WSO2 API Manager 4.2.0 Advanced - Integration Profile

Advanced Concepts

WSO2 Training

CC by 4.0

Module Outline

- Product configurations
- High-level messaging architecture
- Passthrough Transport
- RabbitMQ Transport
- JMS Transport
- Message Builders
- Message Formatters
- Message Relay

Product Configurations

- TOML-based product configurations.
- A single **deployment.toml** file to apply the most critical product configurations.
- See the complete list of available product configurations.

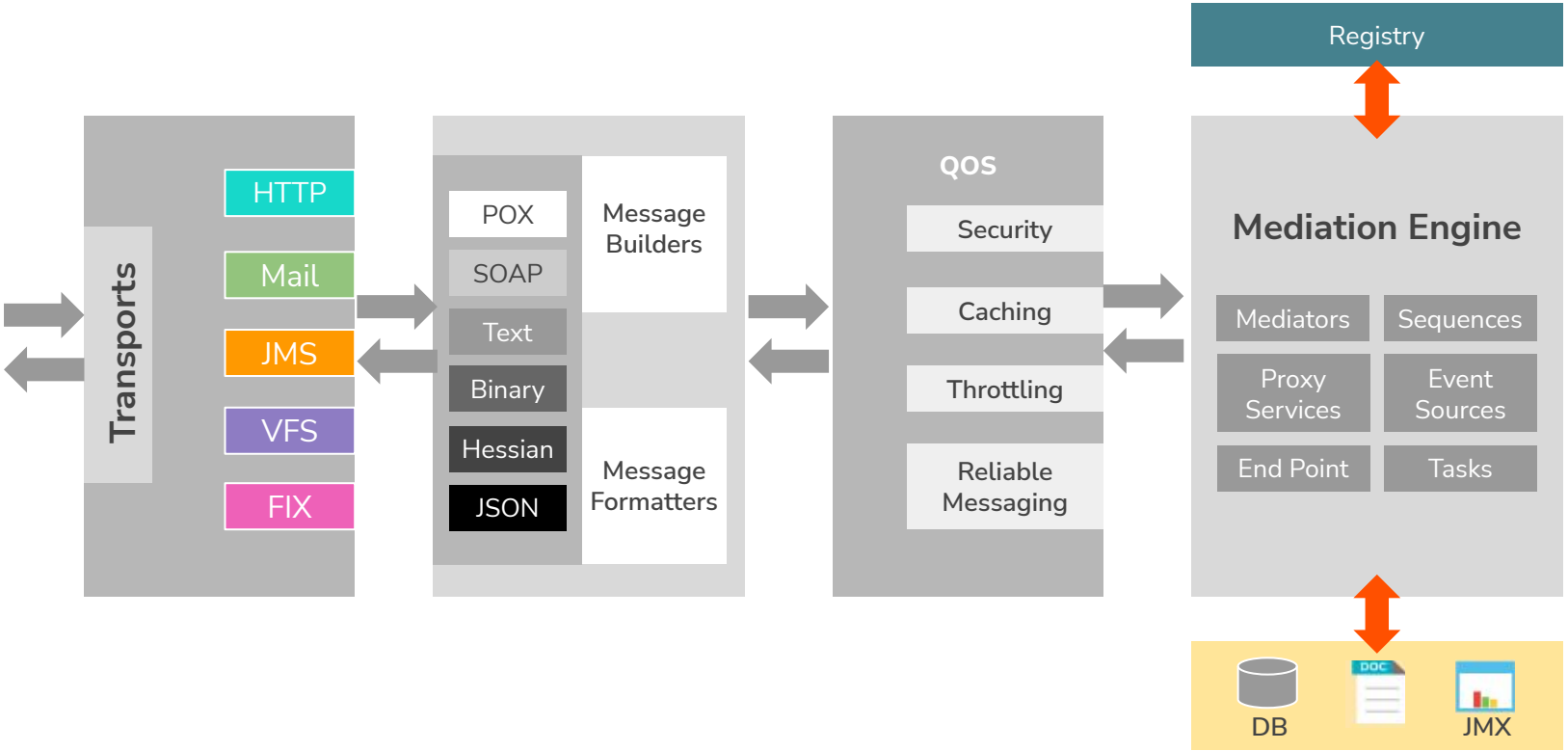
```
[server]
hostname = "localhost"

[keystore.tls]  # IMPORTANT! Be sure to change this heading to [keystore.primary] when you use the product.
file_name = "wso2carbon.jks"
password = "wso2carbon"
alias = "wso2carbon"
key_password = "wso2carbon"

[truststore]
file_name = "client-truststore.jks"
password = "wso2carbon"
alias = "symmetric.key.value"
algorithm = "AES"

## Following are set of example configs. Please refer docs for complete set of configurations.

[transport.http]
socket_timeout = 180000
disable_connection_keepalive = false
connection_timeout = 90000
```

