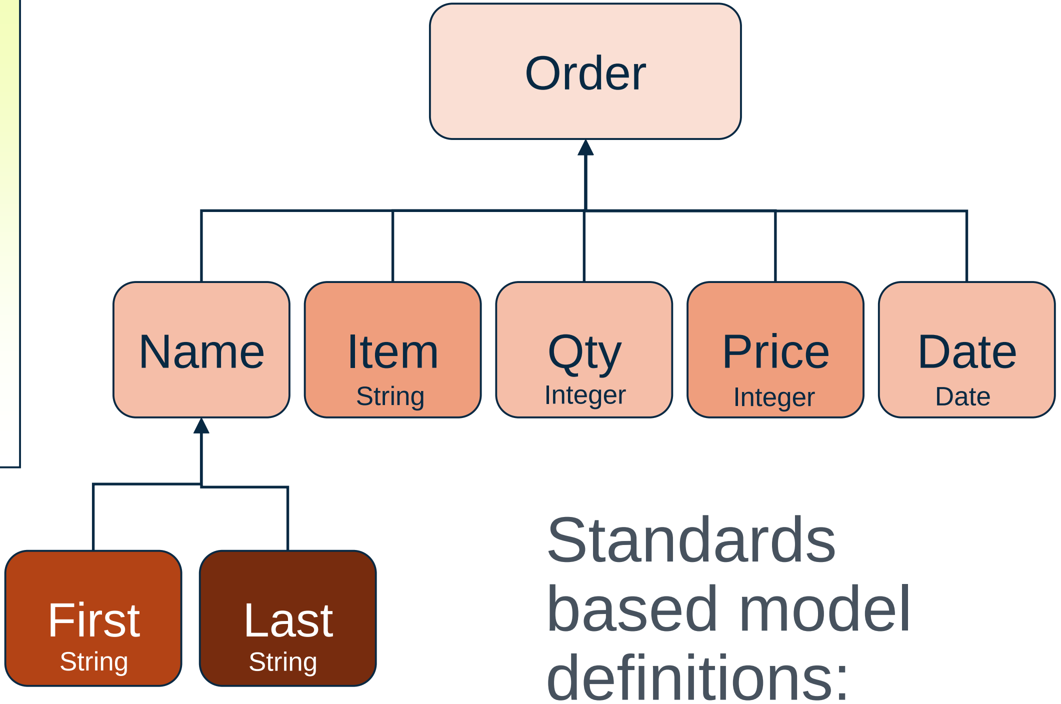
Message Modeling

Physical
Logical

```
<order>
  <name>
    <first>John</first>
    <last>Smith</last>
  </name>
  <item>Graphics Card</item>
  <quantity>32</quantity>
  <price>200</price>
  <date>07/11/09</date>
</order>
```

```
John,Smith,Graphics Card,
32,200,07/11/09
```

```
John Smith.............
Graphics Card.........
322002007110 9.........
```



Standards based model definitions:

- XML
- DFDL

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Notes for slide 33 (1 of 2)

Here is an example of how a physical data structure could be mapped to a logical tree

- Notice how multiple physical formats can correspond to the same logical tree. The first physical format is an XML structure that shows our Order message. The second is a comma separated value (CSV) structure of the same. The third comprises a set of fixed length fields in a custom wire format.
- By manipulating the logical tree inside the Integration Broker rather than the physical bit-stream, the nodes can be completely unaware of the physical format of the data being manipulated. It also makes it easy to introduce new message formats into the broker.

Applications have and require diverse data formats

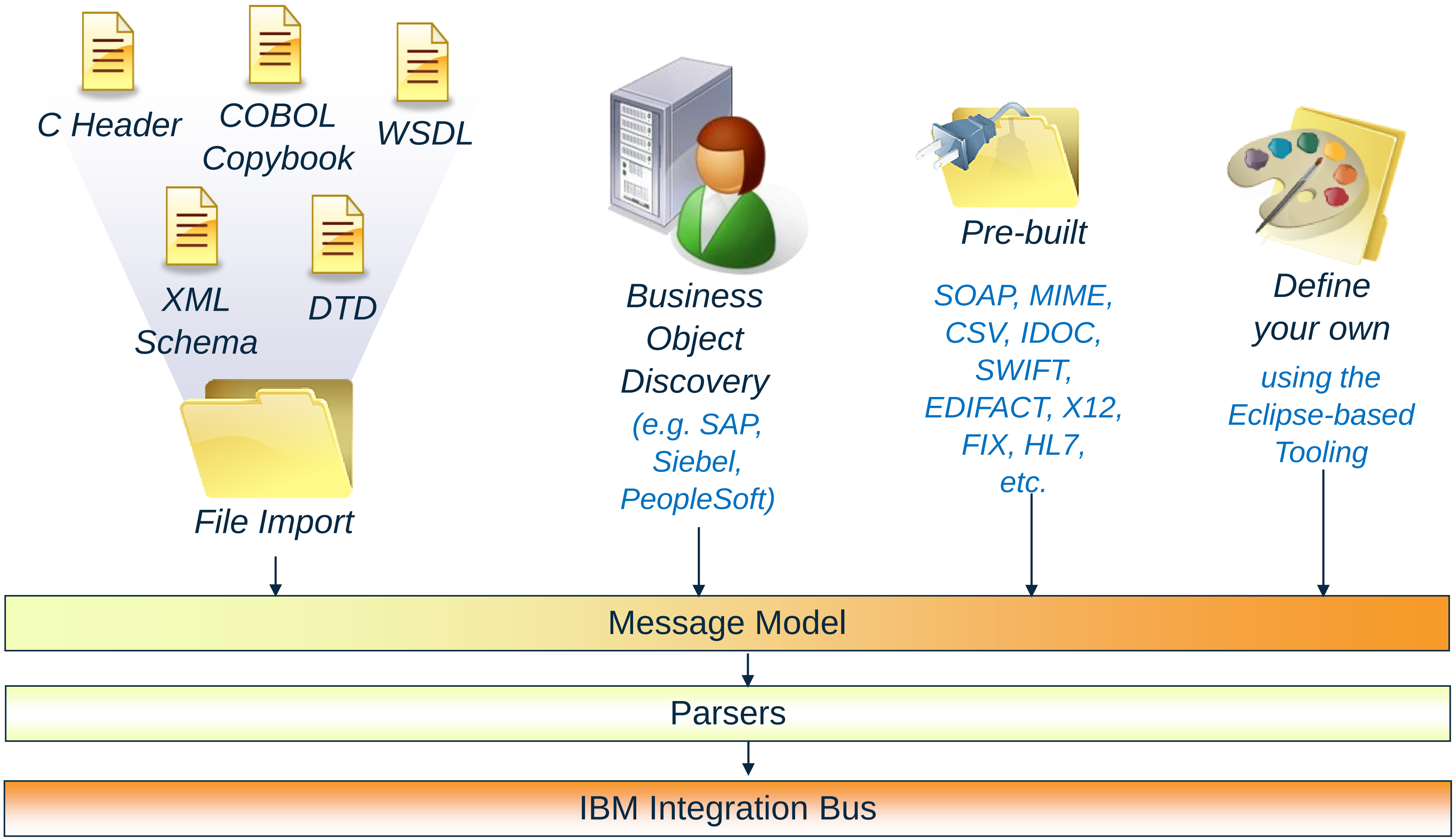
- We all know that XML is the data format that's going to solve every data processing problem that exists! We also know that "XML++", the follow-on compatible meta format that someone in a research laboratory is working on will solve all the problems we don't even know we have today! The fact is that, without wanting to appear cynical, every generation goes through this process. Surely it was the same when COBOL superseded assembler.
- The fact is, that for historic, technical, whimsical, political, geographical, industrial and a whole host of other reasons you probably never even thought of, a hugely diverse range of data formats exist and are used successfully by a myriad of applications every second of every day. It's something that we have to live with and embrace because it isn't going to get any better any time soon.
- The advantage IBM Integration Bus brings by modelling all these messages is that we can rise above the message format detail; so that whether it's a tag delimited SWIFT or EDIFACT message, a custom record format closely mapping a C or COBOL data structure, or good old XML, we can talk about messages in a consistent, format independent way. Integration Bus can manage this diversity.

Notes for slide 33 (2 of 2)

The Logical Message Model

- Reconsider messages and their structure. When we architect messages (no matter what the underlying transport technology), we concern ourselves firstly with the logical structure. For example, a funds transfer message might contain an amount in a particular currency, a transaction date and the relevant account details of the parties involved. These are the important business elements of the message; when discussing the message, we refer to these elements.
- However, when we come to realize the message, we have to choose a specific data format. This may be driven by many factors, but we have to choose one. You may be aware of the advantages of various message formats or have your own personal favourite, or may fancy inventing a new one, but the fact remains that you have to choose a physical *wire format*. So for our transfer message, we might decide to use XML, with its elements, attributes and PCDATA (and a DTD, if we're being really exact), or we might map more closely to a C data structure modelling our message with ints, shorts, chars etc. and worry about *their* various representations(!)
- The Logical message model provided by Integration Bus allows one to describe a message in terms of a tree of elements, each of which has a (possibly user defined) type. At the message tree leaf nodes, the elements have simple types such as strings, integers, decimals, booleans etc. Moreover, elements can have various constraints and qualifiers applied to them that more fully describe them; e.g. elements might be optional, appear in a certain order or only contain certain values.

Creating message models


C Header
COBOL Copybook
WSDL
XML Schema
DTD
File Import
Business Object Discovery
(e.g. SAP, Siebel, PeopleSoft)
Pre-built
SOAP, MIME, CSV, IDOC, SWIFT, EDIFACT, X12, FIX, HL7, etc.
Define your own
using the Eclipse-based Tooling
Message Model
Parsers
IBM Integration Bus

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Notes for slide 36

This slide describes some of the options available for creating message models.

1. If you have messages described by COBOL copybooks, C header files XML DTDs/Schemas or WSDL, use the IBM Integration Bus supplied importers to generate your message model. A wide range of importers exist, so that you can kick start your message modelling.
2. If you wish to use the SAP, Siebel or PeopleSoft nodes inside IBM Integration Bus, you can construct message models directly from the Business Objects on these systems.
3. You can use pre-built models such as those for SWIFT messages.
4. Finally, you can use graphical modelling available in the Message Broker Toolkit to model your messages. You've seen application connections and processing constructed using message flows and nodes; the Message Broker Toolkit provides a similarly visual approach to message modelling.

Powerful Message Transformation Options

Mapping

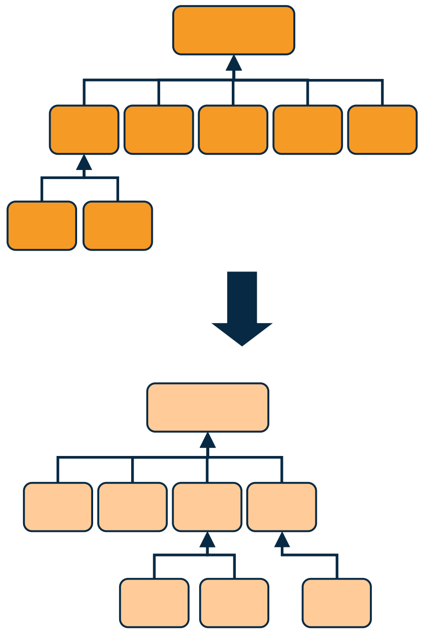
- graphical, easy to use
- drag and drop fields, apply functions

Java Compute

- embed Java programs
- ability to use XPath for tree access

Compute

- describe powerful transformations quickly
- uses SQL-based language (ESQL)



XSL Transform

- convert XML to anything
- uses standard XSL Style sheets

.NETCompute

- use any of the 40+ .NET languages (e.g. C#, VB.NET)
- access COM objects
- Windows only

Notes for slide 38 (1 of 2)

There are several options available to you out-of-the-box for transforming between message formats:

1. Use Graphical Mapping to visually represent messages and transform them. The Integration broker has a mapping node to allows users to visualize and transform messages. The mapping node presents input and output message views; i.e. visualisation of message definitions. Users can “map” elements from the input message to the output message using “drag and drop” operations. More complex operations are possible, such as field concatenation. Graphical mapping is most effective when you have relatively simple transformations to perform and you don’t want to use a programming language (ESQL or Java).
2. XSLT (eXtensible Stylesheet Language Transformations) is a language for describing message transformations. There is a node in IBM Integration Bus that allows you to convert XML messages using style sheets developed in this language.
3. The Compute node uses ESQL (Extended Structured Query Language). This is a language derived from SQL3, and is particularly suited to manipulating both database and message data. You do this with a single syntax (words) which has a common semantic (meaning). ESQL addresses message fields using a natural syntax. For example, the fourth traveller in a travel request message could be:

   ```
   InputRoot.Body.TravelRequestMessage.TravellerDetails[4].LastName
   ```

   ESQL has a rich set of basic data types and operators, as well as the kind of statements and functions you're used to from procedural programming languages, to allow you to perform powerful transformations.

