WebSphere Message Broker Best Practice

How to develop Message Flows with a reusable Framework for Errorhandling, Logging and Routing

Sabine Busch
(Certified IT-Specialist, ISSW Germany, buschs@de.ibm.com)

Jan-Philipp Schiller
(IT-Specialist, ISSW Germany, jan.schiller@de.ibm.com)

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

6 Lessons learned

7 Outlook

8 Questions

Motivation and preconditions

- Framework was designed during a customer engagement in collaboration with the customer
- Current customer situation: Isolated flows which grew in complexity in course of time are difficult to maintain
- Use of standard WMB 7.0 product and features
- Customer: Different developers with various skill levels (WMB and WMQ)
- Reduce development time
- Platform reuse for future projects
- Change routing information during runtime of WebSphere Message Broker
- Usable for the “most” WMB scenarios

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

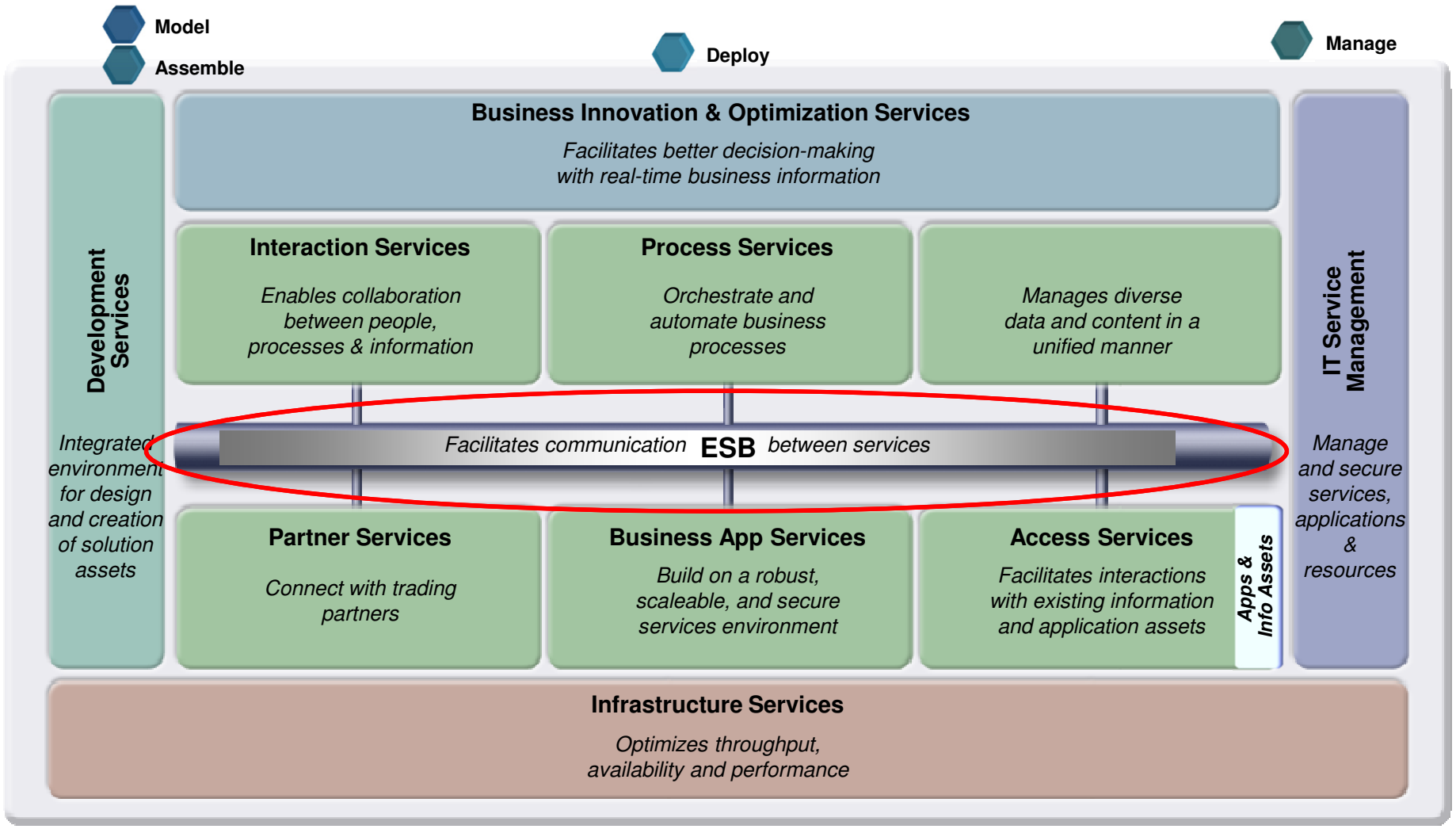
5 Demo

6 Lessons learned

7 Outlook

8 Questions

IBM SOA Reference Architecture



Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

6 Lessons learned

7 Outlook

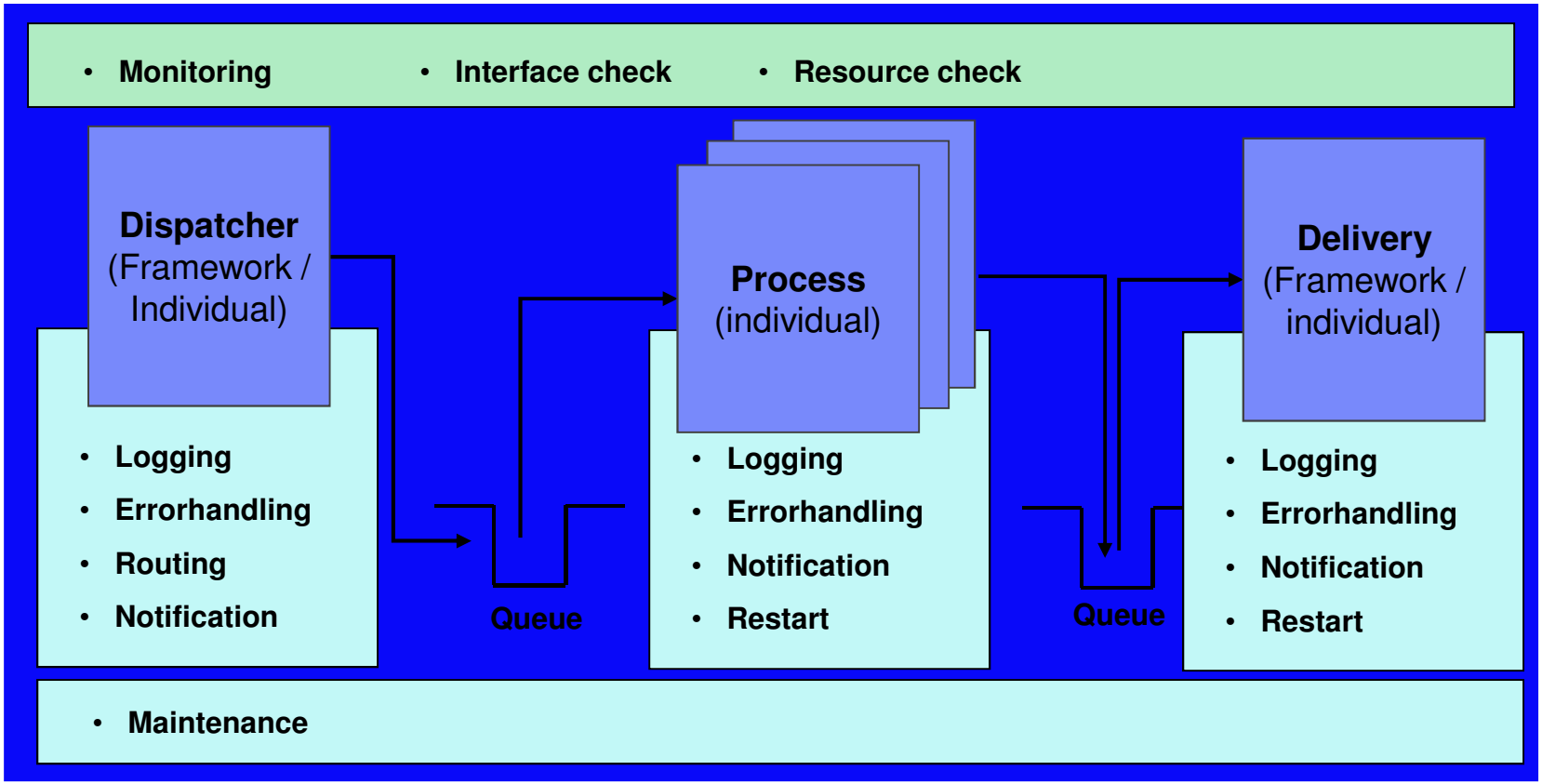
8 Questions

Why a framework?

- Out-of-the-box WMB does not provide common services like
  - Errorhandling
  - Routing
  - Logging
  - Restarting
  - Notification
  - ...
- Such services have to be developed in nearly every customer engagement
- Make use of identified patterns
- Reuse of components
- Reduce effort and development time
- Developers should concentrate on process flow (e.g. Mapping, tranformation, sequencing etc.)