Product Configurations

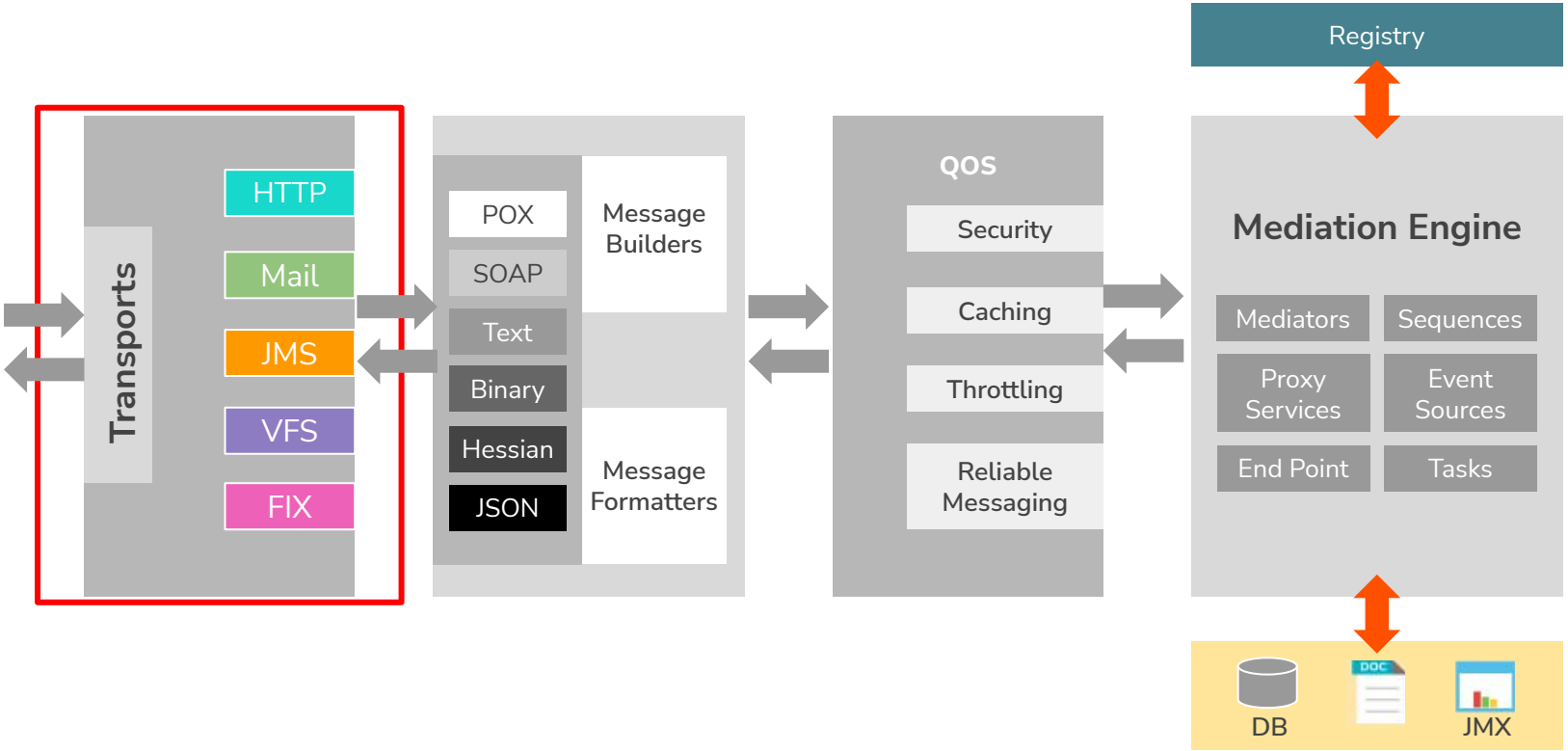
Benefits of TOML-based config model:

- Focus on configurations and not product components.
- Minimize human errors during product configurations.
- Certain mandatory configurations inferred without user intervention.
- Consistency in configuration grouping.