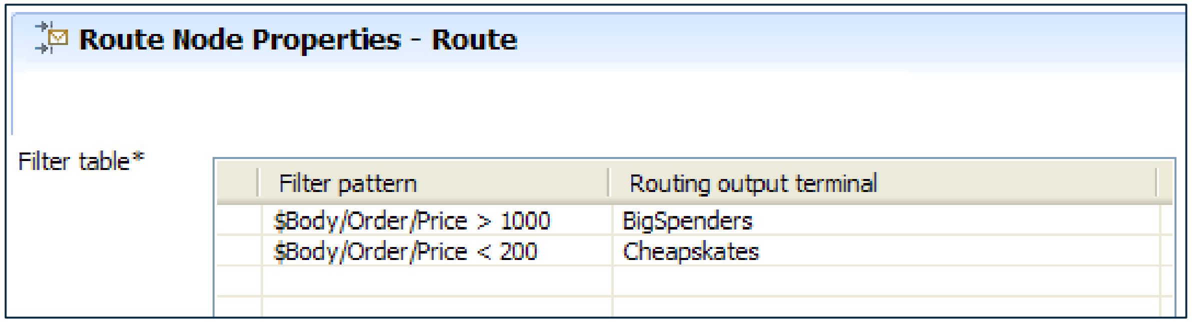
Notes for slide 38 (2 of 2)

Continuing the options available to you for transforming between message formats:

1. You can also use the power of Java to route and transform your messages. The JavaCompute node is fully integrated into the Eclipse Development Environment to providing usability aids such as content assist and incremental compilation. You can refer to elements using an expressive XPATH syntax, so that message navigation and element creation and modification are vastly simplified, and comparable in simplicity to ESQL field references. JDBC type 4 support allows you to perform database and message tree operations in the Java Compute node. On z/OS Java workload is eligible for offload onto the zSeries Application Assist Processors zAAP.
2. .NET is available as an option on Windows platforms, and allows you to transform data and invoke general purpose integration logic on any of the .NET supported platforms. Templates for Visual Studio and an integrated Visual Studio debugger make this a natural choice for Windows programmers.
   – for example, you can use this node to integrate your Excel spreadsheets!
3. You can mix and match your transformation styles, or use just one throughout your enterprise. It will probably be your skill set which determines whether you chose to use Graphical Mapping, Java, XSLT or ESQL to perform your message routing and transformation.

Easily Address Message Elements

```
public class jcn extends MbJavaComputeNode {
  public void evaluate(MbMessageAssembly assembly) throws MbException {
   ...
   String lastName =
       (String)assembly.getMessage().evaluateXPath("/Body/Order/Name/Last");
   ...
  }
}
```



Route Node Properties - Route

Filter table*

Filter pattern	Routing output terminal
$Body/Order/Price > 1000	BigSpenders
$Body/Order/Price < 200	Cheapskates

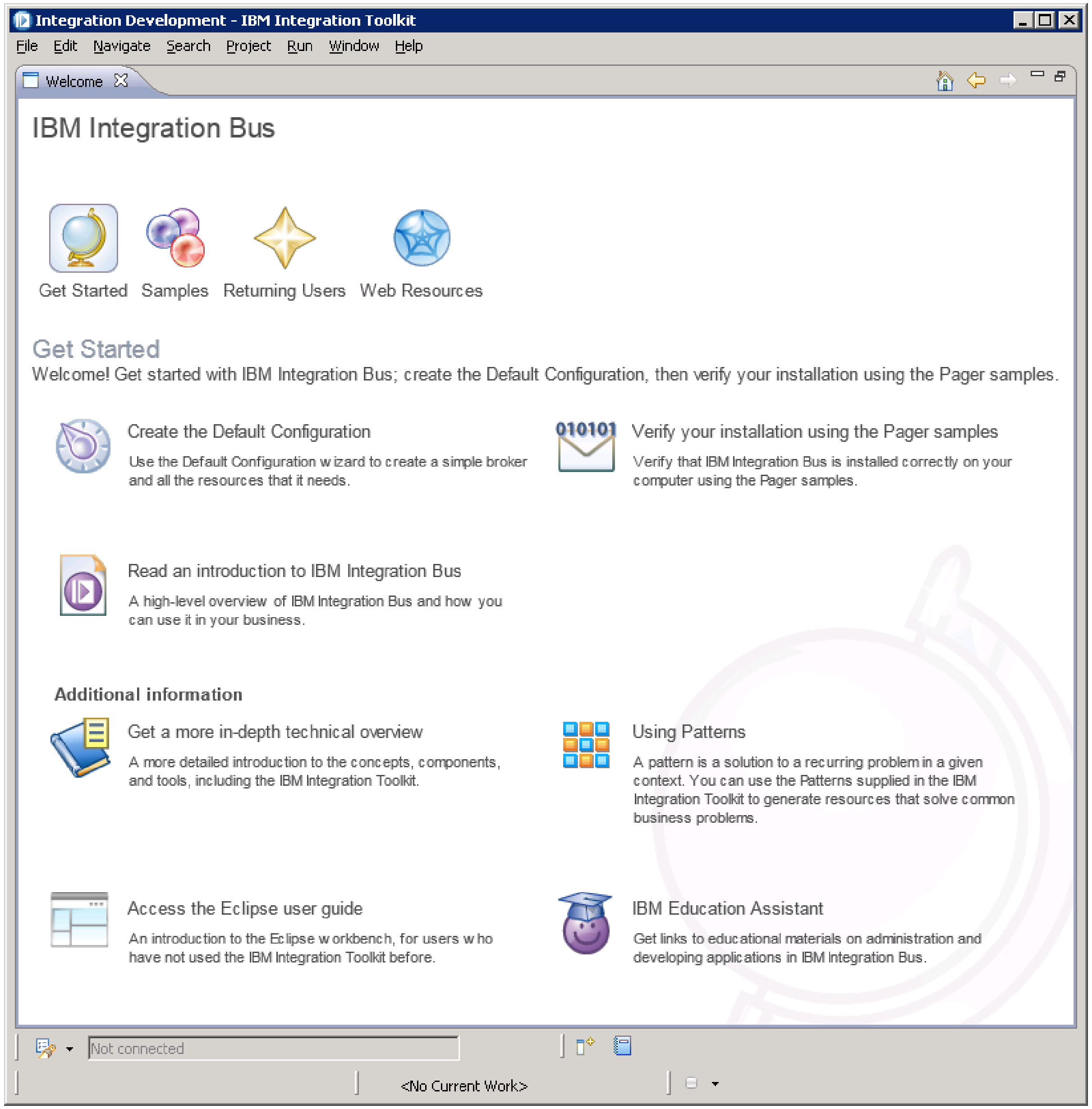
```
IF Body.Order.Date < '2008/01/01' THEN
   INSERT INTO Database.OldOrders (LastName,Item,Quantity)
   VALUES (Body.Order.Name.Last,
           Body.Order.Item,
           Body.Order.Quantity);
ENDIF;
```

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Notes for slide 41

- Each node’s configuration (which includes some Java logic in the case of the Java Compute node) dictates what you want the node to do, and this may include manipulation of one or more elements in the message tree.
- Here are some examples of node configurations that address elements in the logical tree.
- In most cases, elements can be addressed using either XPath (as shown in the JavaCompute and Route) or ESQL (as shown in the DataInsert node).

Development: the Integration Toolkit


Integration Development - IBM Integration Toolkit
File Edit Navigate Search Project Run Window Help
Welcome
Get Started
Samples
Returning Users
Web Resources
Not connected
<No Current Work>

IBM Integration Bus

Get Started

Welcome! Get started with IBM Integration Bus; create the Default Configuration, then verify your installation using the Pager samples.

Create the Default Configuration

Use the Default Configuration wizard to create a simple broker and all the resources that it needs.

Verify your installation using the Pager samples

Verify that IBM Integration Bus is installed correctly on your computer using the Pager samples.

Read an introduction to IBM Integration Bus

A high-level overview of IBM Integration Bus and how you can use it in your business.

Additional information

Get a more in-depth technical overview

A more detailed introduction to the concepts, components, and tools, including the IBM Integration Toolkit.

Using Patterns

A pattern is a solution to a recurring problem in a given context. You can use the Patterns supplied in the IBM Integration Toolkit to generate resources that solve common business problems.

Access the Eclipse user guide

An introduction to the Eclipse workbench, for users who have not used the IBM Integration Toolkit before.

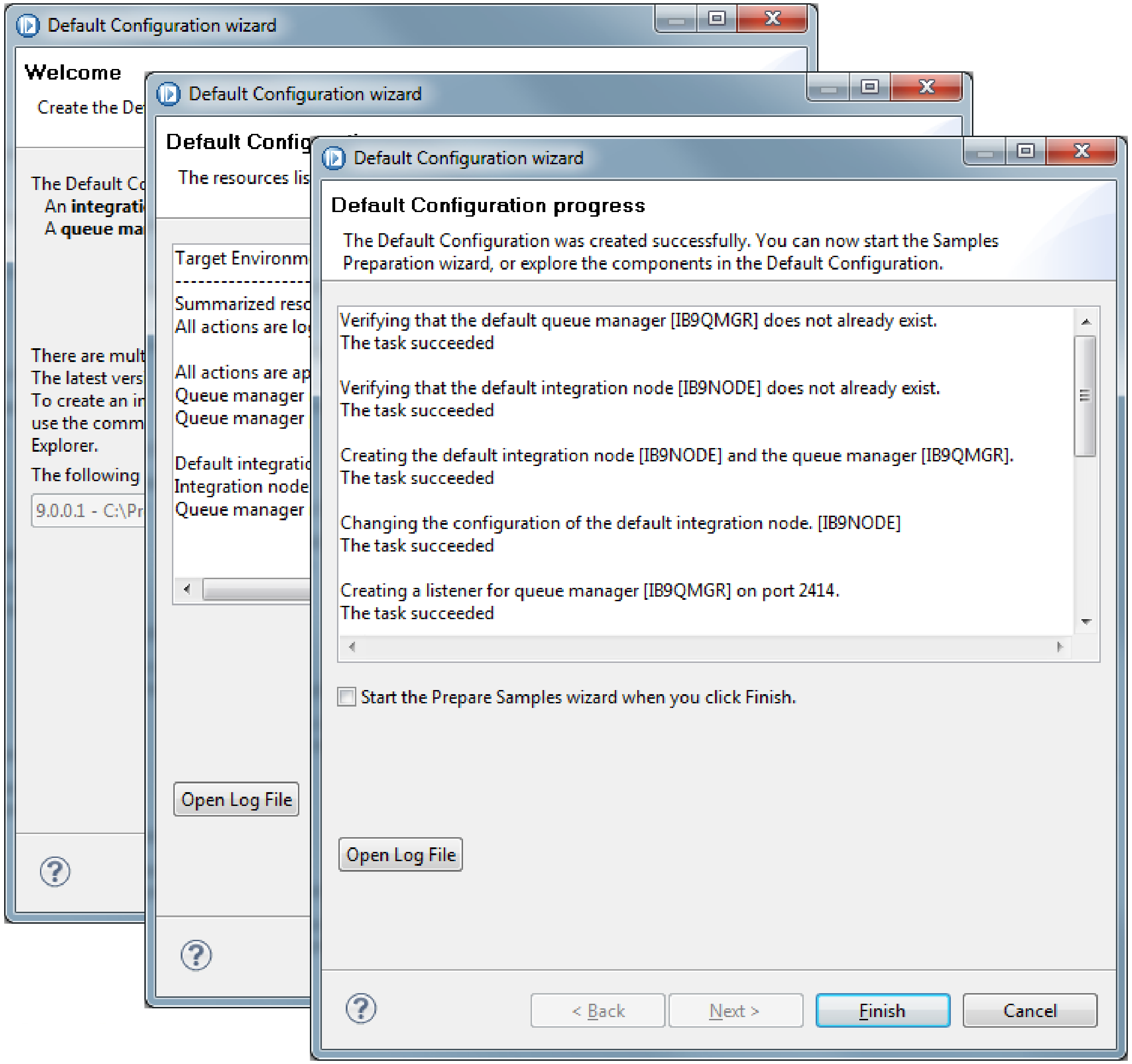
IBM Education Assistant

Get links to educational materials on administration and developing applications in IBM Integration Bus.

- Eclipse based tooling for developing integrations
- easy “getting started” accelerators

SHARE in Orlando 2015

Default Configuration Wizard


Default Configuration wizard

Welcome

Create the De

The Default C
An integrati
A queue ma

There are mult
The latest vers
To create an i
use the comm
Explorer.

The following

9.0.0.1 - C:\Pr

Default Configuration wizard

Default Config

The resources lis

Target Environm

Summarized reso
All actions are lo

All actions are ap
Queue manager
Queue manager

Default integrati
Integration node
Queue manager

Default Configuration wizard

Default Configuration progress

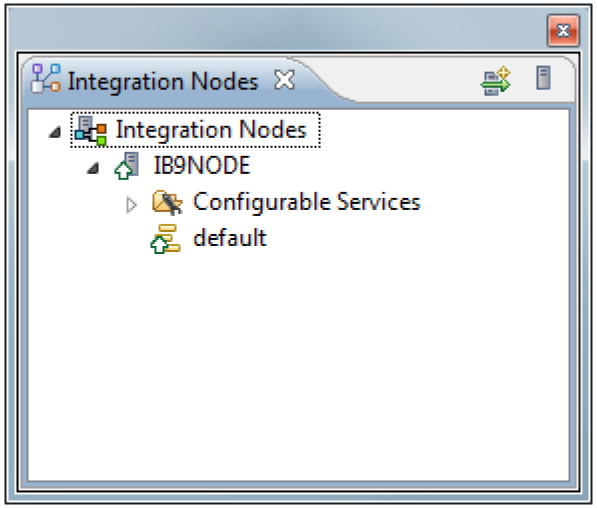
The Default Configuration was created successfully. You can now start the Samples Preparation wizard, or explore the components in the Default Configuration.

```
Verifying that the default queue manager [IB9QMGR] does not already exist.
The task succeeded

Verifying that the default integration node [IB9NODE] does not already exist.
The task succeeded

Creating the default integration node [IB9NODE] and the queue manager [IB9QMGR].
The task succeeded

Changing the configuration of the default integration node. [IB9NODE]
The task succeeded

Creating a listener for queue manager [IB9QMGR] on port 2414.
The task succeeded
```

☐ Start the Prepare Samples wizard when you click Finish.

Use the DCW to easily create:

- Queue Manager
- Integration Node
- Integration Server



Quick Starts

Start building your application with one of the following tasks.

Start by creating an application

An **Application** is a container for all the resources that are required to create a solution. More...

Start by creating an integration service

An **Integration Service** is an application with a well-defined interface and structure. More...

Start by creating a library

A **Library** is a logical grouping of related code, data, or both. More...

Start from WSDL and/or XSD files

Use this task to create an **Integration Service**, **Application** or **Library** which includes your WSDL and/or XSD files.