High level view of a generic framework application

ESB Framework part **GUI / Framework console part**



Functional overview

- Service driven framework
- Separation into the following services and components (in brackets)
  - Dispatcher
  - Processing
  - Delivery
  - (Value Mapping)
  - (Errorhandling)
  - (Logging)
  - Notification
  - SAP-Status
  - Restart
  - Maintenance
  - Monitoring
  - Sequencing
- Toolbox: Flows as templates
- Configuration database (key component)
- Administration GUI

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture /ESB related patterns

3 Why a Framework?!

4 Framework architecture

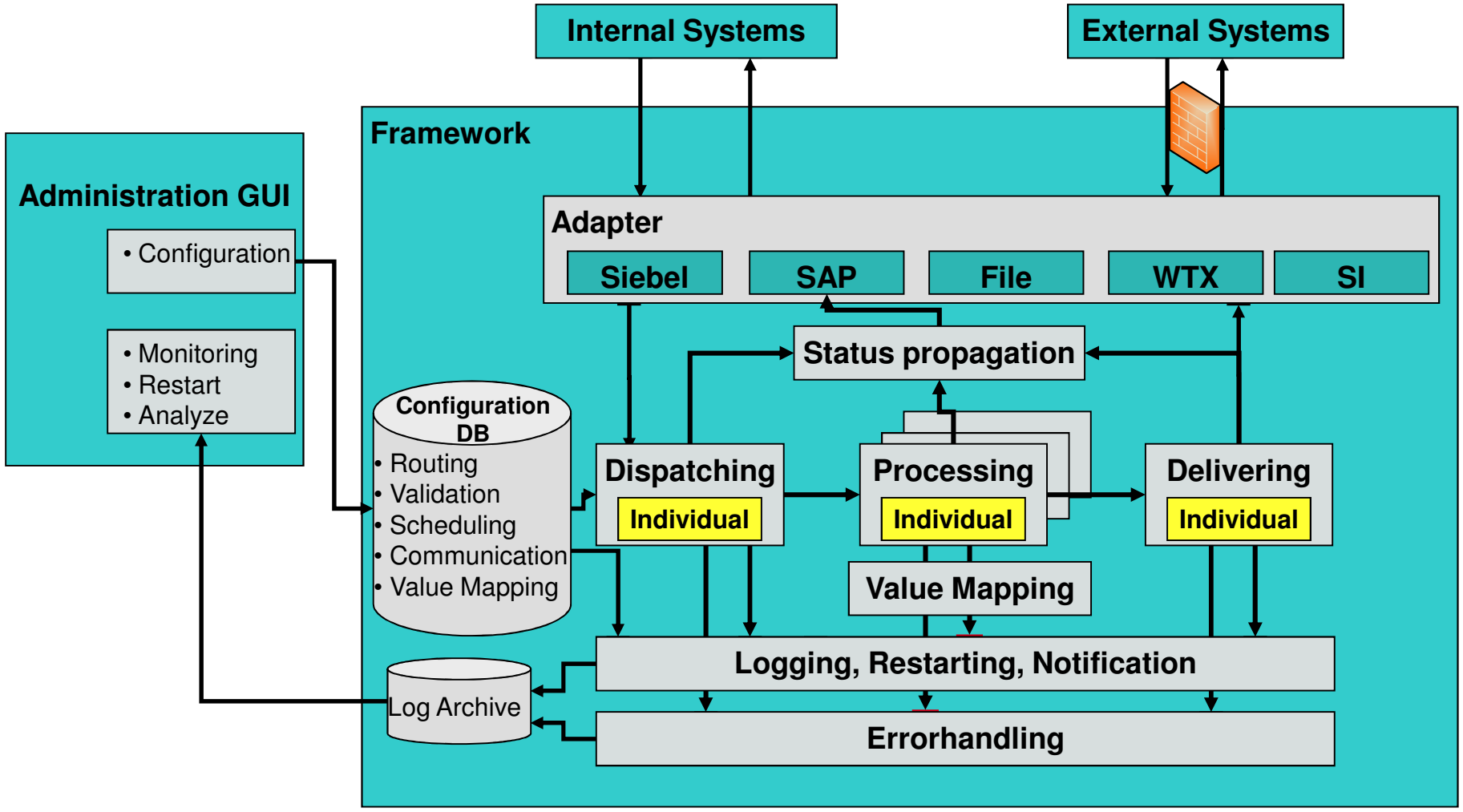
5 Demo

6 Lessons learned

7 Outlook

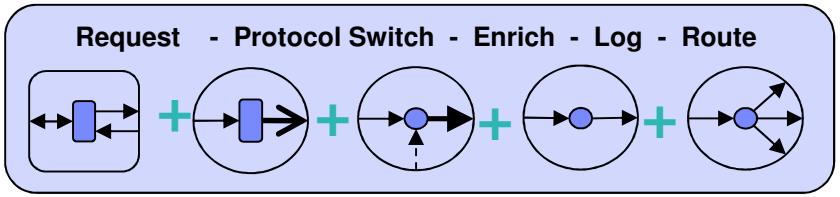
8 Questions

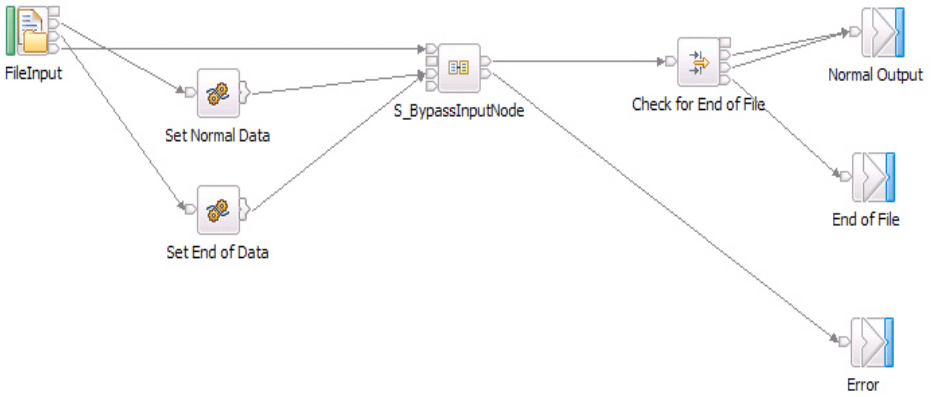
Framework architecture



Service: Dispatcher (I)

- Receiving Messages from internal and external Systems via
  - WebSphere MQ
  - HTTP/S
  - Flat File (via FTP)
  - SAP Adapter
  - Siebel Adapter
  - SQL
  - WebSphere TX
  - SOAP
  - JMS
  - Extendable

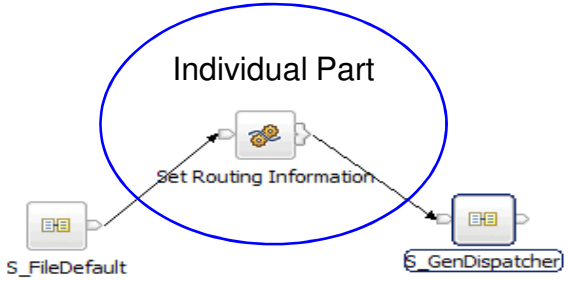




Sub Flow for Dispatcher

Service: Dispatcher (II)

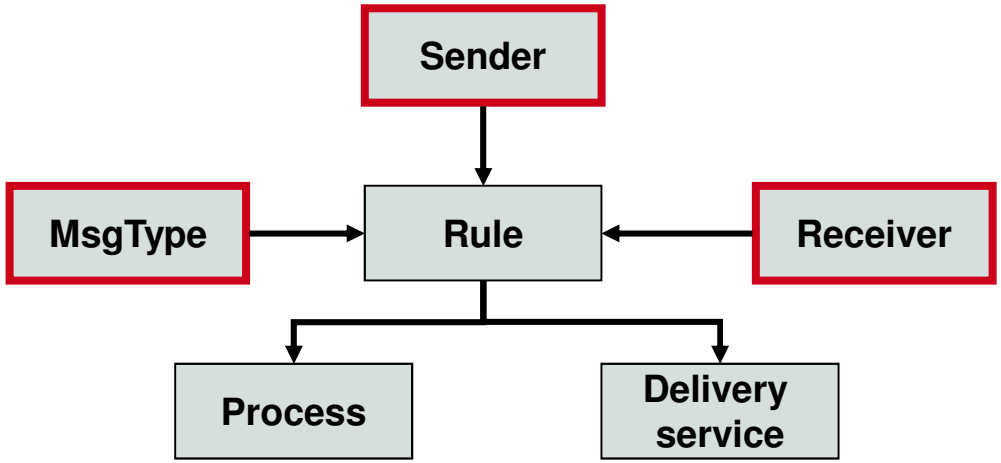
- Central point for message processing
- Identifies sender, message type, receiver and determines process and delivery flow
- Determine routing information for further processing in individual part of the flow or set directly
- Routing information regarding process is maintained in the configuration database via GUI. The dispatcher service extracts this out of the configuration database
  → **only this service accesses the configuration DB**
- Dispatching service stores message via logging service into log-event database
- Sends message to next process step
- In case of error: logs erroneous message, its metadata as well as error reason via error handling for a later restart



Example of Dispatcher Flow

Service: Dispatcher (III) – Rules for routing

Routing defines the path a message takes through the MessageFlows. A path can be assembled dynamically by changing the settings in the database via the GUI.