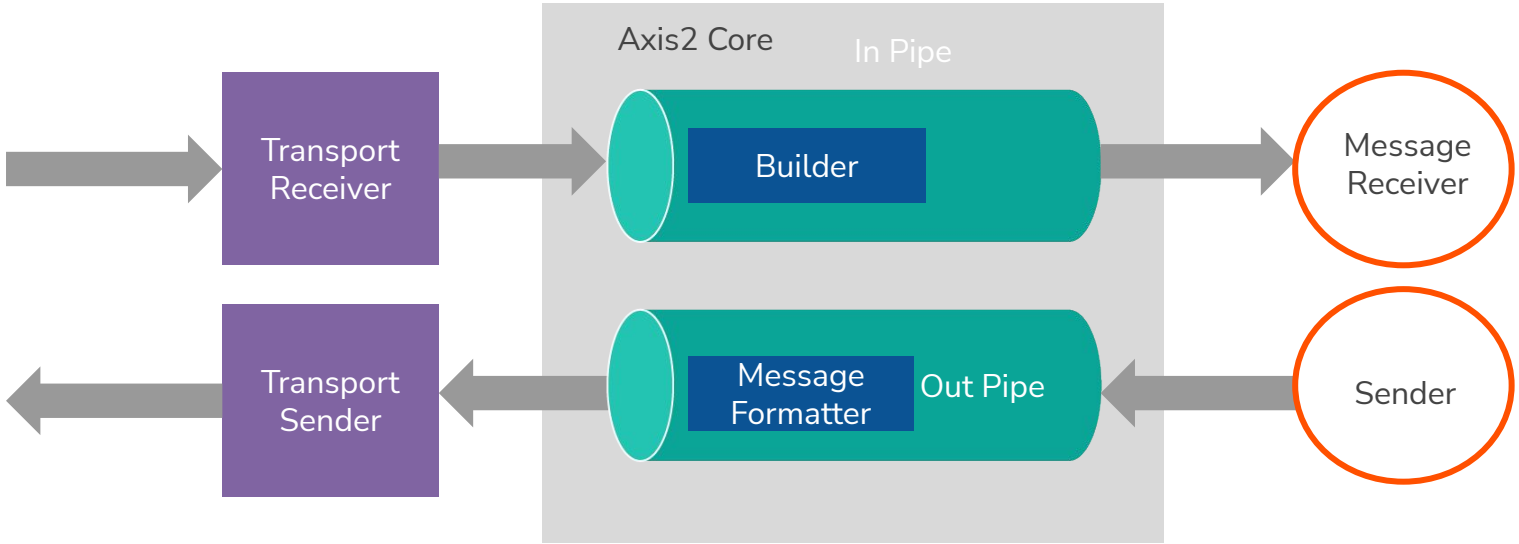
High-Level Architecture



Transports


Registry
Transports
HTTP
Mail
JMS
VFS
FIX
POX
SOAP
Text
Binary
Hessian
JSON
Message Builders
Message Formatters
QOS
Security
Caching
Throttling
Reliable Messaging
Mediation Engine
Mediators
Sequences
Proxy Services
Event Sources
End Point
Tasks
DB
JMX

Transports


Axis2 Core
In Pipe
Transport Receiver
Builder
Message Receiver
Transport Sender
Message Formatter
Out Pipe
Sender

Configuring Transports

- All transports are configured by default.
- Change default configurations using the *deployment.toml* file.

Example:
VFS Transport

```
[transport.vfs]
listener.enable = true
listener.keystore.file_name = "$ref{keystore.tls.file_name}"
listener.keystore.type = "$ref{keystore.tls.type}"
listener.keystore.password = "$ref{keystore.tls.password}"
listener.keystore.key_password = "$ref{keystore.tls.key_password}"
listener.keystore.alias = "$ref{keystore.tls.alias}"
listener.parameter.customParameter = ""
sender.enable = true
sender.parameter.customParameter = ""
```

- Defined as Sender-Receiver pairs.
- Loaded during server startup.

Transport Receiver/Sender

Transport Receiver implementation

- Implementations of this interface should specify how incoming messages are received and processed before handing them over to the Axis2 engine for further processing.
- Editing the *deployment.toml* file should be followed by a server restart.