Start by discovering a service

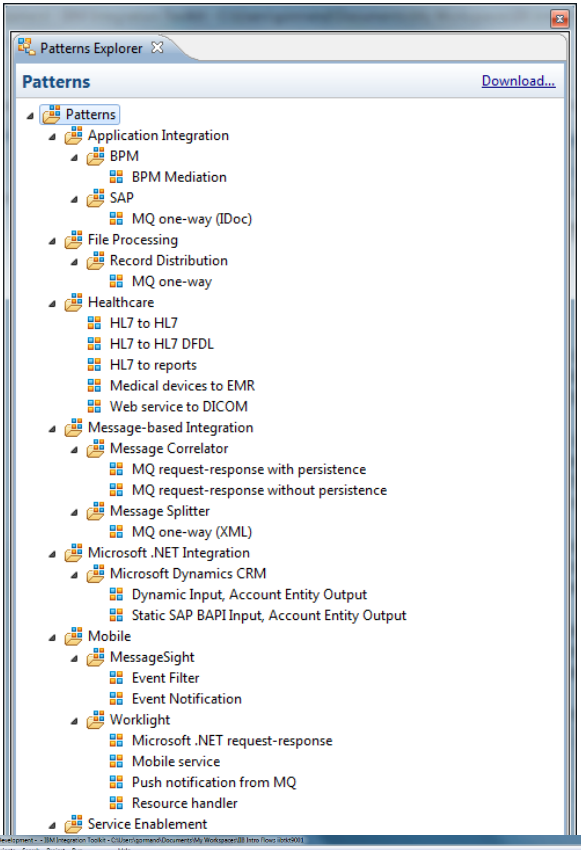
A **Service** allows you to invoke the remote system using discovered resources.

Start from patterns

A **Pattern** is a reusable solution that encapsulates a tested approach to solving a common architecture, design, or deployment task. More...

Start from samples

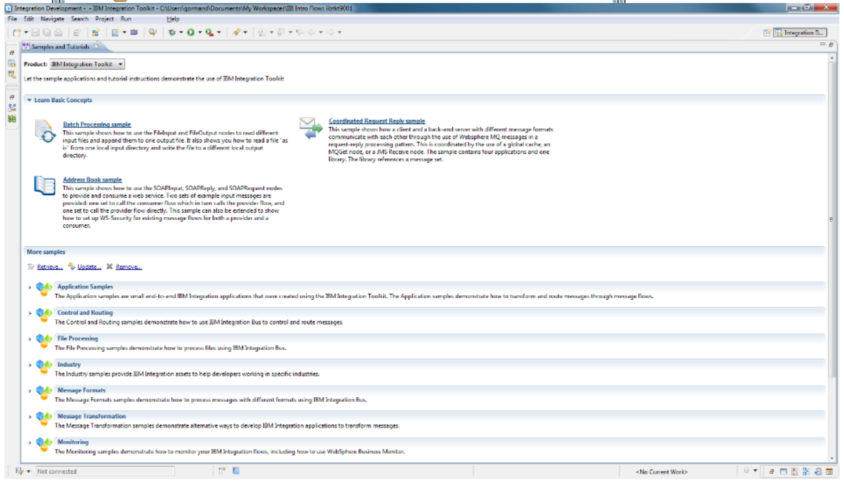
Use the **Samples** to learn more about the features in IBM Integration Bus. More...



Patterns Explorer

Patterns Download...

- Patterns
  - Application Integration
    - BPM
      - BPM Mediation
    - SAP
      - MQ one-way (IDoc)
  - File Processing
    - Record Distribution
      - MQ one-way
  - Healthcare
    - HL7 to HL7
    - HL7 to HL7 DFDL
    - HL7 to reports
    - Medical devices to EMR
    - Web service to DICOM
  - Message-based Integration
    - Message Correlator
      - MQ request-response with persistence
      - MQ request-response without persistence
    - Message Splitter
      - MQ one-way (XML)
  - Microsoft .NET Integration
    - Microsoft Dynamics CRM
      - Dynamic Input, Account Entity Output
      - Static SAP BAPI Input, Account Entity Output
  - Mobile
    - MessageSight
      - Event Filter
      - Event Notification
    - Worklight
      - Microsoft .NET request-response
      - Mobile service
      - Push notification from MQ
      - Resource handler
  - Service Enablement



Integration Development - - IBM Integration Toolkit - C:\Users\gomand\Documents\My Workspaces\IIB Intro Flows iibtkt9001

Product: IBM Integration Toolkit

Let the sample applications and tutorial instructions demonstrate the use of IBM Integration Toolkit

Learn Basic Concepts

Batch Processing sample
This sample shows how to use the FileInput and FileOutput nodes to read different input files and append them to one output file. It also shows you how to read a file "as is" from one local input directory and write the file to a different local output directory.

Coordinated Request Reply sample
This sample shows how a client and a back-end server with different message formats communicate with each other through the use of Websphere MQ messages in a request-reply processing pattern. This is coordinated by the use of a global cache, an MQGet node, or a JMS Receive node. The sample contains four applications and one library. The library references a message set.

Address Book sample
This sample shows how to use the SOAPInput, SOAPReply, and SOAPRequest nodes to provide and consume a web service. Two sets of example input messages are provided: one set to call the consumer flow which in turn calls the provider flow, and one set to call the provider flow directly. This sample can also be extended to show how to set up WS-Security for existing message flows for both a provider and a consumer.

More samples

Retrieve... Update... Remove...

Application Samples
The Application samples are small end-to-end IBM Integration applications that were created using the IBM Integration Toolkit. The Application samples demonstrate how to transform and route messages through message flows.

Control and Routing
The Control and Routing samples demonstrate how to use IBM Integration Bus to control and route messages.

File Processing
The File Processing samples demonstrate how to process files using IBM Integration Bus.

Industry
The Industry samples provide IBM Integration assets to help developers working in specific industries.

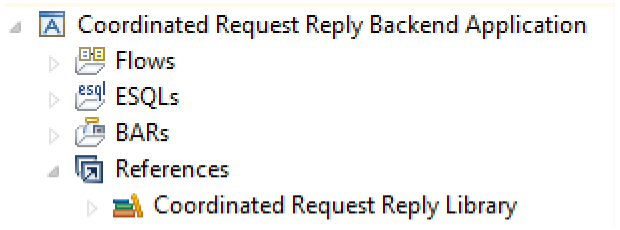
Message Formats
The Message Formats samples demonstrate how to process messages with different formats using IBM Integration Bus.

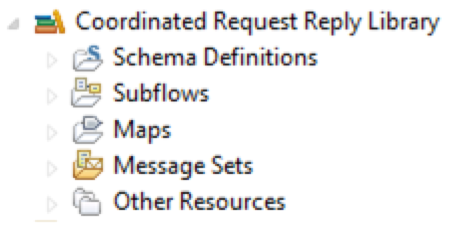
Message Transformation
The Message Transformation samples demonstrate alternative ways to develop IBM Integration applications to transform messages.

Monitoring
The Monitoring samples demonstrate how to monitor your IBM Integration flows, including how to use WebSphere Business Monitor.

Logically separate your applications

- You can separate your applications during development and runtime
- Application
  - means of encapsulating resources to solve a specific connectivity problem
  - application can reference one or more libraries
- Library
  - a logical grouping of related routines and/or data
  - libraries help with reuse and ease of resource management
  - library can reference one or more libraries


Coordinated Request Reply Backend Application
- Flows
- ESQLs
- BARs
- References
  - Coordinated Request Reply Library

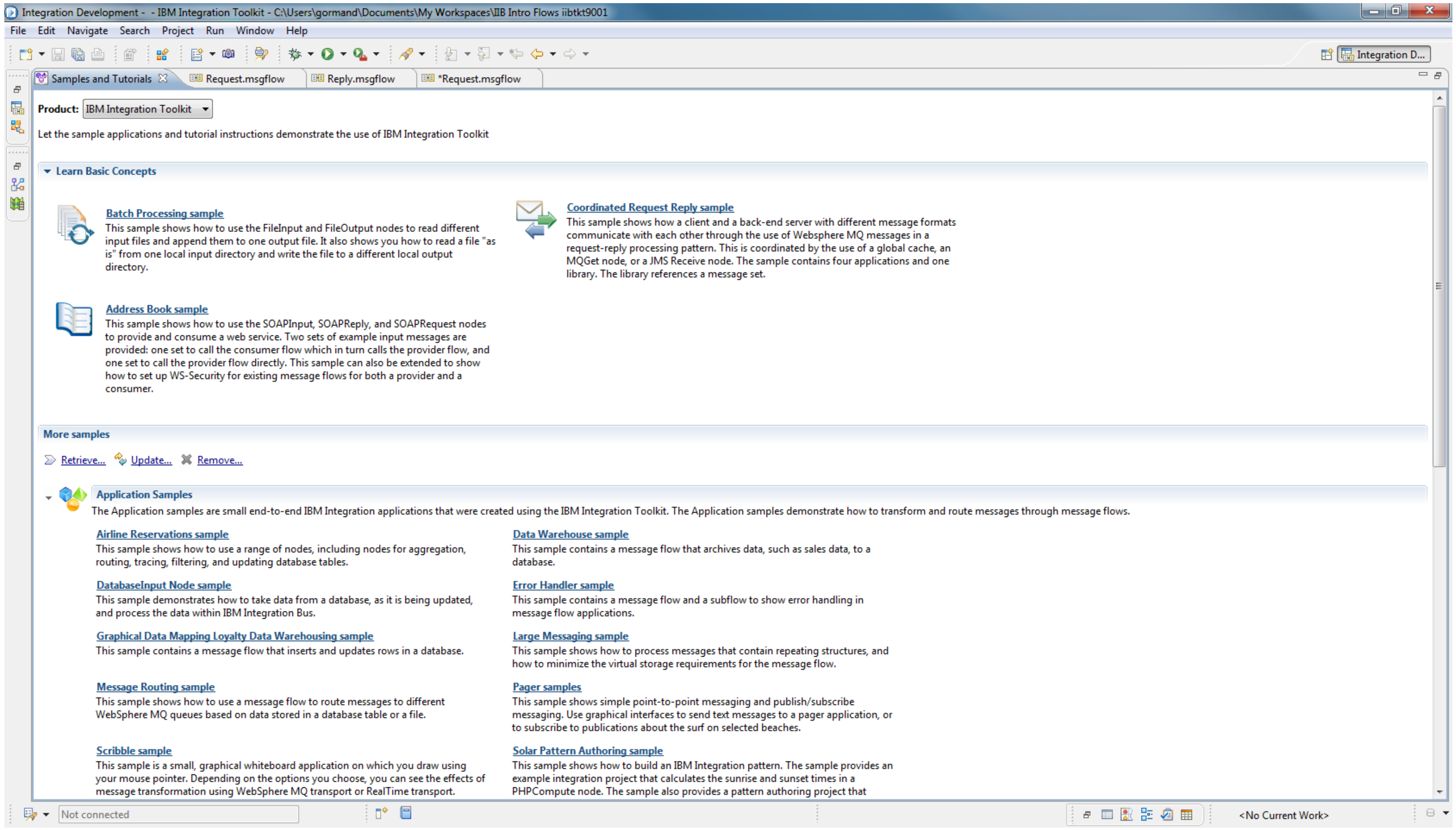

Coordinated Request Reply Library
- Schema Definitions
- Subflows
- Maps
- Message Sets
- Other Resources

Notes for slide 46

Features of Applications and Libraries

- Applications promote encapsulation and isolation
  - typically contain “main” message flows and dependent resources
  - ESQL, Java, maps, message models, subflows, adapter files, etc.
  - dependent resources could live in referenced libraries
  - multiple applications can be packaged into a single BAR
  - multiple applications can be deployed to an execution group
  - visibility of resource restricted to containing application
  - referenced libraries are deployed inside application container (by copy)
- Libraries facilitate re-use and simplify resource management
  - typically contain reusable helper routines and resources
  - subflows, ESQL, Java, maps, message models, adapter files, etc.
  - use multiple libraries to group related resources (e.g. by type or function)
  - multiple applications can reference the same library
  - each application gets its own copy of the library during package/deploy
  - libraries are packaged as part of referencing application in the BAR
  - a library can reference other libraries

Samples


Integration Development - - IBM Integration Toolkit - C:\Users\gormand\Documents\My Workspaces\IIB Intro Flows iibtkt9001
File Edit Navigate Search Project Run Window Help
Integration D...
Samples and Tutorials
Request.msgflow
Reply.msgflow
*Request.msgflow
Not connected
<No Current Work>

Product: IBM Integration Toolkit

Let the sample applications and tutorial instructions demonstrate the use of IBM Integration Toolkit

Learn Basic Concepts

Batch Processing sample
This sample shows how to use the FileInput and FileOutput nodes to read different input files and append them to one output file. It also shows you how to read a file "as is" from one local input directory and write the file to a different local output directory.

Coordinated Request Reply sample
This sample shows how a client and a back-end server with different message formats communicate with each other through the use of Websphere MQ messages in a request-reply processing pattern. This is coordinated by the use of a global cache, an MQGet node, or a JMS Receive node. The sample contains four applications and one library. The library references a message set.

Address Book sample
This sample shows how to use the SOAPInput, SOAPReply, and SOAPRequest nodes to provide and consume a web service. Two sets of example input messages are provided: one set to call the consumer flow which in turn calls the provider flow, and one set to call the provider flow directly. This sample can also be extended to show how to set up WS-Security for existing message flows for both a provider and a consumer.

More samples

Retrieve... Update... Remove...

Application Samples
The Application samples are small end-to-end IBM Integration applications that were created using the IBM Integration Toolkit. The Application samples demonstrate how to transform and route messages through message flows.

Airline Reservations sample
This sample shows how to use a range of nodes, including nodes for aggregation, routing, tracing, filtering, and updating database tables.

Data Warehouse sample
This sample contains a message flow that archives data, such as sales data, to a database.

DatabaseInput Node sample
This sample demonstrates how to take data from a database, as it is being updated, and process the data within IBM Integration Bus.

Error Handler sample
This sample contains a message flow and a subflow to show error handling in message flow applications.