From SAP (Example)

Sender is extracted from SAP Header

Receiver is extracted from SAP Header

MsgType is extracted from SAP Header

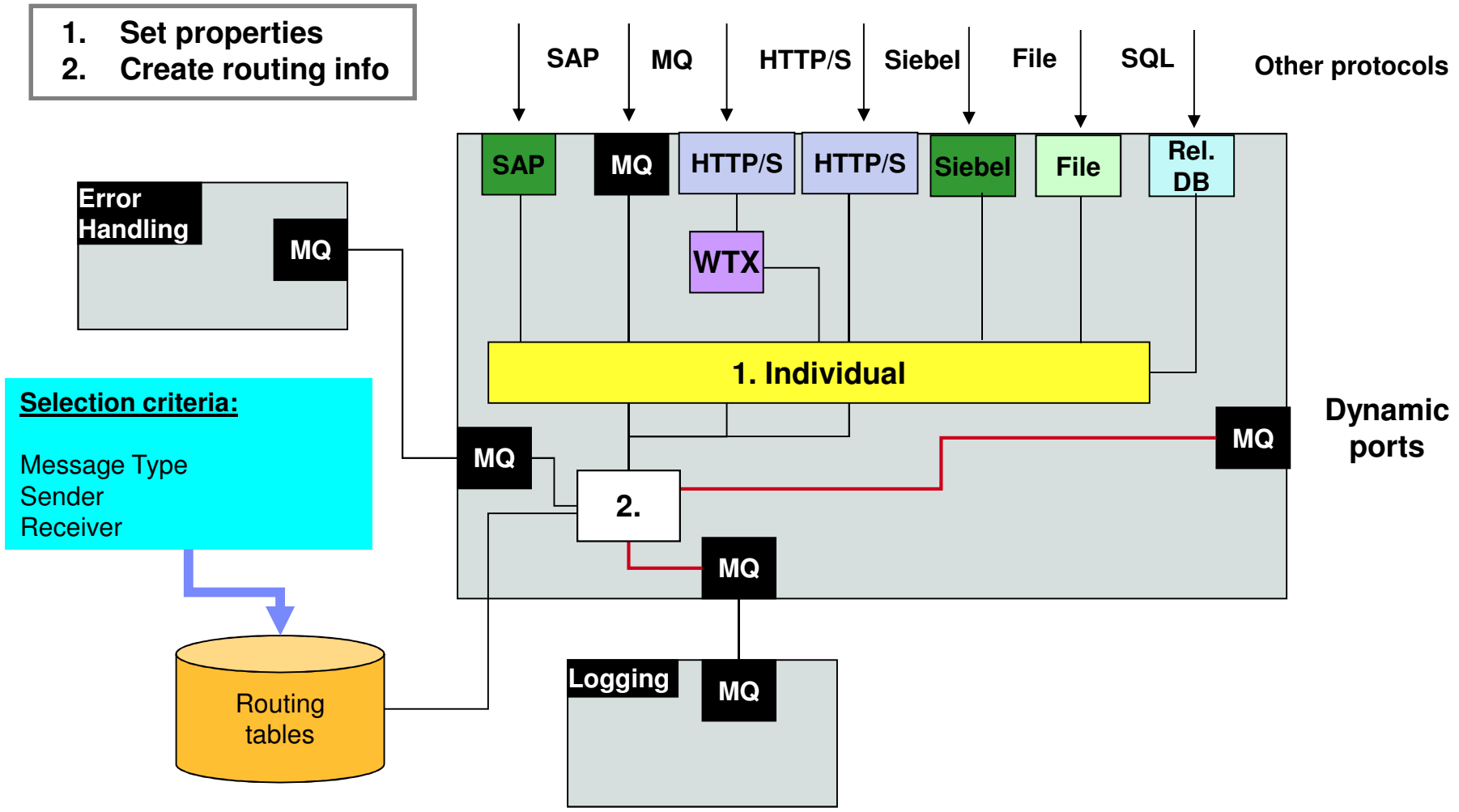
From external (Example)

Sender is extracted from message, URL or MQ name

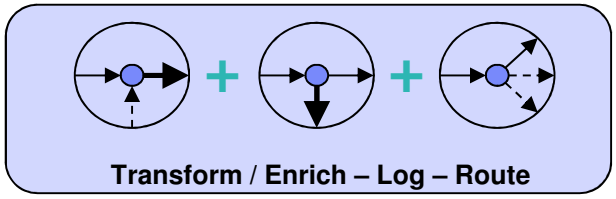
Receiver is extracted from message, URL

MsgType is extracted from message / URL

Service: Dispatcher (IV)

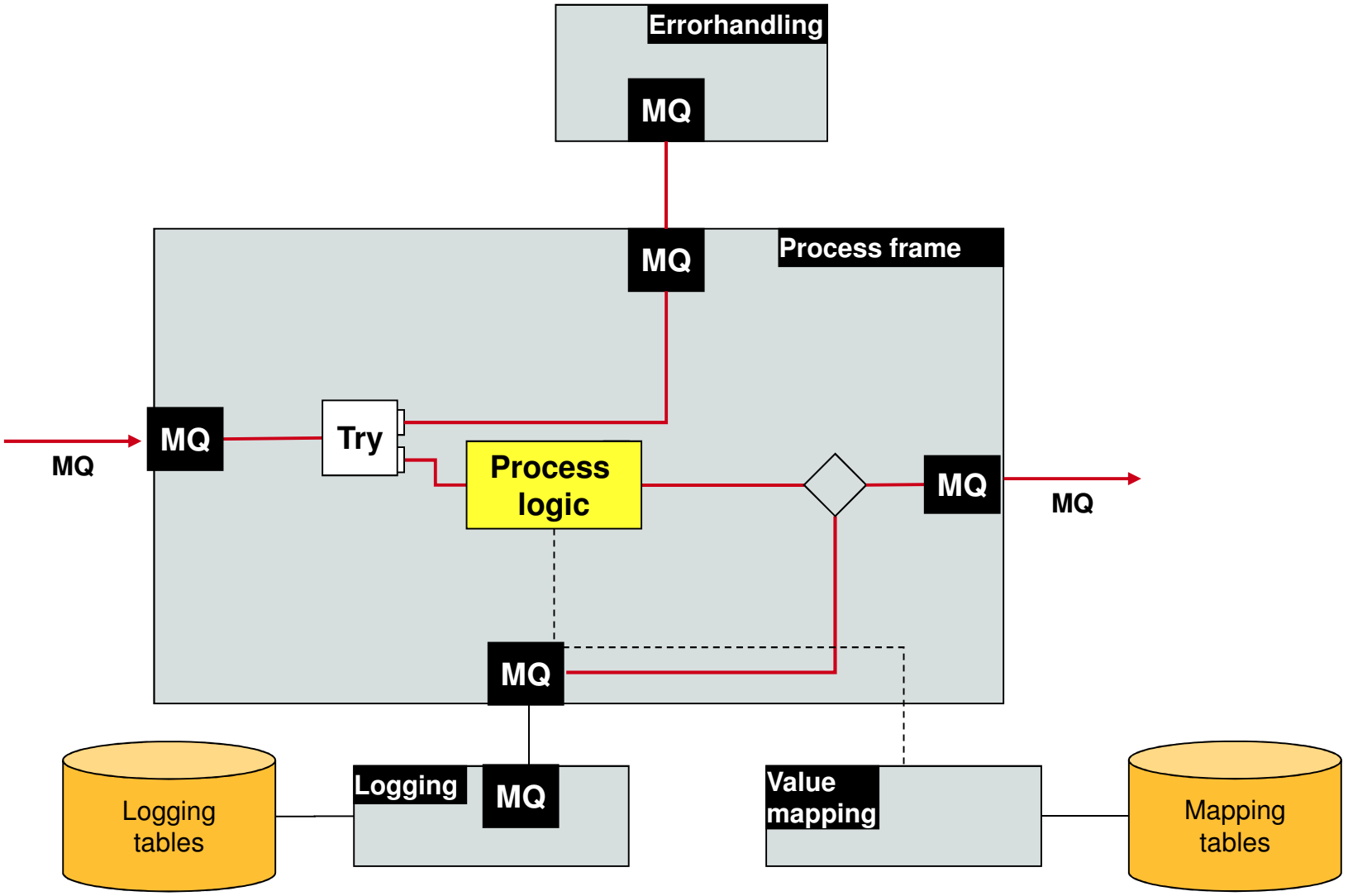


Service: Processing (I)

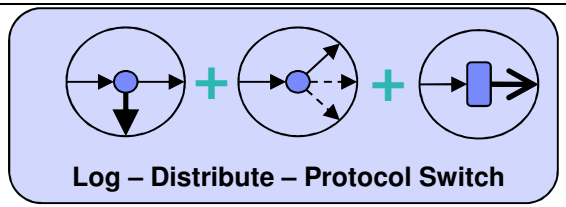


- Responsible for message format transformation and process logic (sequencing, collecting, mapping, …)
- Is **not** part of the framework. The developer has to develop the process flows
- Receives XML and delivers XML (XMLNSC parser)
- MQ entry point (MQ InputNode)
- Uses common framework services and components
- Each process flow is decoupled by MQ queues at start and end of processing
- Logging is done via logging service → generates a log message
- Sends message to next the process step (e.g. Delivery)
- Error case:
  Logs erroneous message, its metadata as well as the error reason via the errorhandling component for a possible later restart (restart service)