Transport Sender implementation

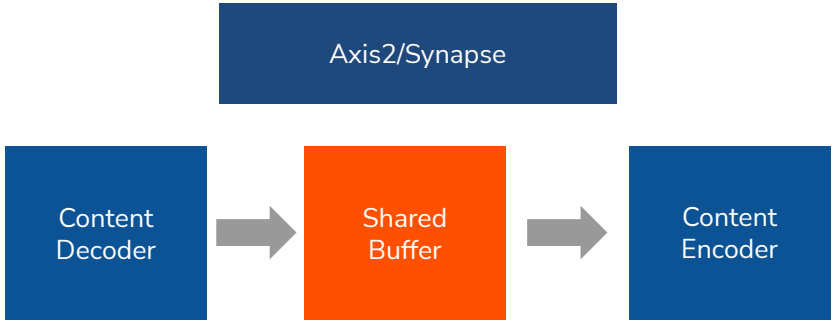
- Implementations of this interface should specify how a message can be sent out from the Axis2 engine.

Non-Blocking Architecture

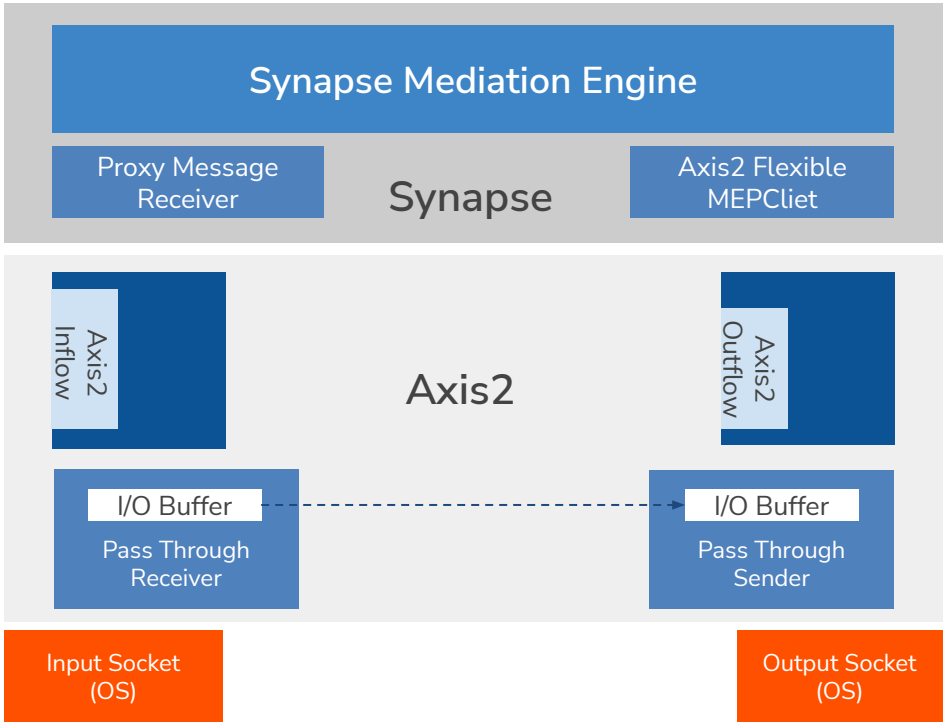
- Non-blocking mediation engine.
- Non-blocking IO handling with JAVA NIO Technology (HTTPCore NIO Library).

Passthrough Transport

- Based on a single buffer model.
- Completely bypasses the Axiom layer.
- On-demand message parsing in the mediation engine.
- The default HTTP/HTTPS transport.