Graphical Data Mapping Loyalty Data Warehousing sample
This sample contains a message flow that inserts and updates rows in a database.

Large Messaging sample
This sample shows how to process messages that contain repeating structures, and how to minimize the virtual storage requirements for the message flow.

Message Routing sample
This sample shows how to use a message flow to route messages to different WebSphere MQ queues based on data stored in a database table or a file.

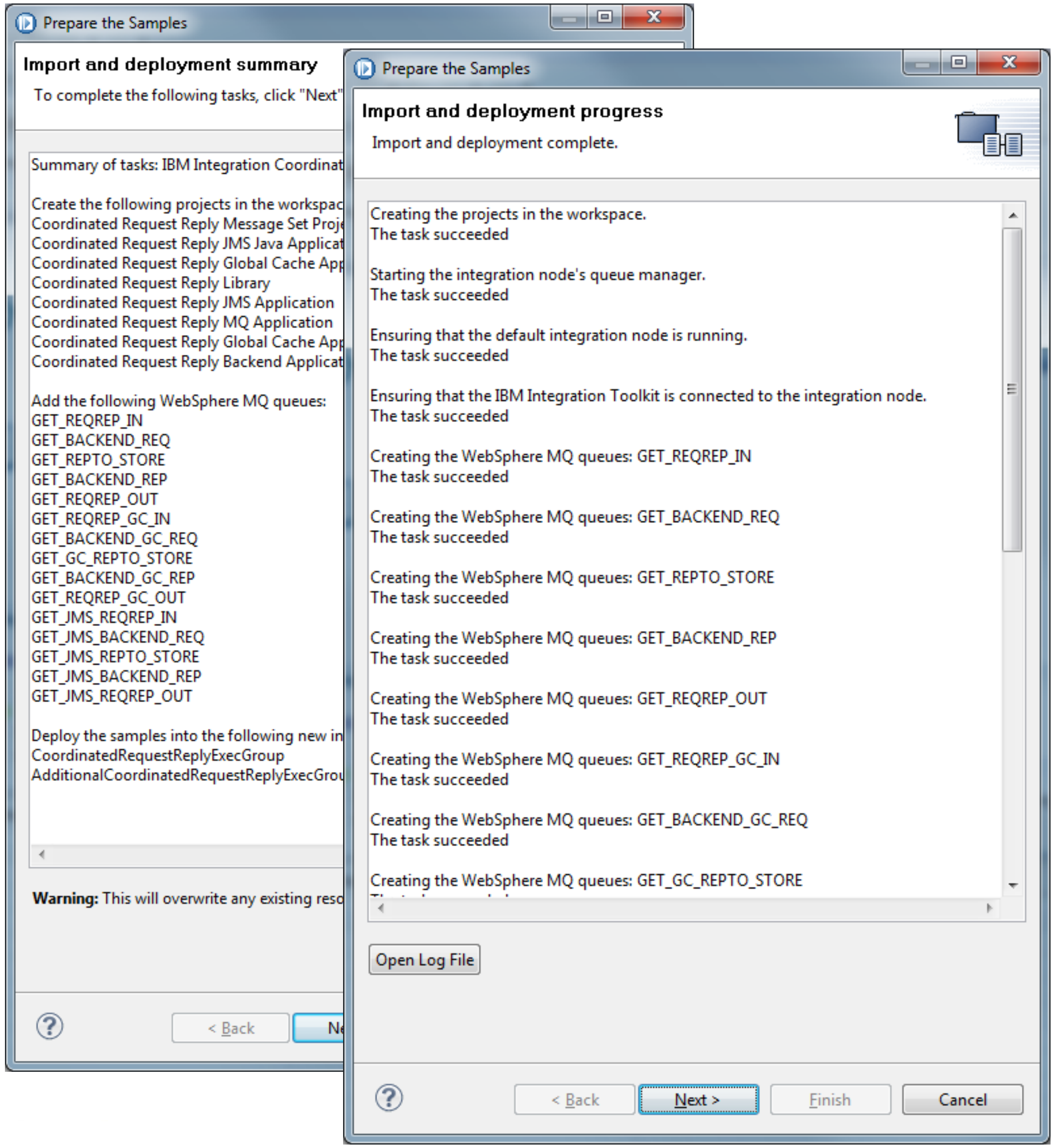
Pager samples
This sample shows simple point-to-point messaging and publish/subscribe messaging. Use graphical interfaces to send text messages to a pager application, or to subscribe to publications about the surf on selected beaches.

Scribble sample
This sample is a small, graphical whiteboard application on which you draw using your mouse pointer. Depending on the options you choose, you can see the effects of message transformation using WebSphere MQ transport or RealTime transport.

Solar Pattern Authoring sample
This sample shows how to build an IBM Integration pattern. The sample provides an example integration project that calculates the sunrise and sunset times in a PHPCompute node. The sample also provides a pattern authoring project that

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Import and Deploy a product sample



Prepare the Samples

Import and deployment summary

To complete the following tasks, click "Next"

Summary of tasks: IBM Integration Coordinat

Create the following projects in the workspac
Coordinated Request Reply Message Set Proj
Coordinated Request Reply JMS Java Applicat
Coordinated Request Reply Global Cache App
Coordinated Request Reply Library
Coordinated Request Reply JMS Application
Coordinated Request Reply MQ Application
Coordinated Request Reply Global Cache App
Coordinated Request Reply Backend Applicat

Add the following WebSphere MQ queues:
GET_REQREP_IN
GET_BACKEND_REQ
GET_REPTO_STORE
GET_BACKEND_REP
GET_REQREP_OUT
GET_REQREP_GC_IN
GET_BACKEND_GC_REQ
GET_GC_REPTO_STORE
GET_BACKEND_GC_REP
GET_REQREP_GC_OUT
GET_JMS_REQREP_IN
GET_JMS_BACKEND_REQ
GET_JMS_REPTO_STORE
GET_JMS_BACKEND_REP
GET_JMS_REQREP_OUT

Deploy the samples into the following new in
CoordinatedRequestReplyExecGroup
AdditionalCoordinatedRequestReplyExecGrou

Warning: This will overwrite any existing reso

Prepare the Samples

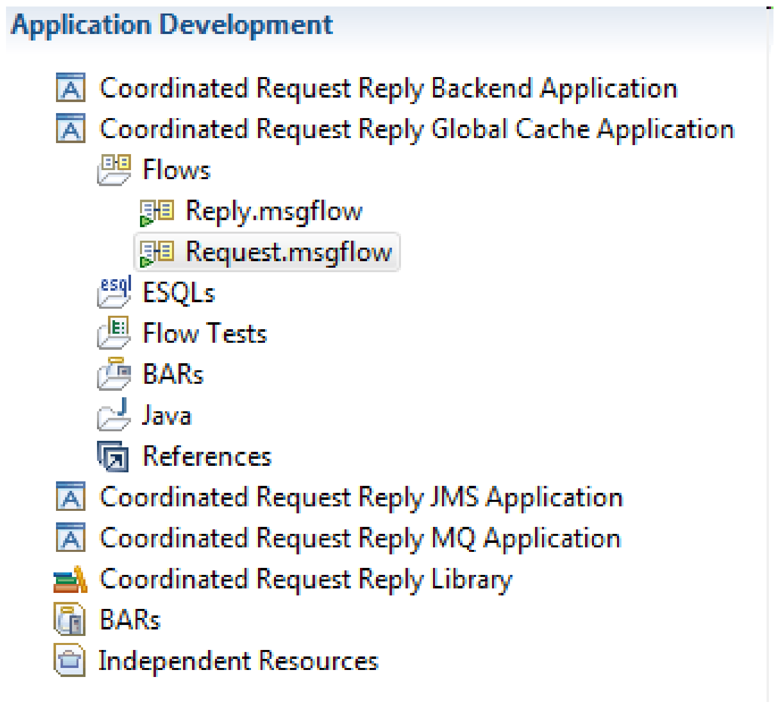
Import and deployment progress

Import and deployment complete.

```
Creating the projects in the workspace.
The task succeeded

Starting the integration node's queue manager.
The task succeeded

Ensuring that the default integration node is running.
The task succeeded

Ensuring that the IBM Integration Toolkit is connected to the integration node.
The task succeeded

Creating the WebSphere MQ queues: GET_REQREP_IN
The task succeeded

Creating the WebSphere MQ queues: GET_BACKEND_REQ
The task succeeded

Creating the WebSphere MQ queues: GET_REPTO_STORE
The task succeeded

Creating the WebSphere MQ queues: GET_BACKEND_REP
The task succeeded

Creating the WebSphere MQ queues: GET_REQREP_OUT
The task succeeded

Creating the WebSphere MQ queues: GET_REQREP_GC_IN
The task succeeded

Creating the WebSphere MQ queues: GET_BACKEND_GC_REQ
The task succeeded

Creating the WebSphere MQ queues: GET_GC_REPTO_STORE
```

- There are many product samples that show how to use IIB
- These are easily imported into the development toolkit workspace, and deployed to the integration server runtime

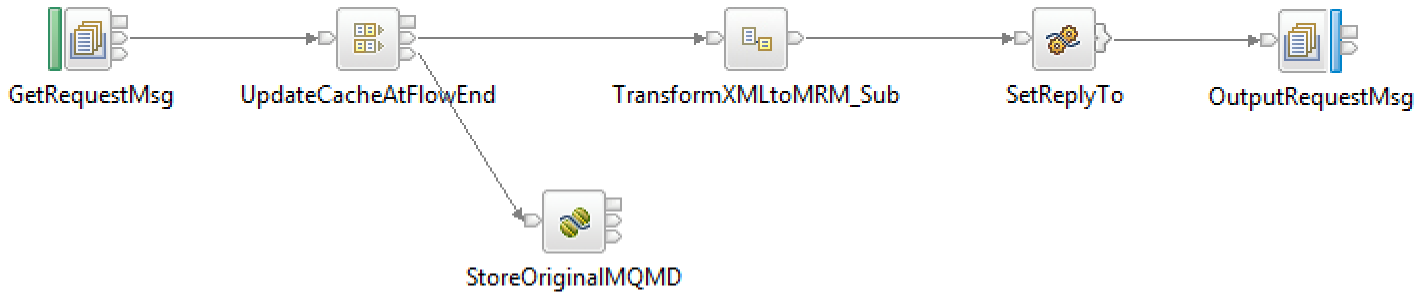
Sample artefacts



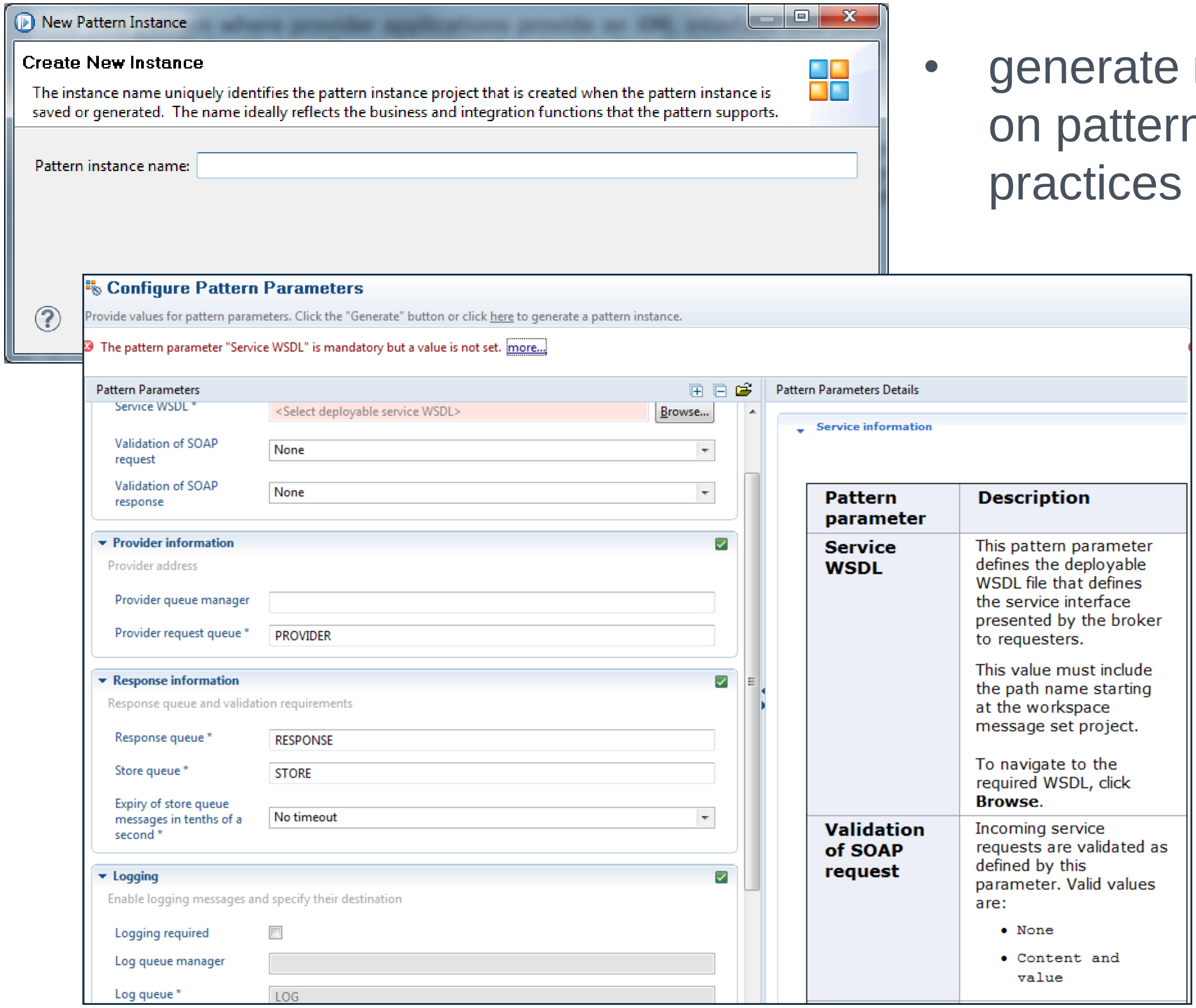
Application Development

- Coordinated Request Reply Backend Application
- Coordinated Request Reply Global Cache Application
  - Flows
    - Reply.msgflow
    - Request.msgflow
  - ESQLs
  - Flow Tests
  - BARs
  - Java
  - References
- Coordinated Request Reply JMS Application
- Coordinated Request Reply MQ Application
- Coordinated Request Reply Library
- BARs
- Independent Resources

- This sample has imported several integration data flows to demonstrate a coordinated request/reply scenario
- The message flows are grouped in applications and libraries as can be seen to the left
- Below is one of the message flows for this sample. It is a complete transaction



Generate Pattern Instances



New Pattern Instance

Create New Instance

The instance name uniquely identifies the pattern instance project that is created when the pattern instance is saved or generated. The name ideally reflects the business and integration functions that the pattern supports.