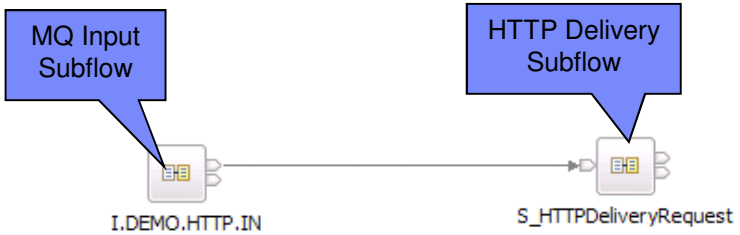
Service: Processing (II)



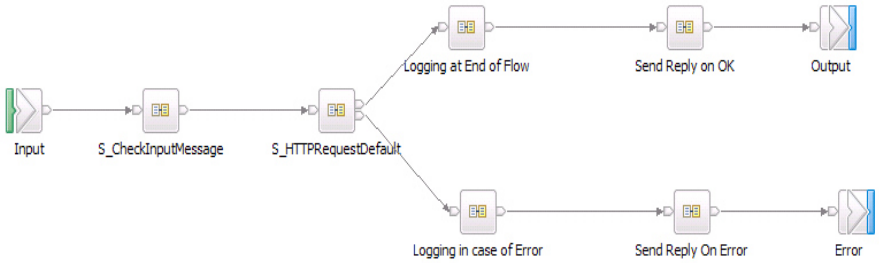
Service: Delivery (I)



- Receives XML from processing service
- MQ entry point
- Sends message payload to assigned adapter according to metadata in MQRFH2 header
- Uses errorhandling, notification and logging service
- Delivery service stores message via logging service
- In case of error logs erroneous message, its metadata as well as error reason via the errorhandling for a later restart
- Sends message to the final application in the correct protocol

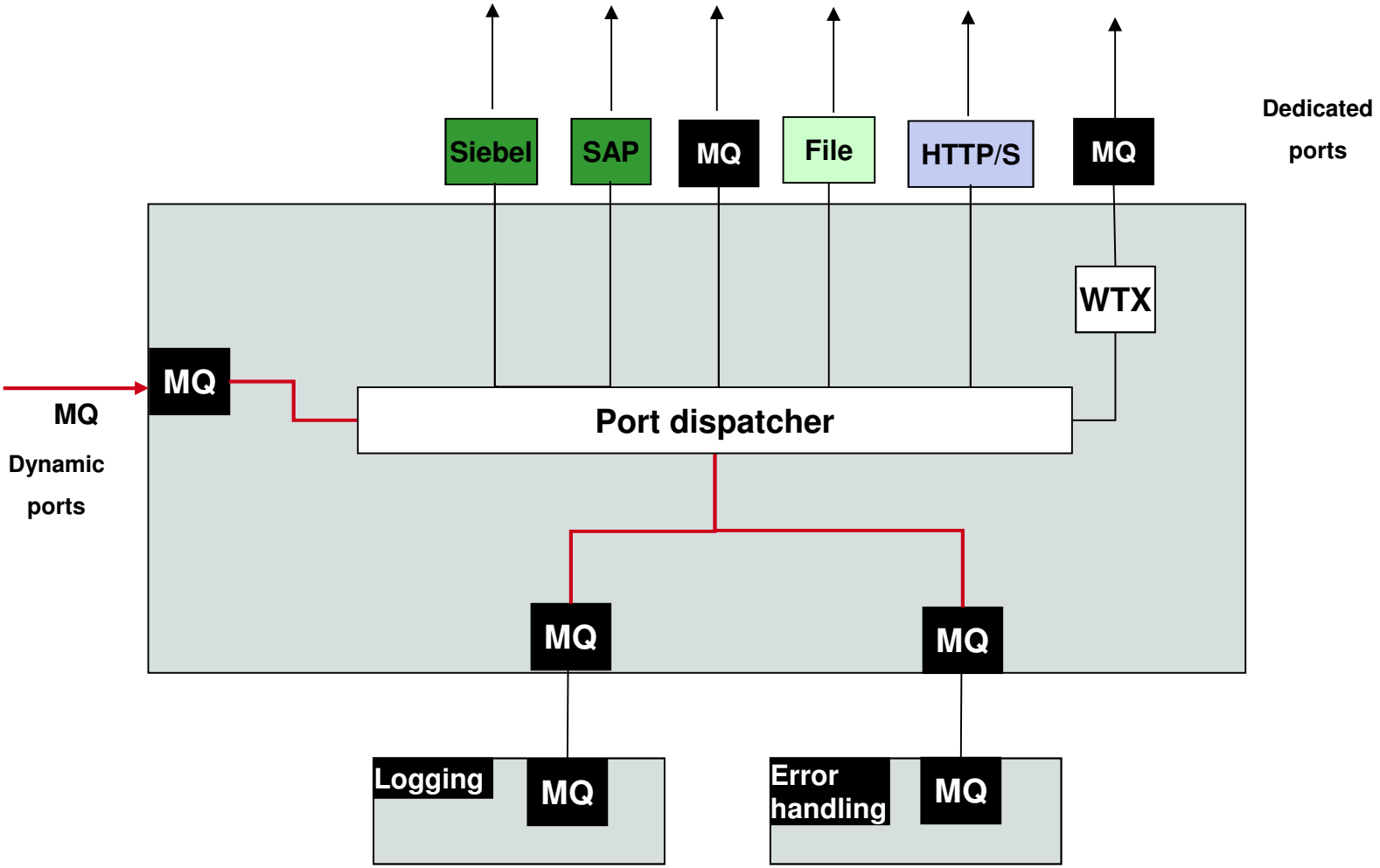


Example of Delivery Flow

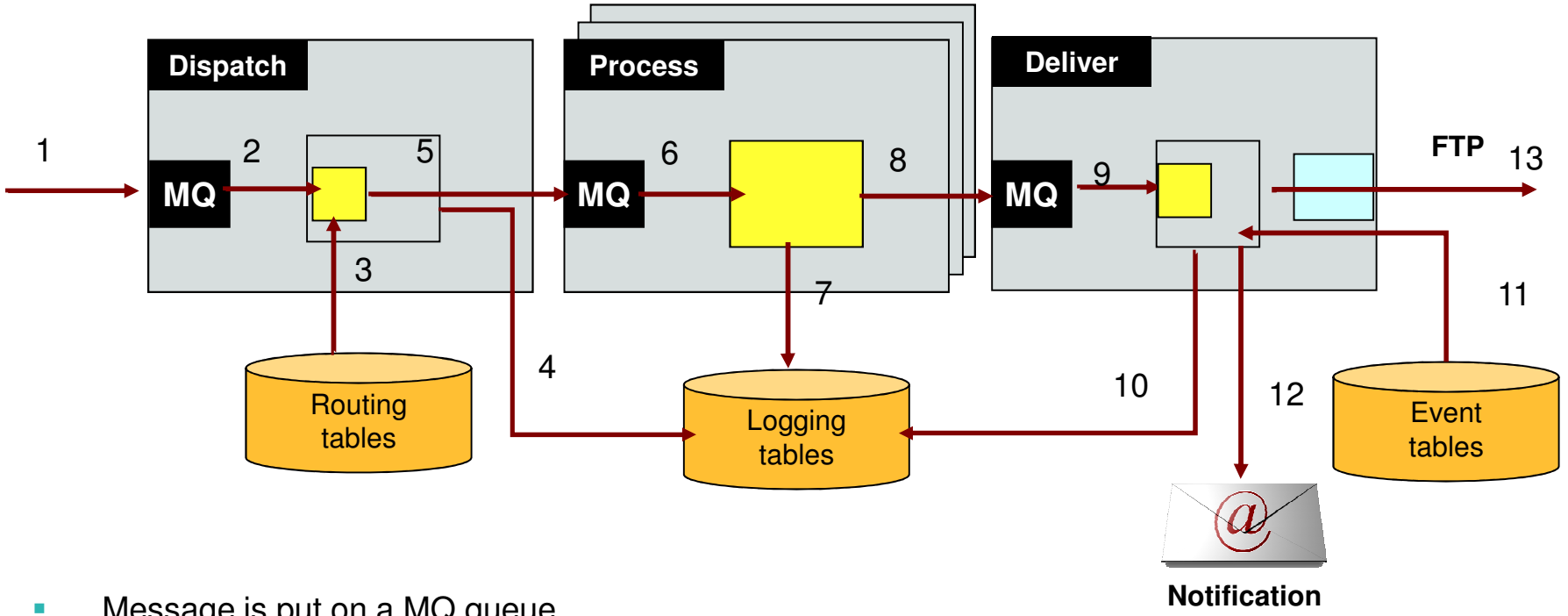


Sub Flow for Delivery

Service: Delivery (II)

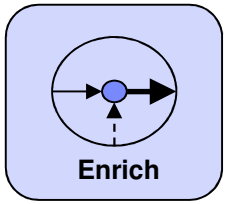


Example 1: Send MQ Message to File Output

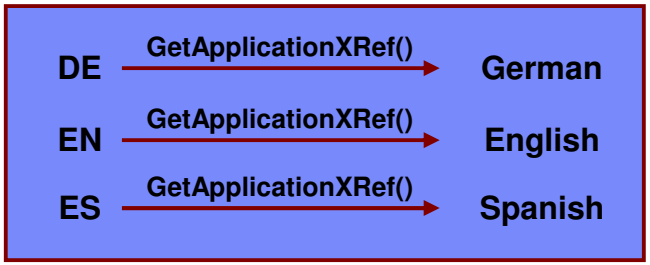


- Message is put on a MQ queue
- Message is delivered into a file system
- Message is logged three times