Pattern instance name:

Configure Pattern Parameters

Provide values for pattern parameters. Click the "Generate" button or click here to generate a pattern instance.

The pattern parameter "Service WSDL" is mandatory but a value is not set. more...

Pattern Parameters

Service WSDL *	<Select deployable service WSDL>
Validation of SOAP request	None
Validation of SOAP response	None

Provider information

Provider address

Provider queue manager	
Provider request queue *	PROVIDER

Response information

Response queue and validation requirements

Response queue *	RESPONSE
Store queue *	STORE
Expiry of store queue messages in tenths of a second *	No timeout

Logging

Enable logging messages and specify their destination

Logging required	☐
Log queue manager	
Log queue *	LOG

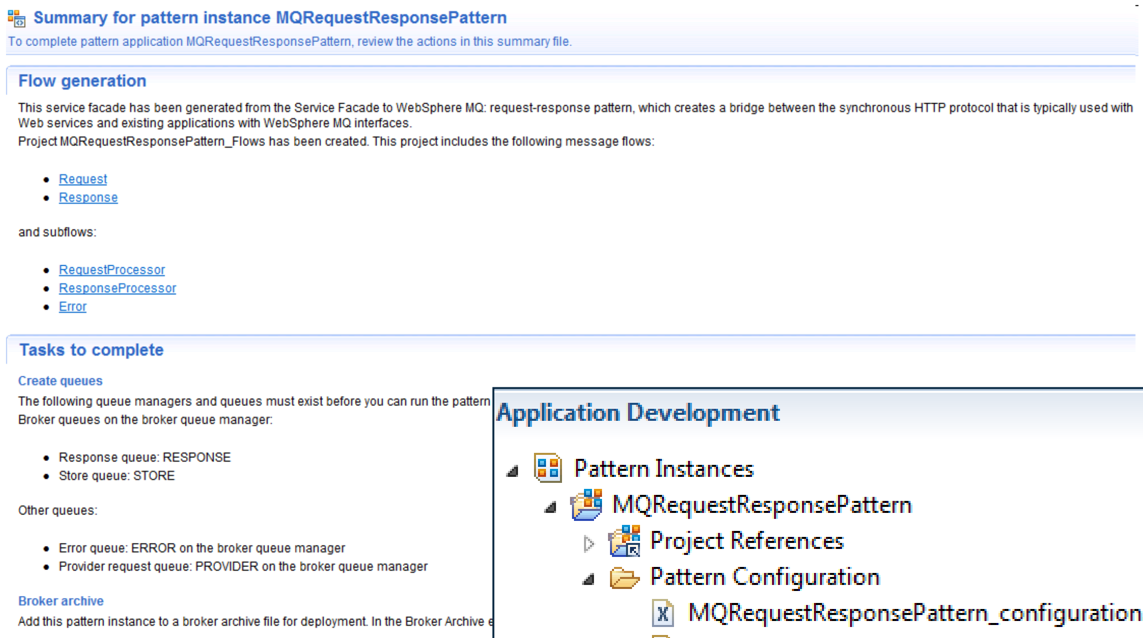
Pattern Parameters Details

Service information

Pattern parameter	Description
Service WSDL	This pattern parameter defines the deployable WSDL file that defines the service interface presented by the broker to requesters. This value must include the path name starting at the workspace message set project. To navigate to the required WSDL, click **Browse**.
Validation of SOAP request	Incoming service requests are validated as defined by this parameter. Valid values are: • `None` • `Content and value`

- generate message flows based on patterns which define best practices
  - properties allow the pattern instance to be customised
  - user defined patterns can be authored and distributed to other developers to enforce best practices

Pattern Artefacts



Summary for pattern instance MQRequestResponsePattern

To complete pattern application MQRequestResponsePattern, review the actions in this summary file.

Flow generation

This service facade has been generated from the Service Facade to WebSphere MQ: request-response pattern, which creates a bridge between the synchronous HTTP protocol that is typically used with Web services and existing applications with WebSphere MQ interfaces.

Project MQRequestResponsePattern_Flows has been created. This project includes the following message flows:

- Request
- Response

and subflows:

- RequestProcessor
- ResponseProcessor
- Error

Tasks to complete

Create queues

The following queue managers and queues must exist before you can run the pattern

Broker queues on the broker queue manager:

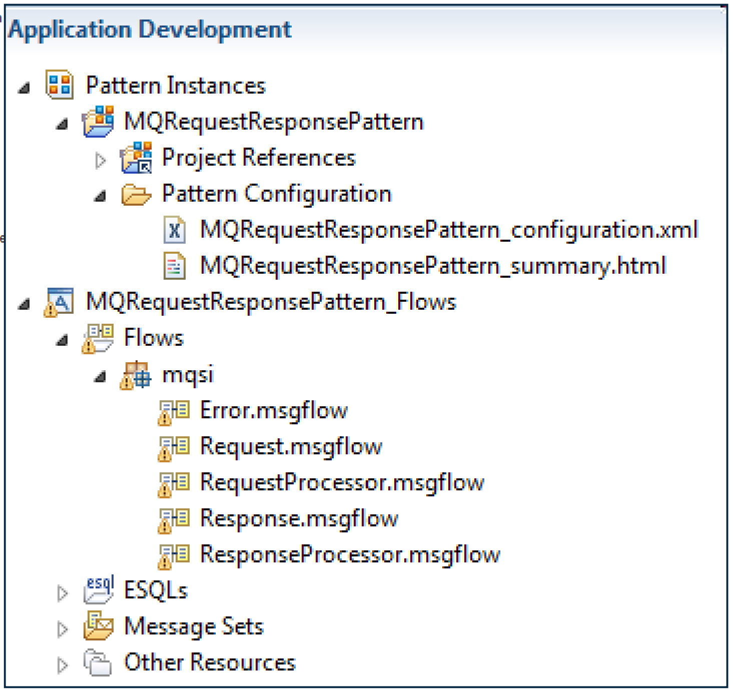
- Response queue: RESPONSE
- Store queue: STORE

Other queues:

- Error queue: ERROR on the broker queue manager
- Provider request queue: PROVIDER on the broker queue manager

Broker archive

Add this pattern instance to a broker archive file for deployment. In the Broker Archive e



Application Development

```
◢ Pattern Instances
    ◢ MQRequestResponsePattern
        ▷ Project References
        ◢ Pattern Configuration
              MQRequestResponsePattern_configuration.xml
              MQRequestResponsePattern_summary.html
◢ MQRequestResponsePattern_Flows
    ◢ Flows
        ◢ mqsi
              Error.msgflow
              Request.msgflow
              RequestProcessor.msgflow
              Response.msgflow
              ResponseProcessor.msgflow
    ▷ ESQLs
    ▷ Message Sets
    ▷ Other Resources
```

- message flows are generated in the pattern instance
- they are ready to be deployed to the integration server runtime
- any tasks required to run the message flows are listed, such as creating MQ Queues

Services

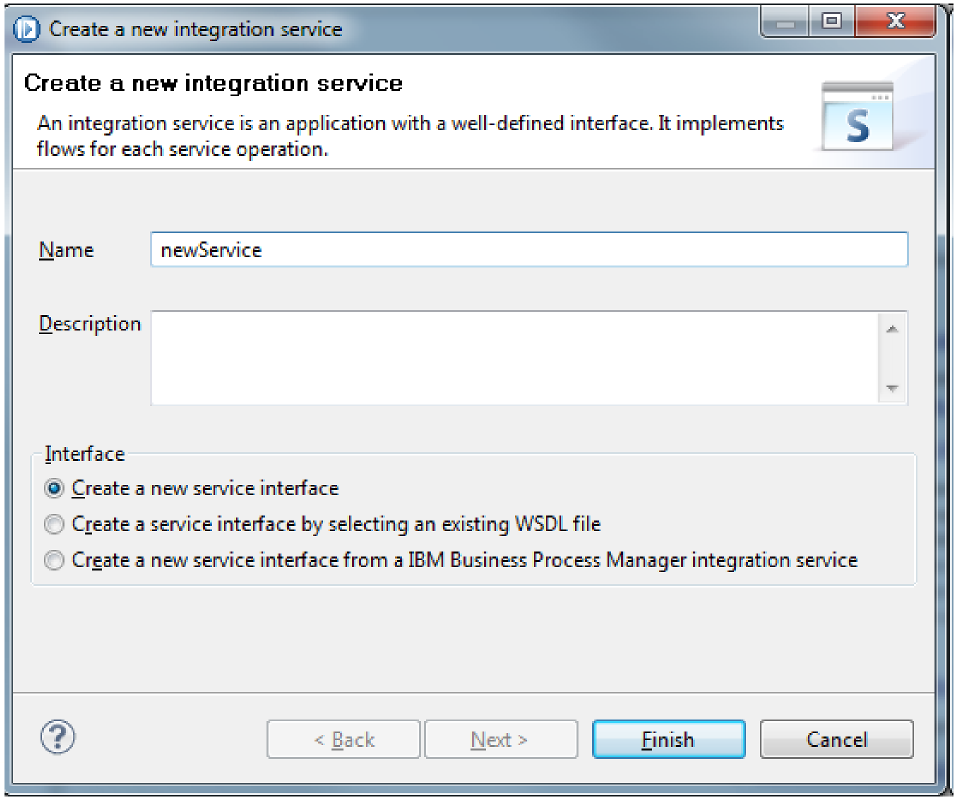
- Within IIB, easily create new services that have a well defined interface and structure
  - A whole new service from scratch including the associated WSDL and message flow
  - A service including the message flows, from an existing WSDL file
  - A service, including the message flows, from an existing IBM BPM service
  - A service based on an existing database
  - A service based on an existing MQ queue manager and queue definition

Creating a new service

Start by creating an integration service

An **Integration Service** is an application with a well-defined interface and structure. Like an application, it is a container for resources required to create a Web Services solution. These resources provide the implementation of the interface with specified operations that you implement as separate message subflows. You can deploy a service. A web service consumer can interrogate the deployed service to return its interface. Collapse

use the wizard to create a brand new service



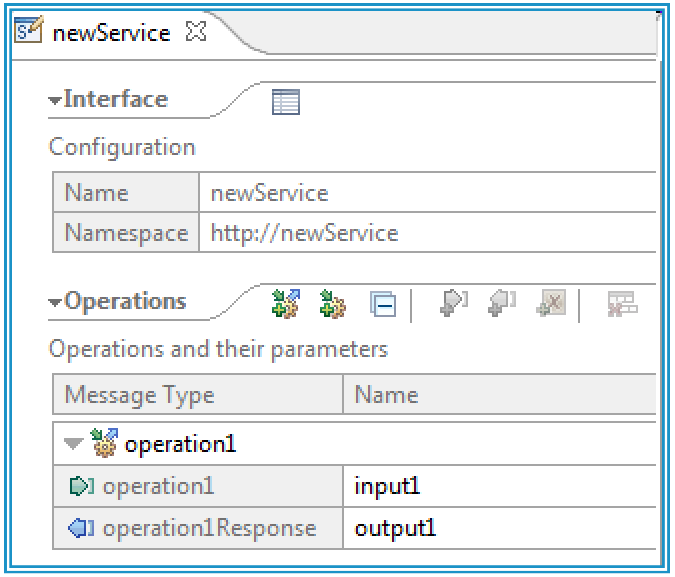
Create a new integration service

Create a new integration service

An integration service is an application with a well-defined interface. It implements flows for each service operation.