Component: Value Mapping

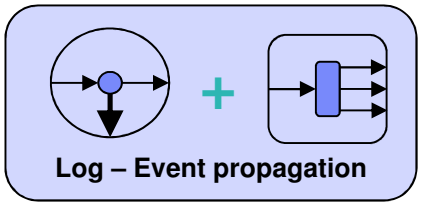


- Translates codes like language codes, partner or material numbers from external to internal representation
- Simple key-value pairs
- Is called by processing services
- Data is stored in a database and maintained via the Administration GUI
- Access to value mapping data via encapsulated procedures

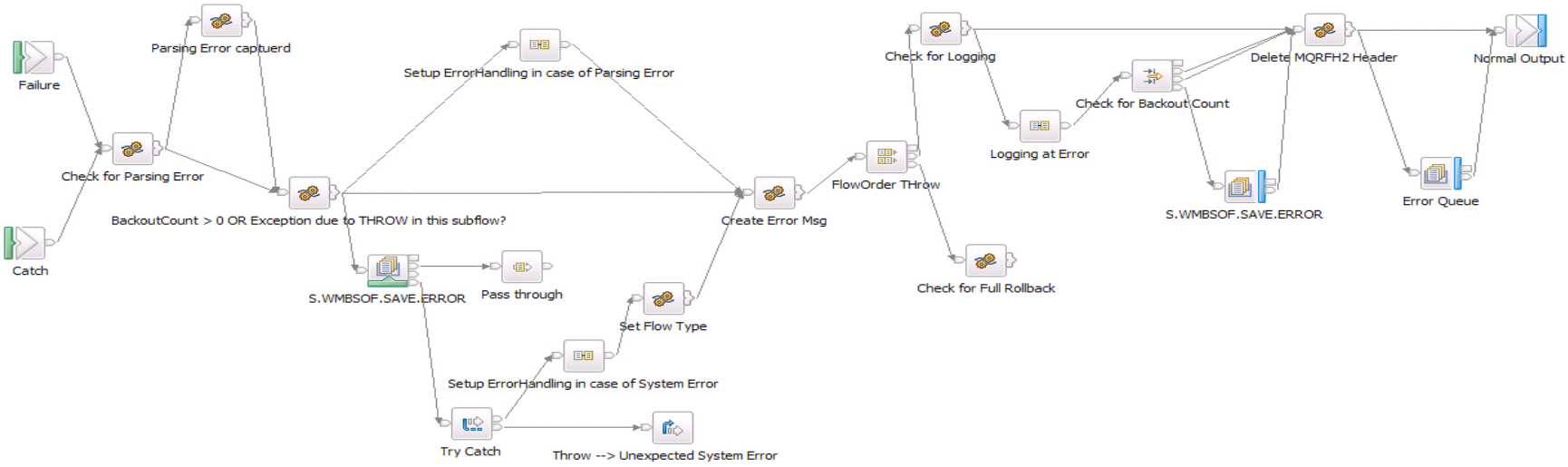


Value Map

Component: Errorhandling

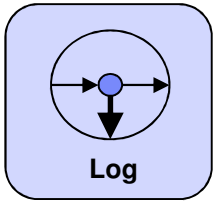


- Logs erroneous message (payload), its metadata as well as error reason for possible restart as XML in the log-event database (XML datatype)
- Receives XML from dispatching, processing or delivery service
- Is part of every MessageFlow as a subflow
- Triggers notification service for registered users (Email)
- Uses logging service

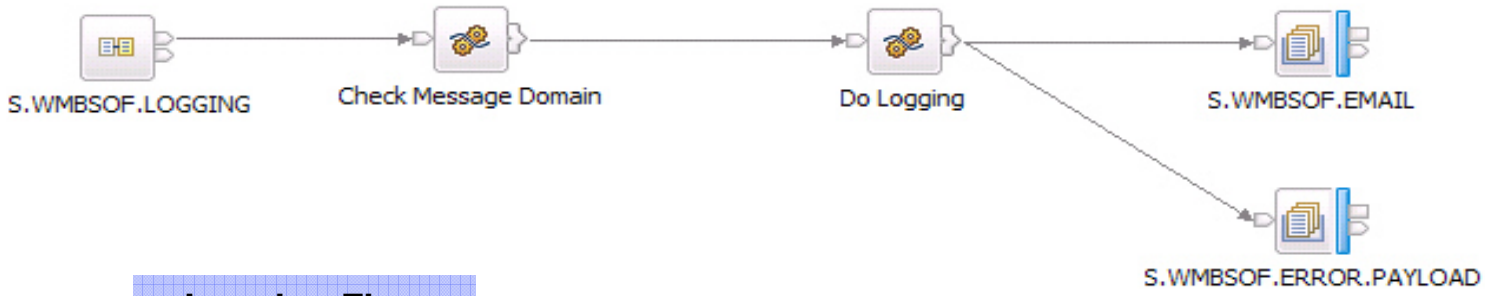


Errorhandling Sub Flow

Service: Logging

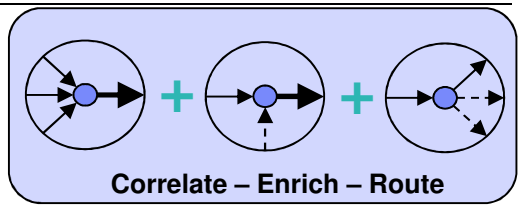


- Responsible for storing messages in a database for analysis, monitoring, tracking, restart or correlation with messages at a later point in time
- Receives XML
- MQ input
- Uses common errorhandling component and notification service
- Logged data is accessible through GUI
- Store XML content in log-event database

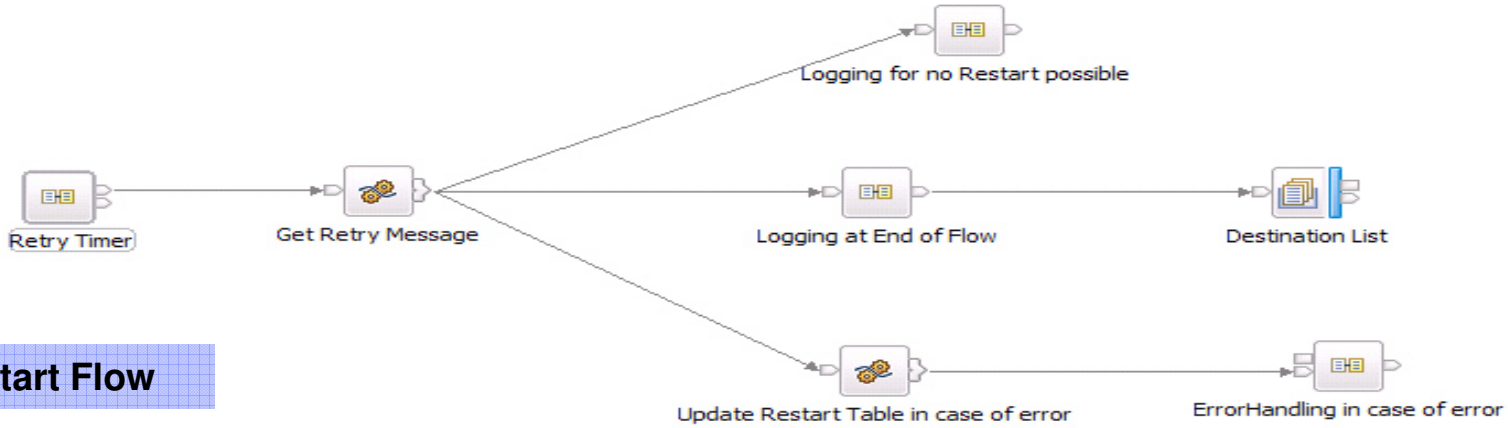


Logging Flow

Service: Restart

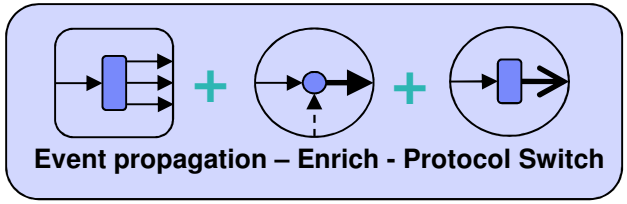


- Looks for erroneous messages in the log-event database and according to the configuration tries to restart message
- If required runs one or more restarts by putting payload into processing service again
- Stops resending after n times. N is a parameter in the database for this specific sender – processing – receiver combination
- Restart is only available for services which have a MQInput node



Restart Flow

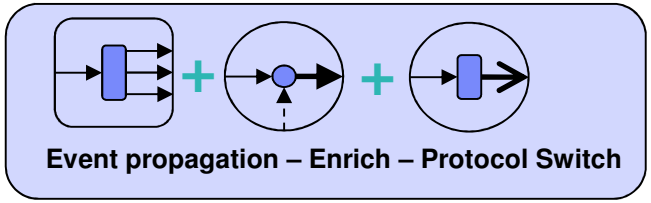
Service: Notification



- Send Email in case of error or successful processing to registered users
- Receives XML
- MQ input
- Uses configurable templates for Email
- Uses different event types for sending notifications
- Templates are defined via Administration GUI