Name	newService
Description	

Interface
- [✓ Create a new service interface]
- Create a service interface by selecting an existing WSDL file
- Create a new service interface from a IBM Business Process Manager integration service



newService

Interface

Configuration

Name	newService
Namespace	http://newService

Operations

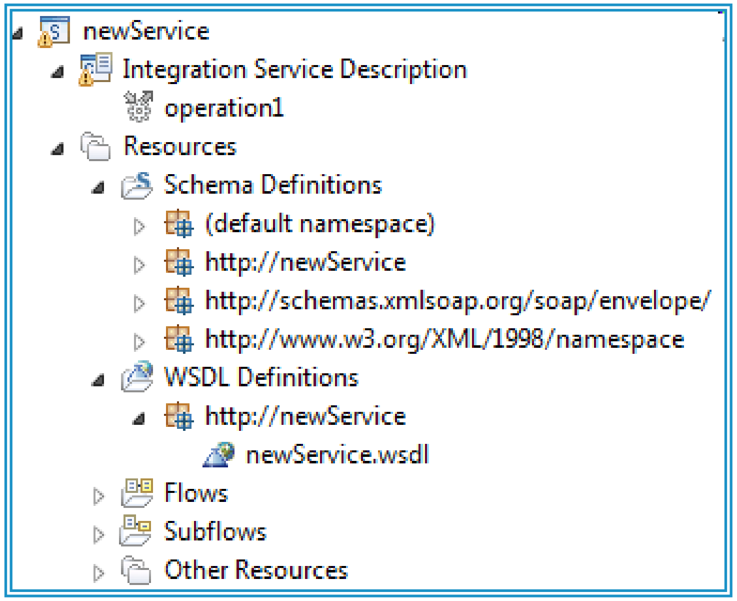
Operations and their parameters

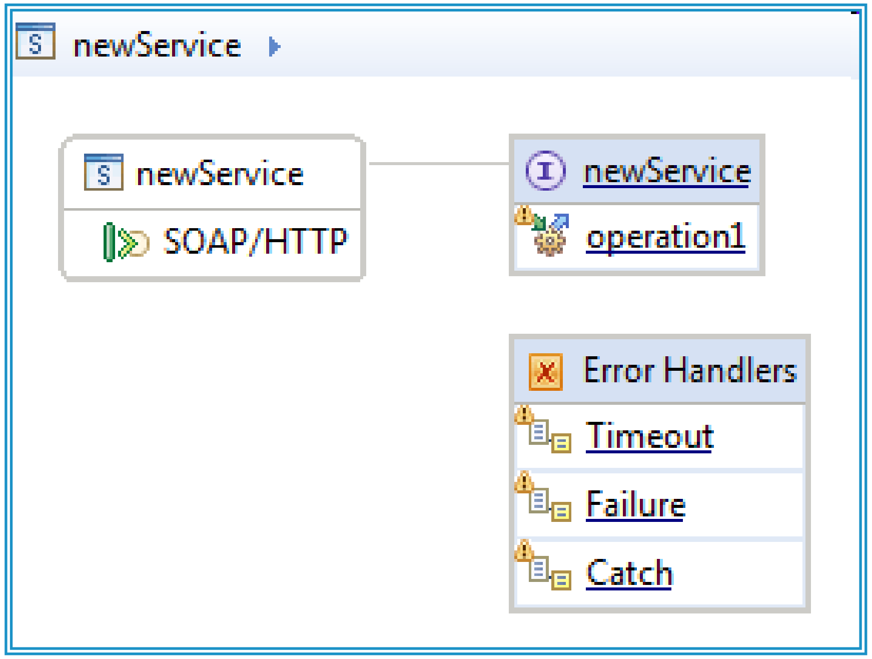
Message Type	Name
operation1	
operation1	input1
operation1Response	output1

Service Artefacts

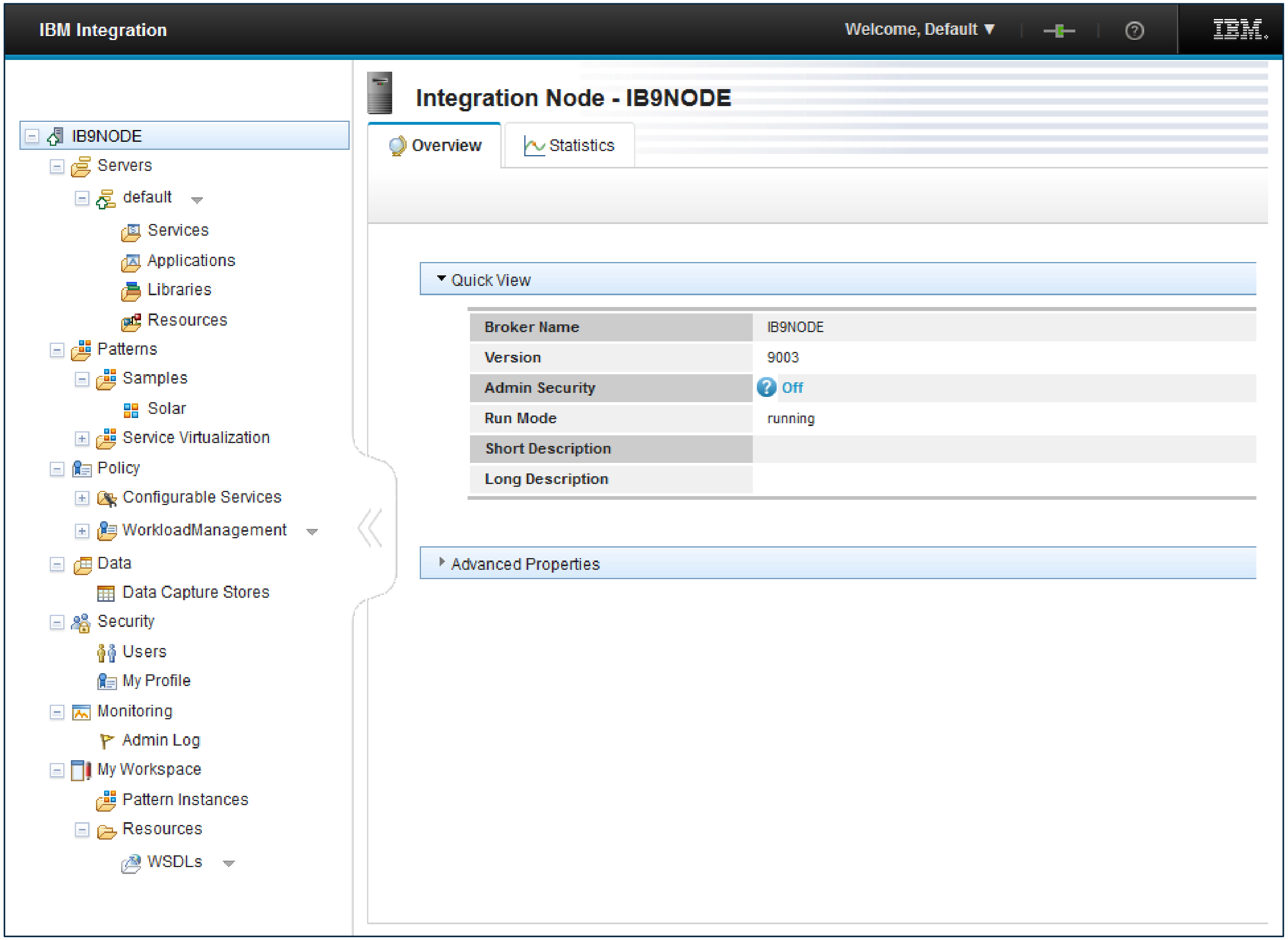
- all artefacts are created for the new service, including the WSDL and a SOAP based flow

- a subflow is included for the operation and error handlers which the developer can then update


newService
Integration Service Description
operation1
Resources
Schema Definitions
(default namespace)
http://newService
http://schemas.xmlsoap.org/soap/envelope/
http://www.w3.org/XML/1998/namespace
WSDL Definitions
http://newService
newService.wsdl
Flows
Subflows
Other Resources



Administration: WebUI

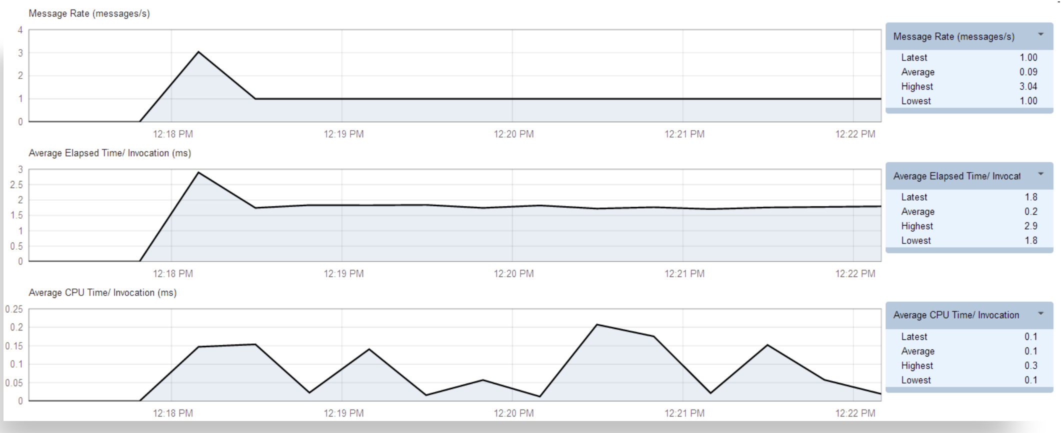


Integration Node - IB9NODE

Broker Name	IB9NODE
Version	9003
Admin Security	Off
Run Mode	running
Short Description	
Long Description	

View runtime statistics using the WebUI

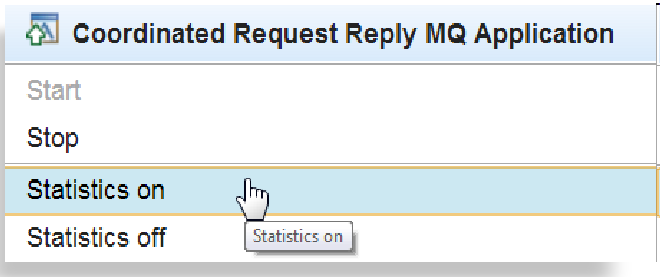
- control statistics at all levels
- easily view and compare flows, helping to understand which are processing the most messages or have the highest elapsed time
- easily view and compare nodes, helping to understand which have the highest CPU or elapsed times.
- view all statistics metrics available for each flow

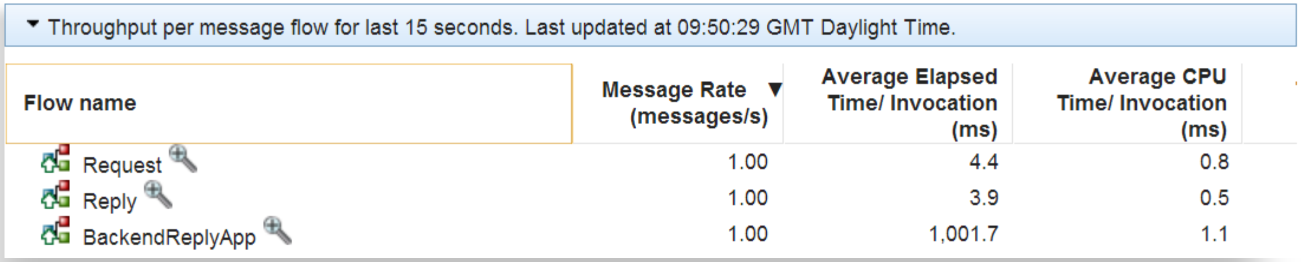


Message Rate (messages/s)	
Latest	1.00
Average	0.09
Highest	3.04
Lowest	1.00

Average Elapsed Time/ Invocat	
Latest	1.8
Average	0.2
Highest	2.9
Lowest	1.8

Average CPU Time/ Invocation	
Latest	0.1
Average	0.1
Highest	0.3
Lowest	0.1



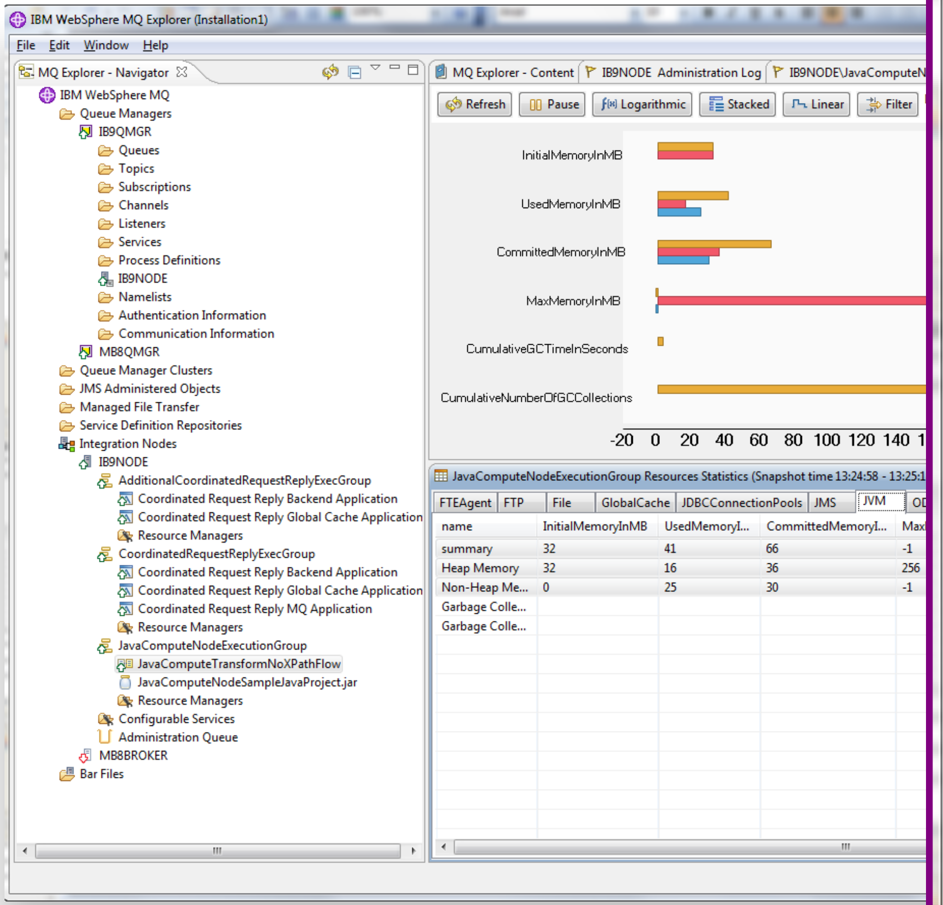


▾ Throughput per message flow for last 15 seconds. Last updated at 09:50:29 GMT Daylight Time.