Notification Flow

Service: SAP-Status

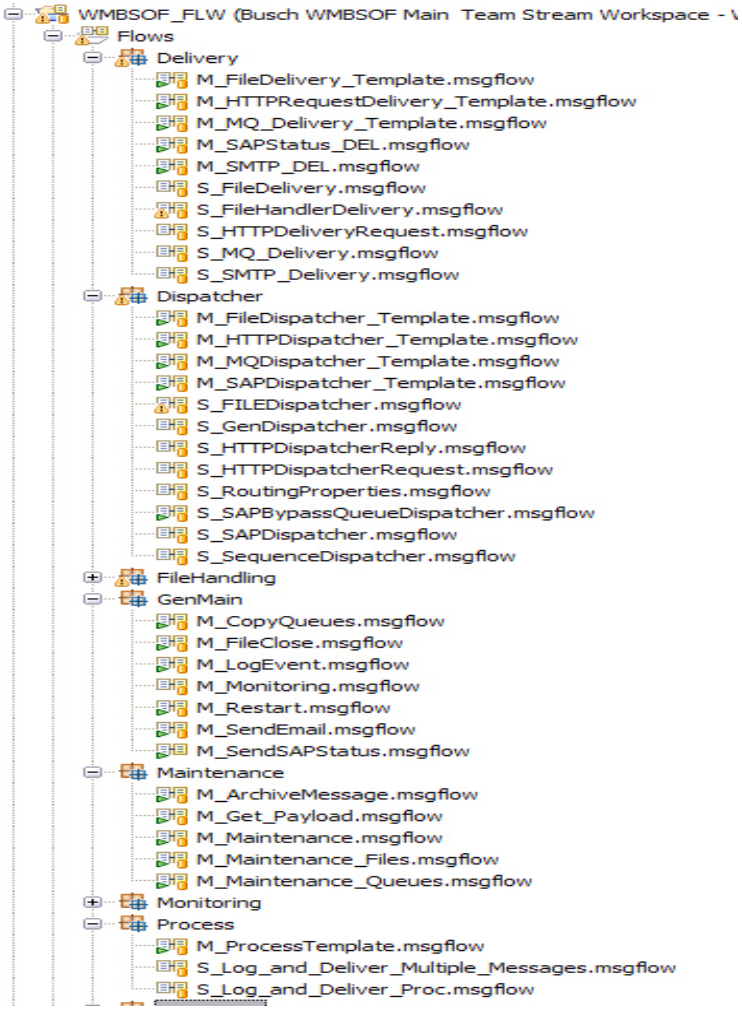


- Send SAP Status for incoming SAP IDOC back to the SAP System
- SAP Status is sent for Process and Delivery Flows
- SAP Status is sent as a BAPI

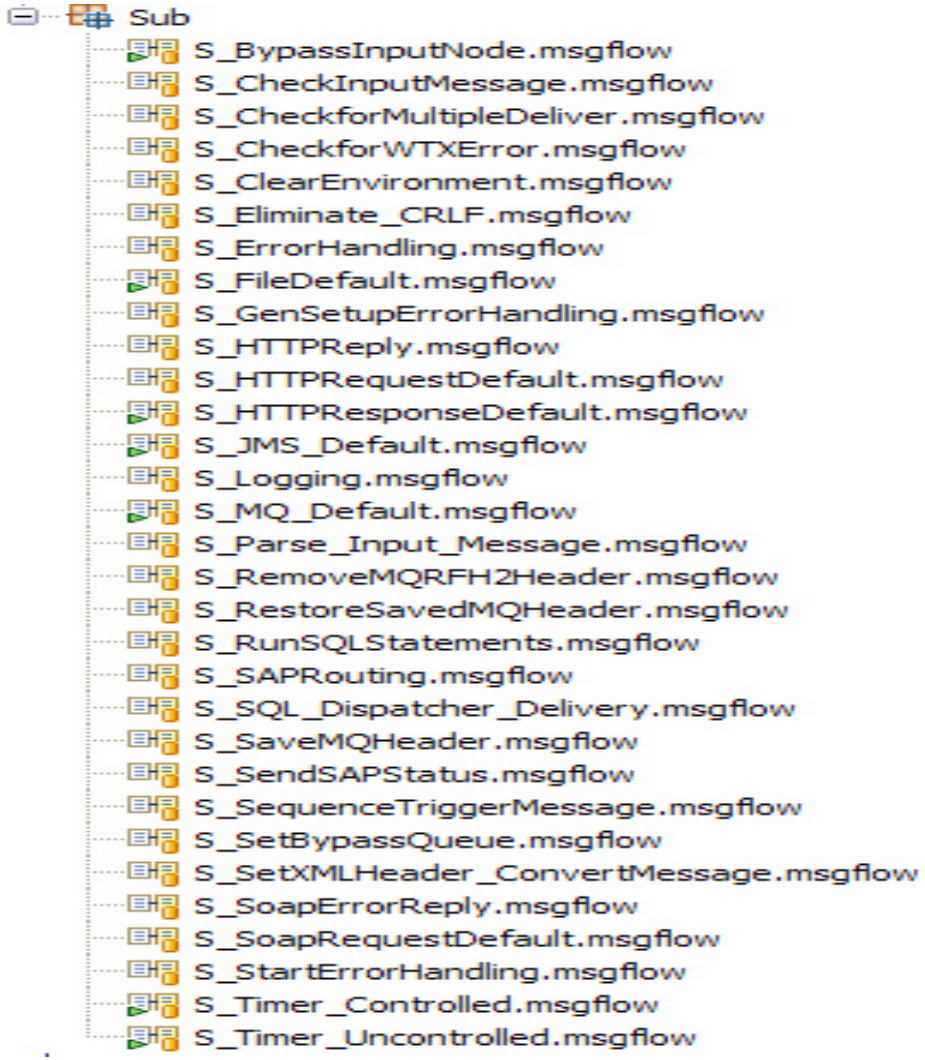
Flow for sending SAP Status

Toolbox (1)

- Template flows are provided for:
  - File Delivery
  - HTTP/s Delivery
  - MQ Delivery
  - Processing
  - File Dispatching
  - HTTP/S Dispatching
  - MQ Dispatching
  - SQL Dispatching
  - WTX for File Dispatching
  - WTX for MQ Dispatching
- Set of subflows: Reduced development time
- Developer can use templates and subflows via drag & drop for flow development
- Efficient code reuse



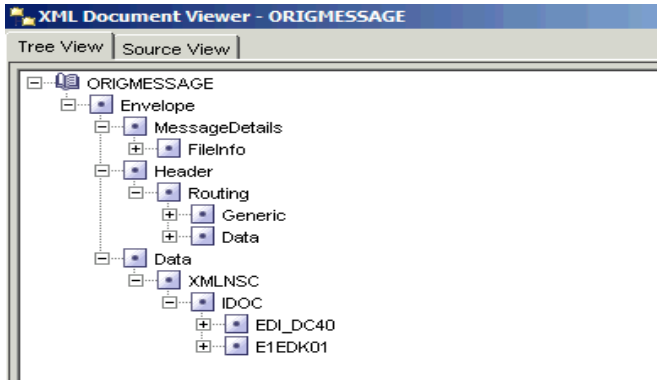
Toolbox (2)



Framework Database (I)