Flow name	Message Rate ▼ (messages/s)	Average Elapsed Time/ Invocation (ms)	Average CPU Time/ Invocation (ms)
Request	1.00	4.4	0.8
Reply	1.00	3.9	0.5
BackendReplyApp	1.00	1,001.7	1.1

Integration Explorer

fully functional administration tool based on MQ Explorer

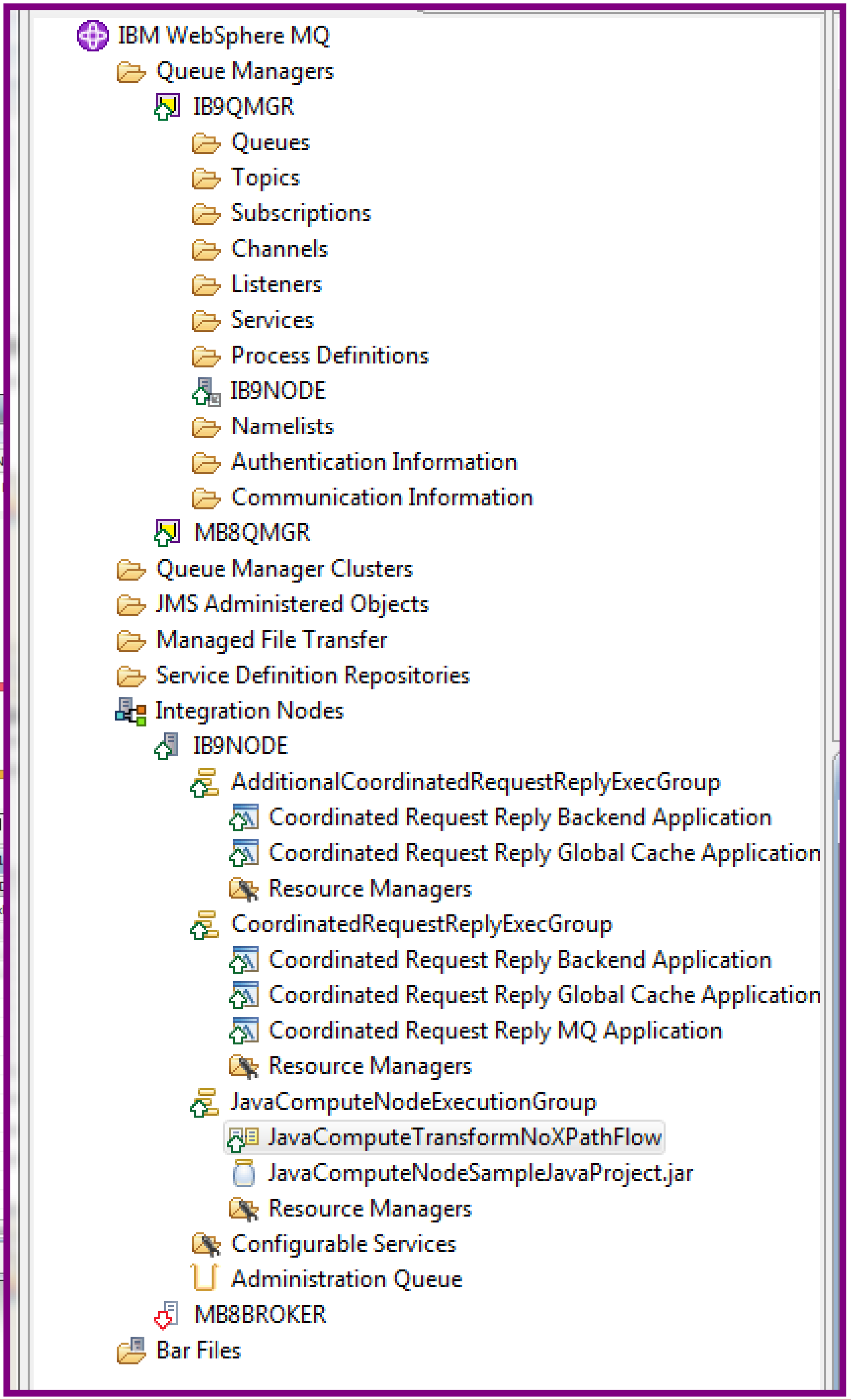


- IBM WebSphere MQ
  - Queue Managers
    - IB9QMGR
      - Queues
      - Topics
      - Subscriptions
      - Channels
      - Listeners
      - Services
      - Process Definitions
      - IB9NODE
      - Namelists
      - Authentication Information
      - Communication Information
    - MB8QMGR
  - Queue Manager Clusters
  - JMS Administered Objects
  - Managed File Transfer
  - Service Definition Repositories
  - Integration Nodes
    - IB9NODE
      - AdditionalCoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Resource Managers
      - CoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Coordinated Request Reply MQ Application
        - Resource Managers
      - JavaComputeNodeExecutionGroup
        - JavaComputeTransformNoXPathFlow
        - JavaComputeNodeSampleJavaProject.jar
        - Resource Managers
      - Configurable Services
      - Administration Queue
    - MB8BROKER
  - Bar Files

JavaComputeNodeExecutionGroup Resources Statistics (Snapshot time 13:24:58 - 13:25:1

FTEAgent | FTP | File | GlobalCache | JDBCConnectionPools | JMS | JVM | OD

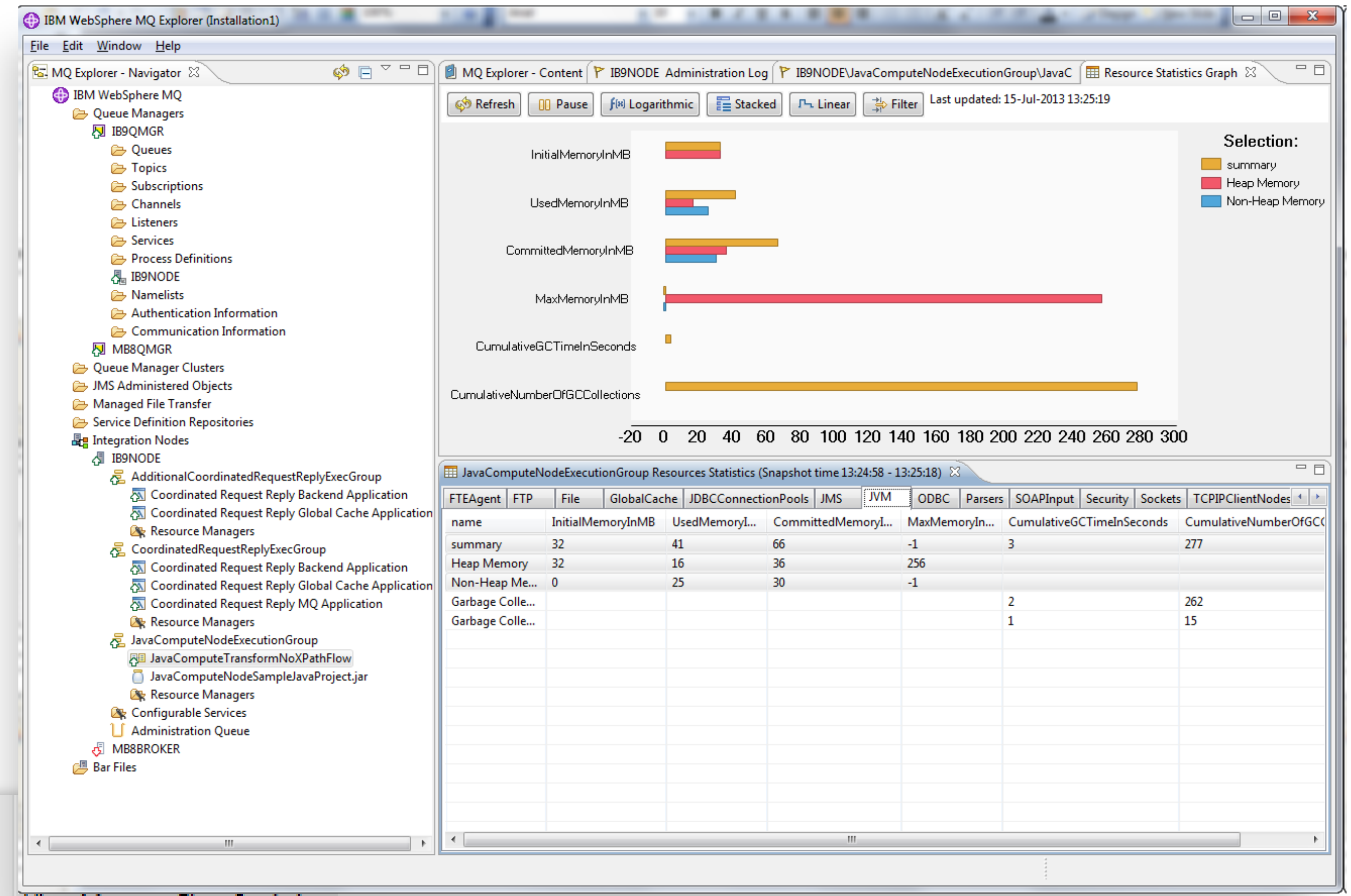
name	InitialMemoryInMB	UsedMemoryI...	CommittedMemoryI...	Max
summary	32	41	66	-1
Heap Memory	32	16	36	256
Non-Heap Me...	0	25	30	-1
Garbage Colle...				
Garbage Colle...				



- IBM WebSphere MQ
  - Queue Managers
    - IB9QMGR
      - Queues
      - Topics
      - Subscriptions
      - Channels
      - Listeners
      - Services
      - Process Definitions
      - IB9NODE
      - Namelists
      - Authentication Information
      - Communication Information
    - MB8QMGR
  - Queue Manager Clusters
  - JMS Administered Objects
  - Managed File Transfer
  - Service Definition Repositories
  - Integration Nodes
    - IB9NODE
      - AdditionalCoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Resource Managers
      - CoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Coordinated Request Reply MQ Application
        - Resource Managers
      - JavaComputeNodeExecutionGroup
        - JavaComputeTransformNoXPathFlow
        - JavaComputeNodeSampleJavaProject.jar
        - Resource Managers
      - Configurable Services
      - Administration Queue
    - MB8BROKER
  - Bar Files

Integration Explorer & Resource Statistics

view resource statistics for resource managers in IIB such as JVM, ODBC, JDBC, etc.

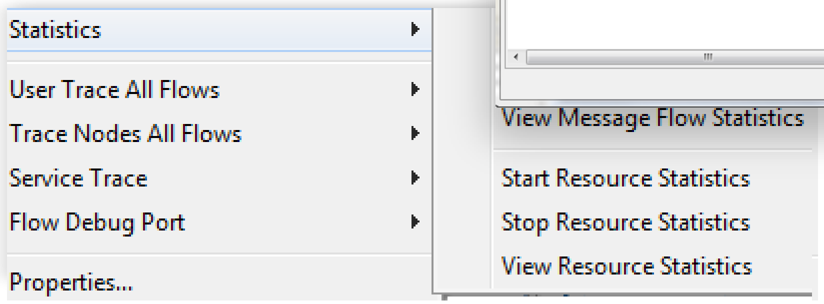


- IBM WebSphere MQ
  - Queue Managers
    - IB9QMGR
      - Queues
      - Topics
      - Subscriptions
      - Channels
      - Listeners
      - Services
      - Process Definitions
      - IB9NODE
      - Namelists
      - Authentication Information
      - Communication Information
    - MB8QMGR
  - Queue Manager Clusters
  - JMS Administered Objects
  - Managed File Transfer
  - Service Definition Repositories
  - Integration Nodes
    - IB9NODE
      - AdditionalCoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Resource Managers
      - CoordinatedRequestReplyExecGroup
        - Coordinated Request Reply Backend Application
        - Coordinated Request Reply Global Cache Application
        - Coordinated Request Reply MQ Application
        - Resource Managers
      - JavaComputeNodeExecutionGroup
        - JavaComputeTransformNoXPathFlow
        - JavaComputeNodeSampleJavaProject.jar
        - Resource Managers
      - Configurable Services
      - Administration Queue
    - MB8BROKER
  - Bar Files

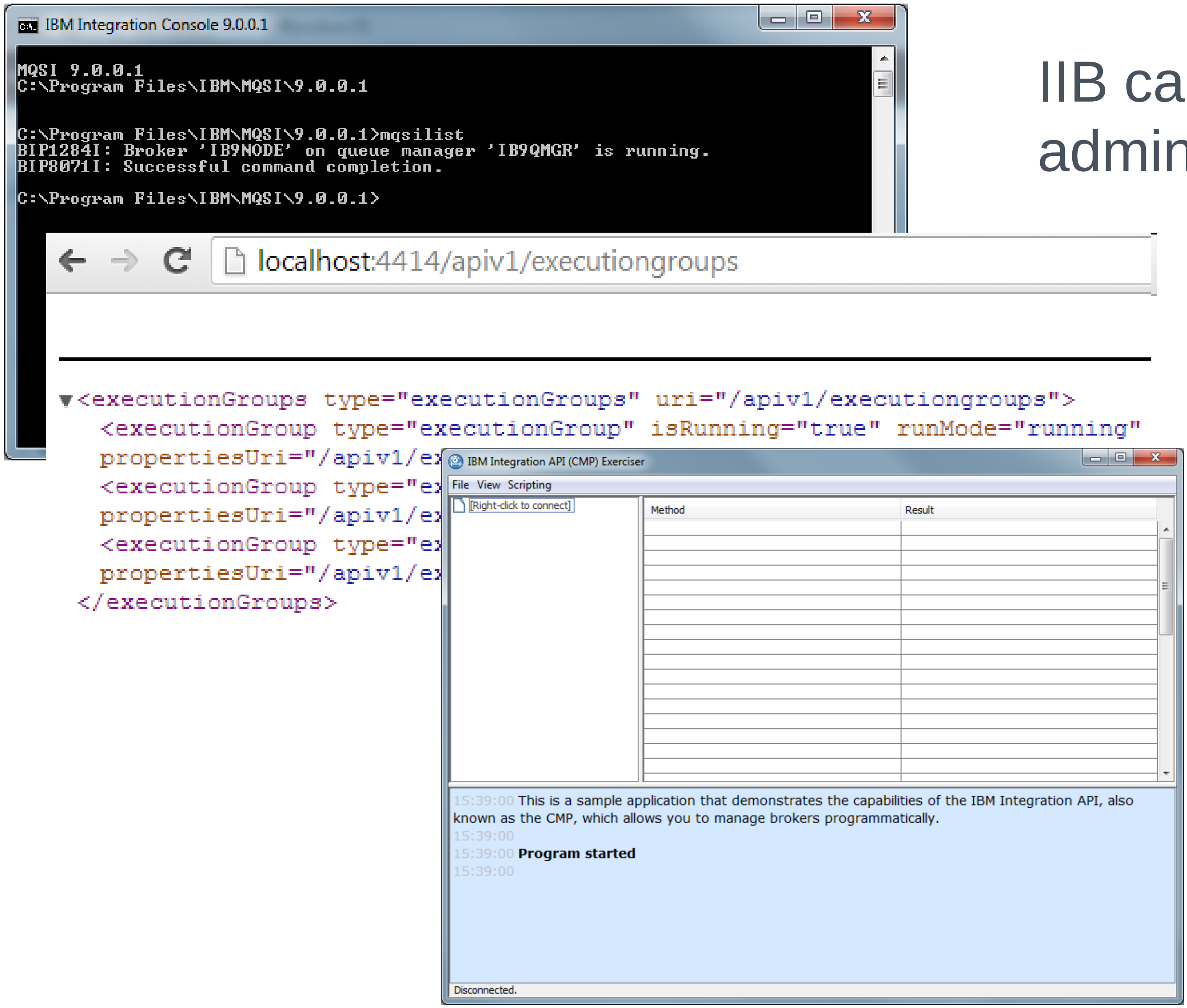
JavaComputeNodeExecutionGroup Resources Statistics (Snapshot time 13:24:58 - 13:25:18)

FTEAgent | FTP | File | GlobalCache | JDBCConnectionPools | JMS | JVM | ODBC | Parsers | SOAPInput | Security | Sockets | TCPIPClientNodes

name	InitialMemoryInMB	UsedMemoryI...	CommittedMemoryI...	MaxMemoryIn...	CumulativeGCTimeInSeconds	CumulativeNumberOfGC(
summary	32	41	66	-1	3	277
Heap Memory	32	16	36	256		
Non-Heap Me...	0	25	30	-1		
Garbage Colle...					2	262
Garbage Colle...					1	15



Other forms of administration


IBM Integration Console 9.0.0.1
IBM Integration API (CMP) Exerciser
File View Scripting
[Right-click to connect]
Method
Result

```
MQSI 9.0.0.1
C:\Program Files\IBM\MQSI\9.0.0.1

C:\Program Files\IBM\MQSI\9.0.0.1>mqsilist
BIP1284I: Broker 'IB9NODE' on queue manager 'IB9QMGR' is running.
BIP8071I: Successful command completion.

C:\Program Files\IBM\MQSI\9.0.0.1>
```

localhost:4414/apiv1/executiongroups

```
▼<executionGroups type="executionGroups" uri="/apiv1/executiongroups">
   <executionGroup type="executionGroup" isRunning="true" runMode="running"
   propertiesUri="/apiv1/ex
   <executionGroup type="ex
   propertiesUri="/apiv1/ex
   <executionGroup type="ex
   propertiesUri="/apiv1/ex
 </executionGroups>
```

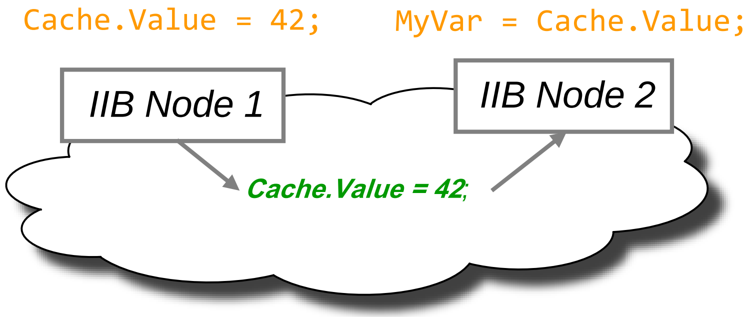
```
15:39:00 This is a sample application that demonstrates the capabilities of the IBM Integration API, also
known as the CMP, which allows you to manage brokers programmatically.
15:39:00
15:39:00 Program started
15:39:00
```

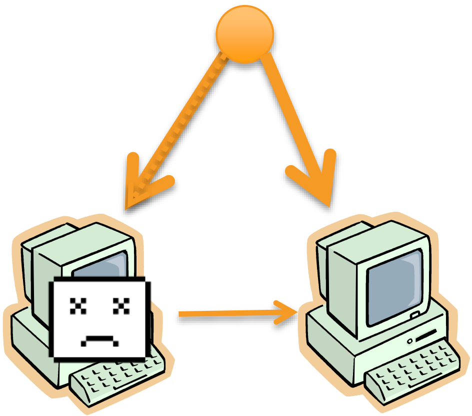
Disconnected.

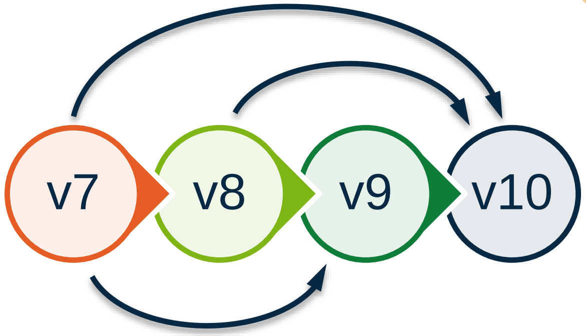
IIB can also be administered using

- command line
- REST interface
- Java API

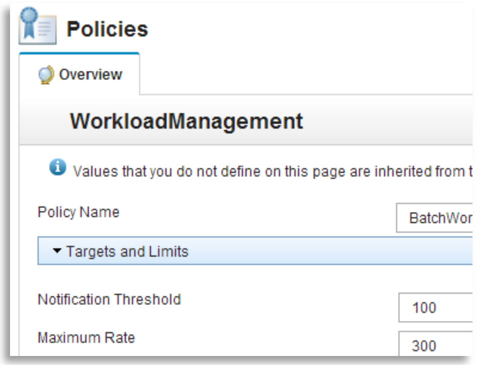
Some other useful features







Chef cookbooks for ot4i
ot4i-cookbooks



Policies

Overview

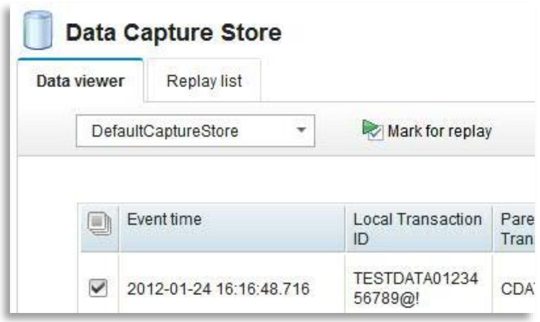
WorkloadManagement

Values that you do not define on this page are inherited from t

Policy Name	BatchWor

▼ Targets and Limits

Notification Threshold	100
Maximum Rate	300



Data Capture Store

Data viewer | Replay list

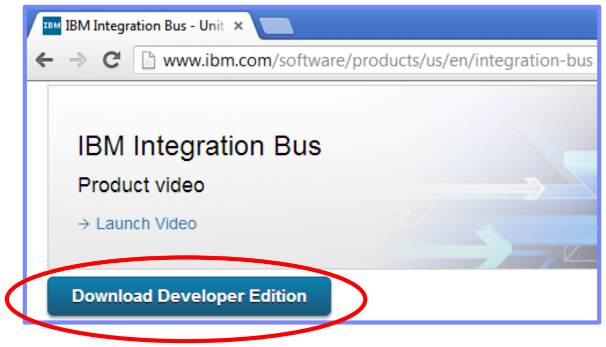
DefaultCaptureStore ▾ | Mark for replay

	Event time	Local Transaction ID	Pare Tran
✓	2012-01-24 16:16:48.716	TESTDATA01234 56789@!	CDA

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Developer Edition: download today!

- free edition of IIB with all nodes available and no time limitations
- message rate limited to 1 TPS per integration flow
- assistance through user community (e.g. mqseries.net)
  - no formal IBM support
- simple to download, install and use
  - single installation package contains ALL required software:
    - MQ 7.5, Integration Bus (Runtime, Toolkit, Explorer)
  - available on Windows and Linux platforms


IBM Integration Bus - Unit
www.ibm.com/software/products/us/en/integration-bus
IBM Integration Bus
Product video
Launch Video
Download Developer Edition

Industry Specific Solutions

Industry Specific Packs providing relevant support for applications in certain industries

- build on IBM Integration Bus functionality
- contain message models / patterns / nodes
- web-based operational monitoring

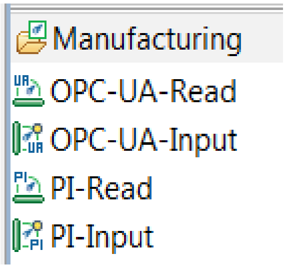
V3.0.0.1 Aug 2014 | V1.0.0.0 Jun 2014 | V1.0.0.0 Dec 2013

Healthcare

- HL7 Message Model
- DICOM / ATNA / Medical Device Integration Nodes

Retail

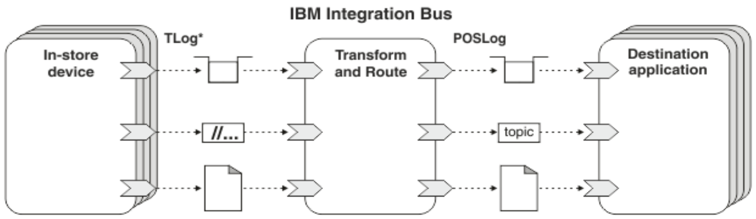
- WebSphere Commerce and Sterling Order Management integration
- TLog / POSLog / ARTS transformation and storage



Manufacturing
OPC-UA-Read
OPC-UA-Input
PI-Read
PI-Input

Manufacturing

- OPC / PI / MQTT nodes

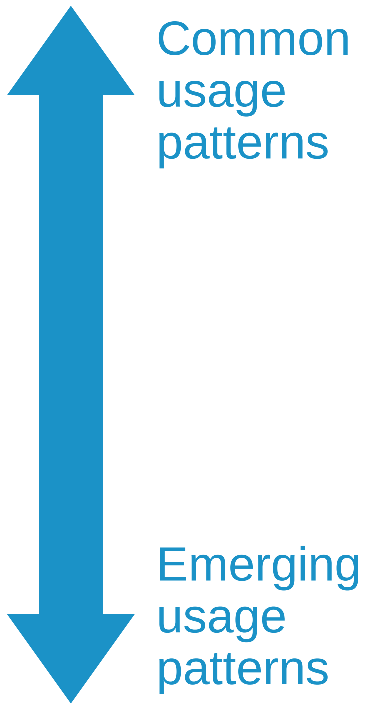


Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Top Integration Usage Patterns

What are the top issues that people want to solve with integration solutions?

- Extend the Reach of Existing Applications
- Distribute Database information to where it’s needed
- Create a File Hub to connect batch and online applications
- Get the most from Packaged Applications
- Take advantage of .NET applications
- Provide a Policy Enforcement Point for Secure Connectivity
- Extend Enterprise to Devices and Mobile
- Monitor your business activity and act intelligently
- Detect and Act Upon Business Events and Rules
- Provide Connectivity and Integration for Business Processes
- Make an inventory and enable Policy based management

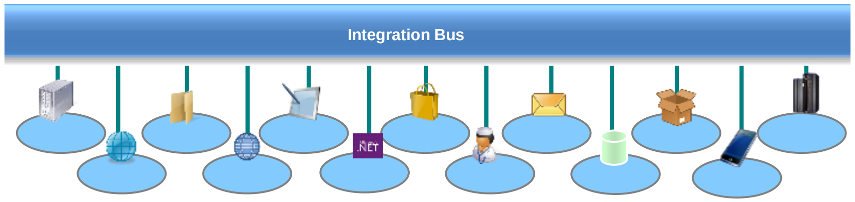


New usage patterns are continually emerging as business needs evolve!

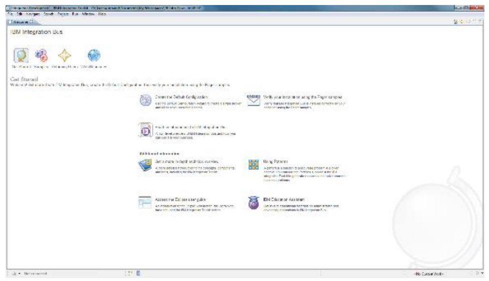
IBM Integration Bus

IBM's Strategic Integration Technology

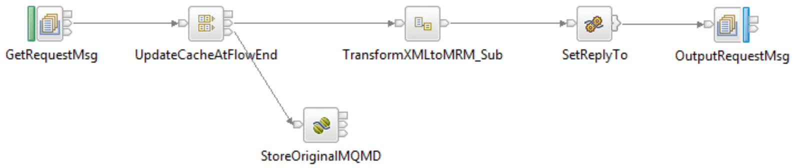
Universal Connectivity FROM anywhere, TO anywhere
- simplify application connectivity for a flexible & dynamic infrastructure



Protocols, Transports, Data Formats & Processing
- Supports a wide range of built-in transports, protocols & systems
  - MQ, JMS 1.1, HTTP(S), SOAP, REST, File (incl. FTP & FTE), Database, TCP/IP, MQTT…
  - CICS, IMS, SAP, SIEBEL, PeopleSoft, JDEdwards, SCA, CORBA, email…
- Supports a broad range of data formats
  - Binary (C/COBOL), XML, CSV, JSON, Industry (SWIFT, EDI, HL7…), IDOCs, User Defined
- Message Processors
  - Route, Filter, Transform, Enrich, Monitor, Publish, Decompose, Sequence, Correlate, Detect…



Simple Programming with Patterns & Graphical Data Flows
- patterns for top-down, parameterized connectivity of common use cases
- graphical data flows represent application & service connectivity
  - custom logic via graphical mapping, PHP, Java, ESQL, XSL & .NET
- Services, Applications and Libraries for structure and reuse



Extensive Management, Performance & Scalability
- extensive administration & systems management facilities for developed solutions
- wide range of operating system and hardware platforms supported, including virtual & cloud options
- high performance transactional processing, additional vertical & horizontal scalability
- deployment options include Trial, Express, Standard and Advanced

Message Rate (messages/s)
Average Elapsed Time/ Invocation (ms)
Average CPU Time/ Invocation (ms)
Latest
Average
Highest
Lowest

Industry Packs for Industry Specific Content

Questions?

Trademark Statement