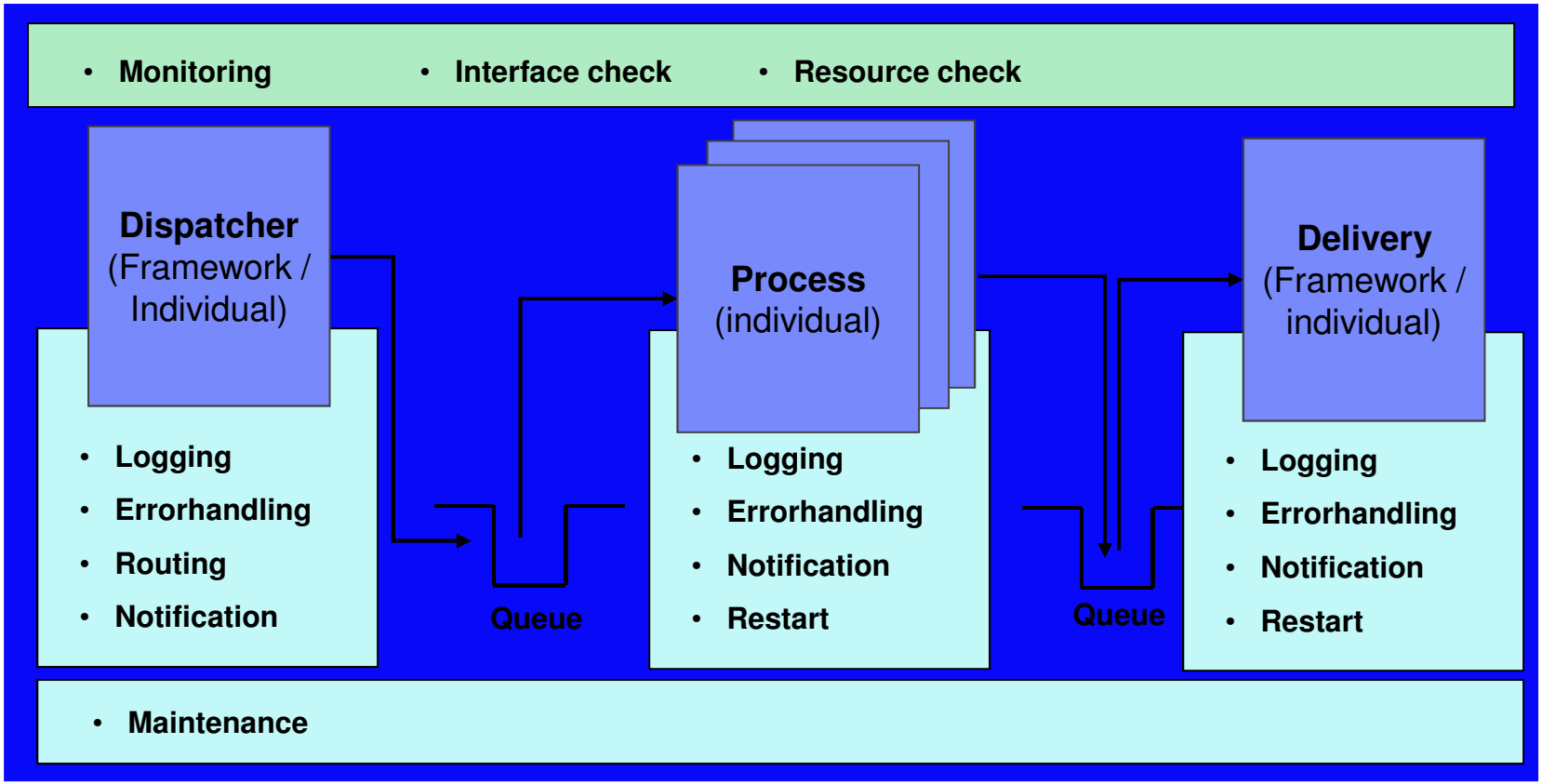
Information Management software

- One of the most significant new features in IBM DB2 9.5 is the XML functionality (pureXML™), which means that XML data is stored and queried in its inherent hierarchical format
- The Framework uses the DB 9.5 XML feature to store the payload information for later query with XPATH
- Database contains
  - Tables to get routing information
  - Tables to log the incoming events and payload information
  - Tables to maintain notification events
  - Table for maintenance



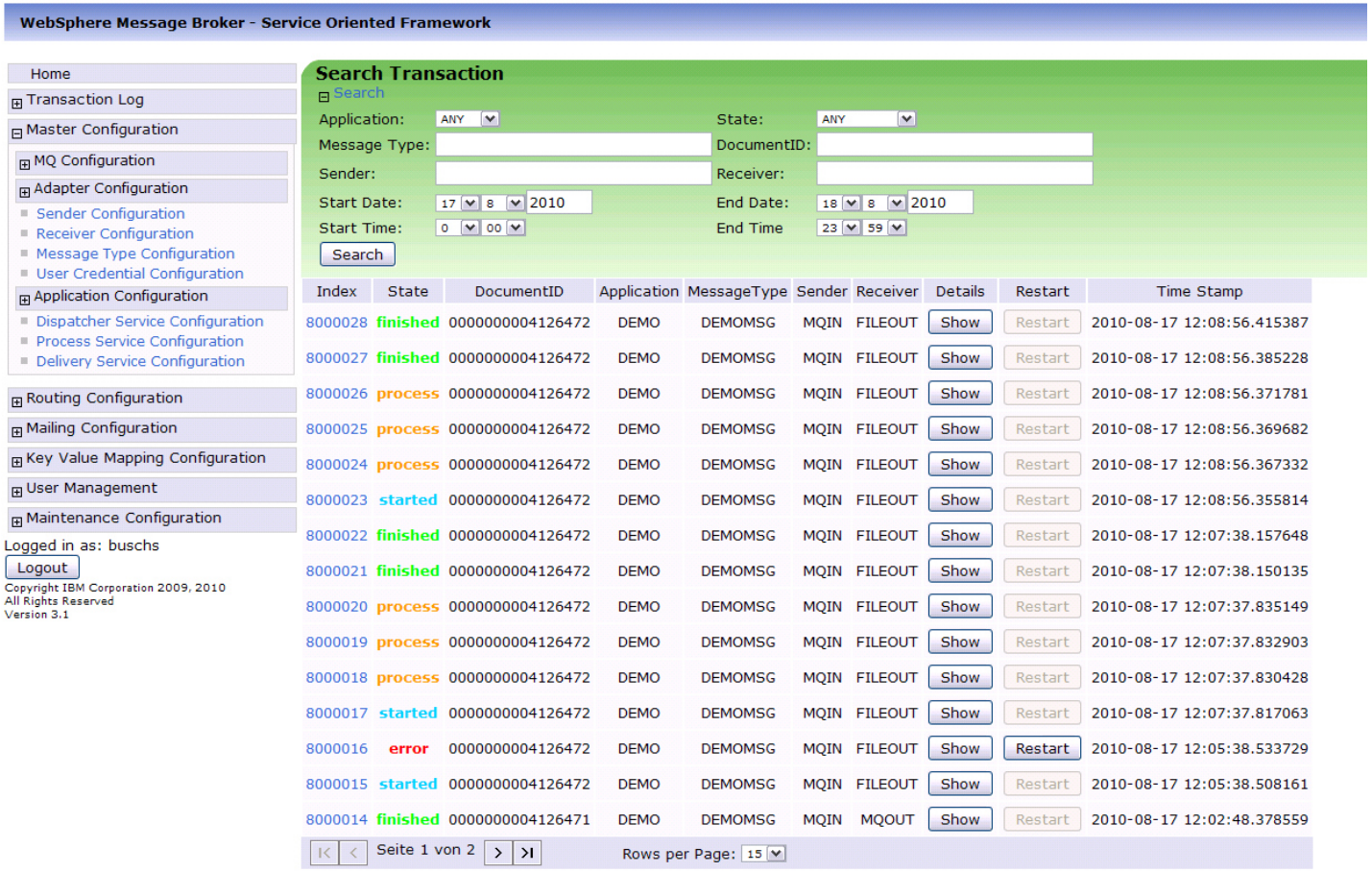
High level view of a generic framework application

ESB Framework part **GUI / Framework console part**



Administration GUI

- Monitoring
- Administration
- Configuration
- Single point of information



Agenda

1 Motivation and preconditions

2 SOA Reference Architecture /ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

6 Lessons learned

7 Outlook

8 Questions

Demo

- Case 1: Create a complete framework end-to-end workflow
  - Create Dispatcher Service: Http Dispatcher
  - Use a preconfigured Process Service
  - Create Delivery Service: File Delivery
  - Configure application with GUI
- Case 2: Switch Dispatcher and Delivery service
  - Switch Dispatcher: Http -> MQ
  - Switch Delivery: File -> MQ

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

6 Lessons learned

7 Outlook

8 Questions

Lessons learned

- Development of framework created additional effort but upcoming projects will be developed faster
- Rapid flow development
- Easy to extend
- Little complexity of flows
- Even developers with little WMB and WMQ skill can use the framework
- Single point of configuration
- Framework is designed for „most common“ WMB use cases
- It is running on a customer **production** system

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

6 Lessons learned

7 Outlook

8 Questions

Outlook

- Use WebSphere Business Monitor for monitoring
- Reuse this framework in future customer projects
- Extend framework to use multiple sources / destinations
  → currently only point-to-point connection
- Integrate upcoming WMB features
  - Interested? Please contact us

Agenda

1 Motivation and preconditions

2 SOA Reference Architecture / ESB related patterns

3 Why a Framework?!

4 Framework architecture

5 Demo

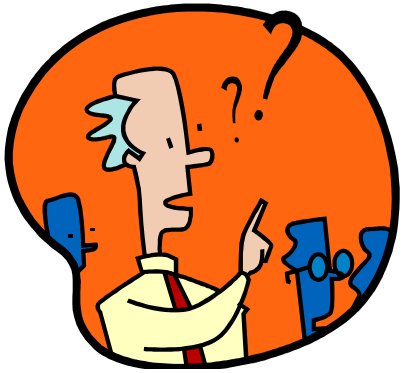
6 Lessons learned

7 Outlook

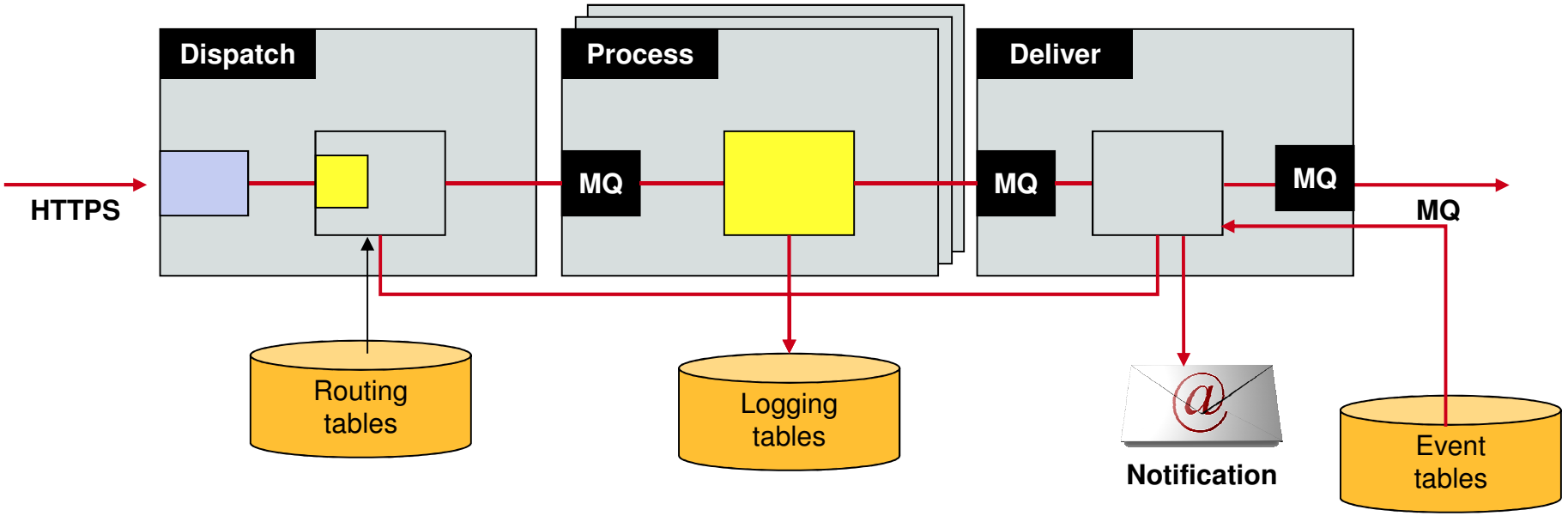
8 Questions

Questions

- Thank you!!

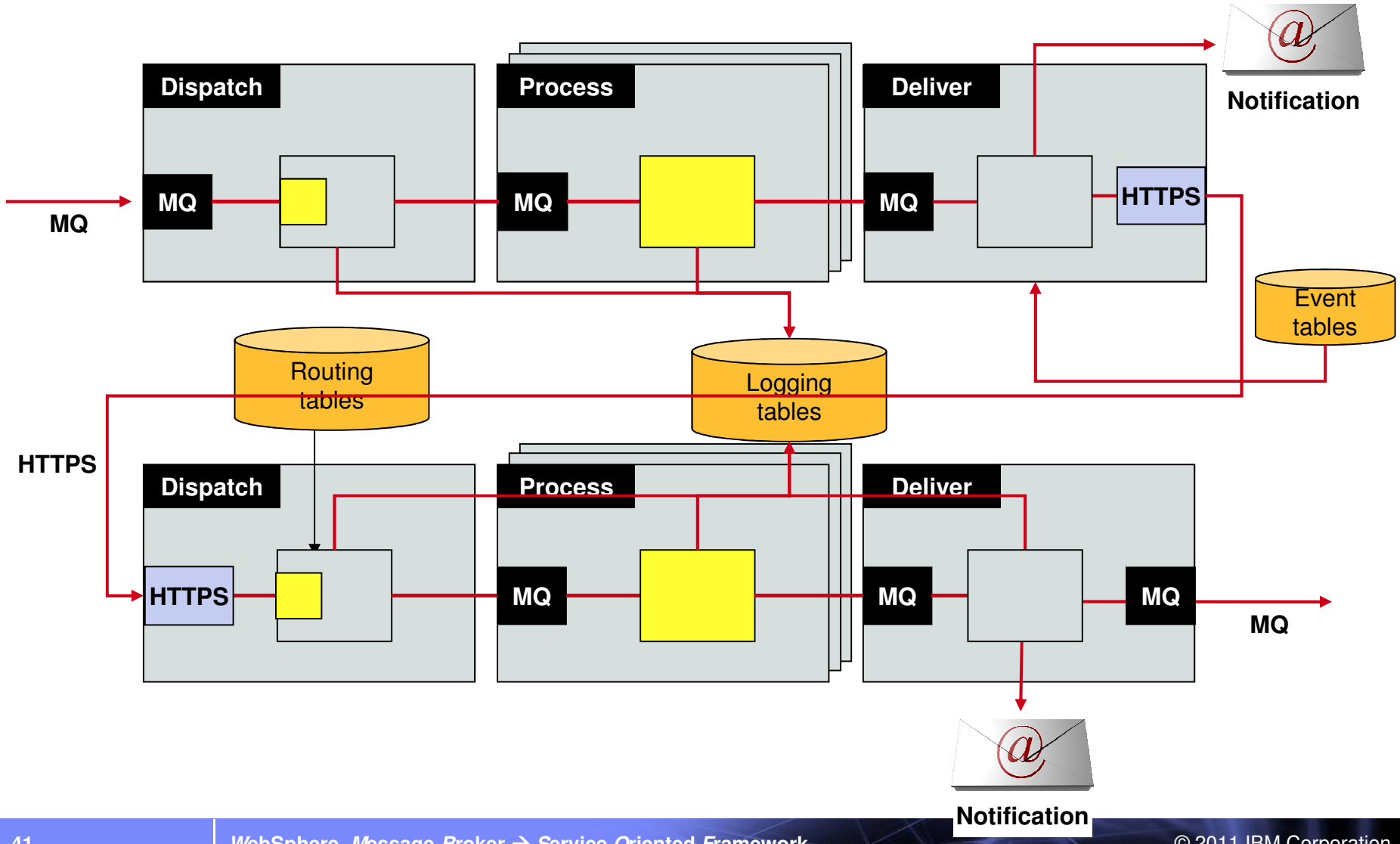


Example 2: Send HTTPS Message to MQ Output

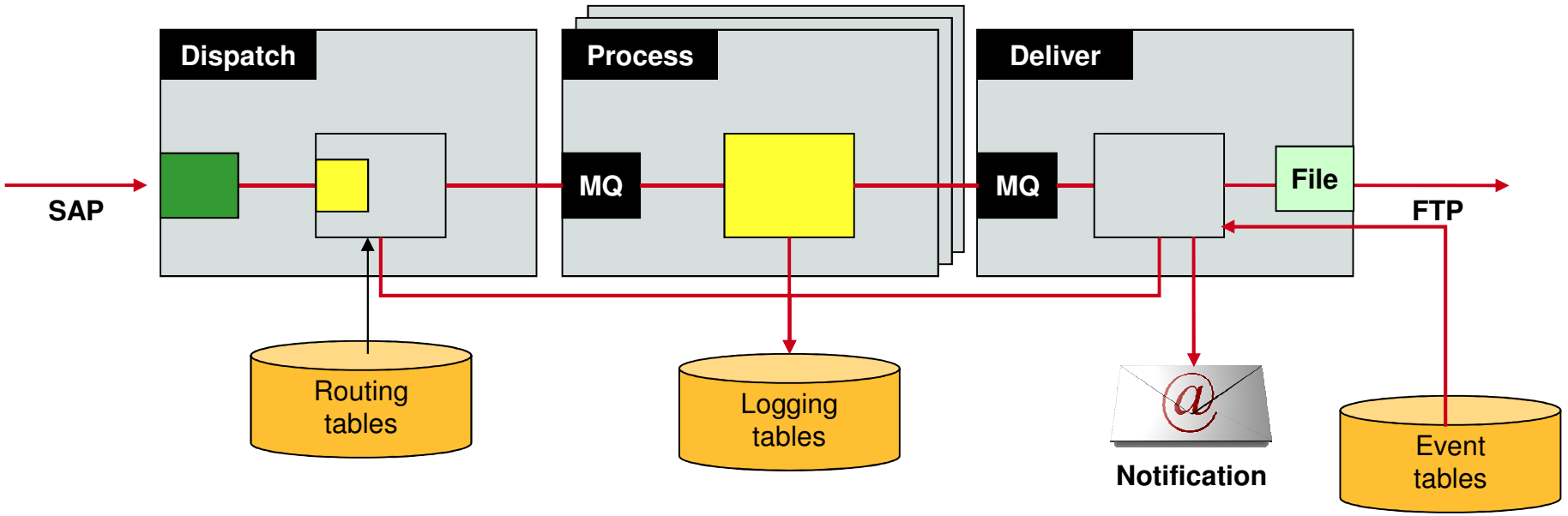


- Message is sent as HTTPS request
- Message is delivered to a MQ queue
- Message is logged three times

Example 3: Concatenation of two scenarios



Example 4: Send SAP Message to File Output



- Message is sent as IDoc from SAP
- Message is delivered to file system
- Message is logged three times (each process step)

Addional Ressources

- IBM Systems Journal Vol. 44, No. 4, 2005 - Service-Oriented Architecture
  http://researchweb.watson.ibm.com/journal/sj44-4.html
- IBM Patterns for e-business library
  http://www.ibm.com/developerworks/patterns/library/index.html
- IBM WebSphere Message Broker product information
  http://www-01.ibm.com/software/integration/wbimessagebroker/
- IBM WebSphere Message Broker Information Center
  http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp
- IBM WebSphere Message Broker Library
  http://www-01.ibm.com/software/integration/wbimessagebroker/library/
- IBM DB2 9 Product Information
  http://www-01.ibm.com/software/data/db2/9/