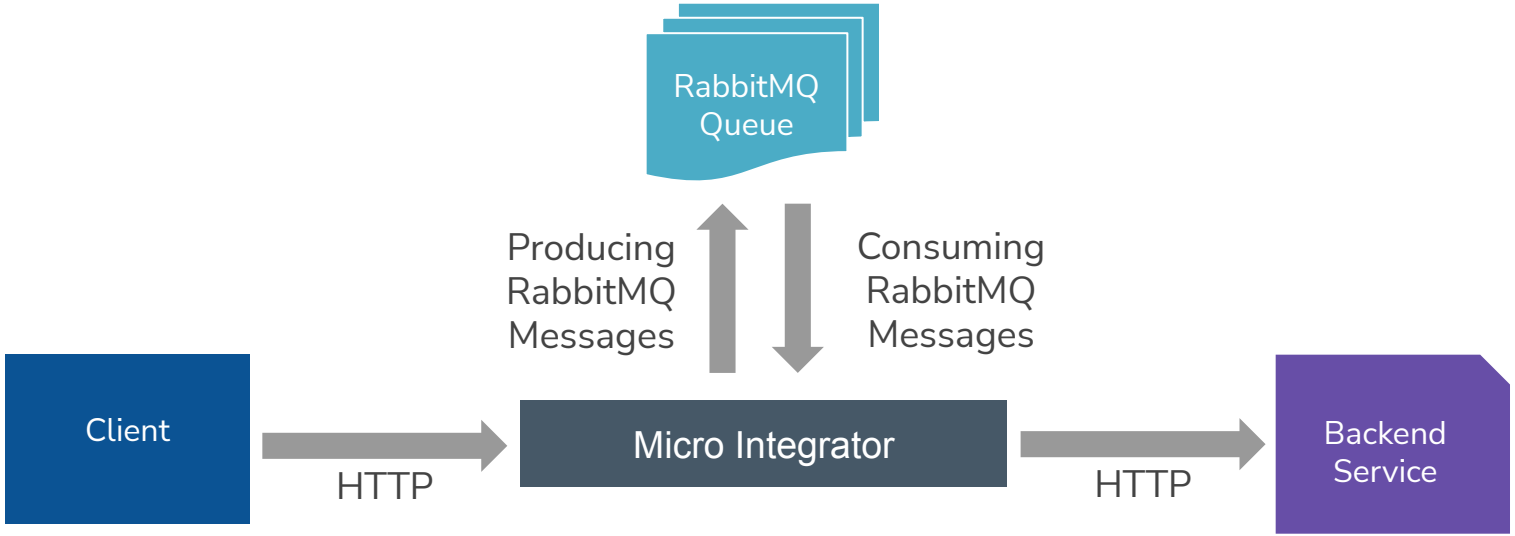


Passthrough Transport



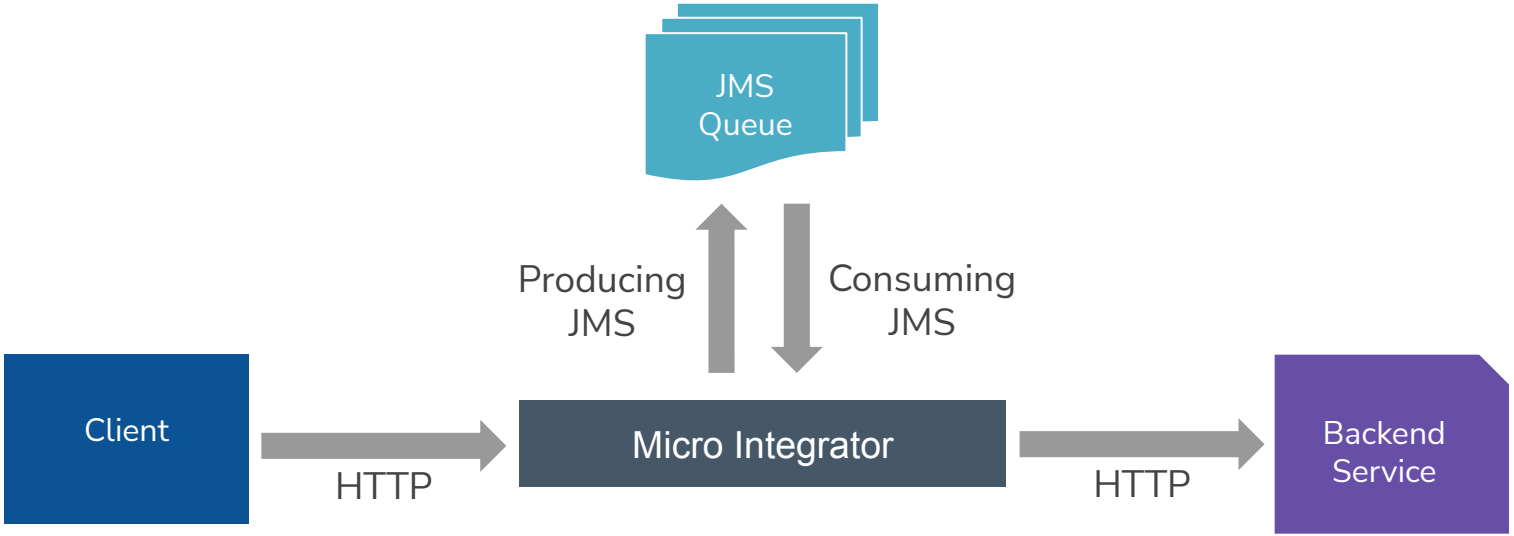
RabbitMQ Transport

WSO2 Micro Integrator as a RabbitMQ Consumer and Producer.

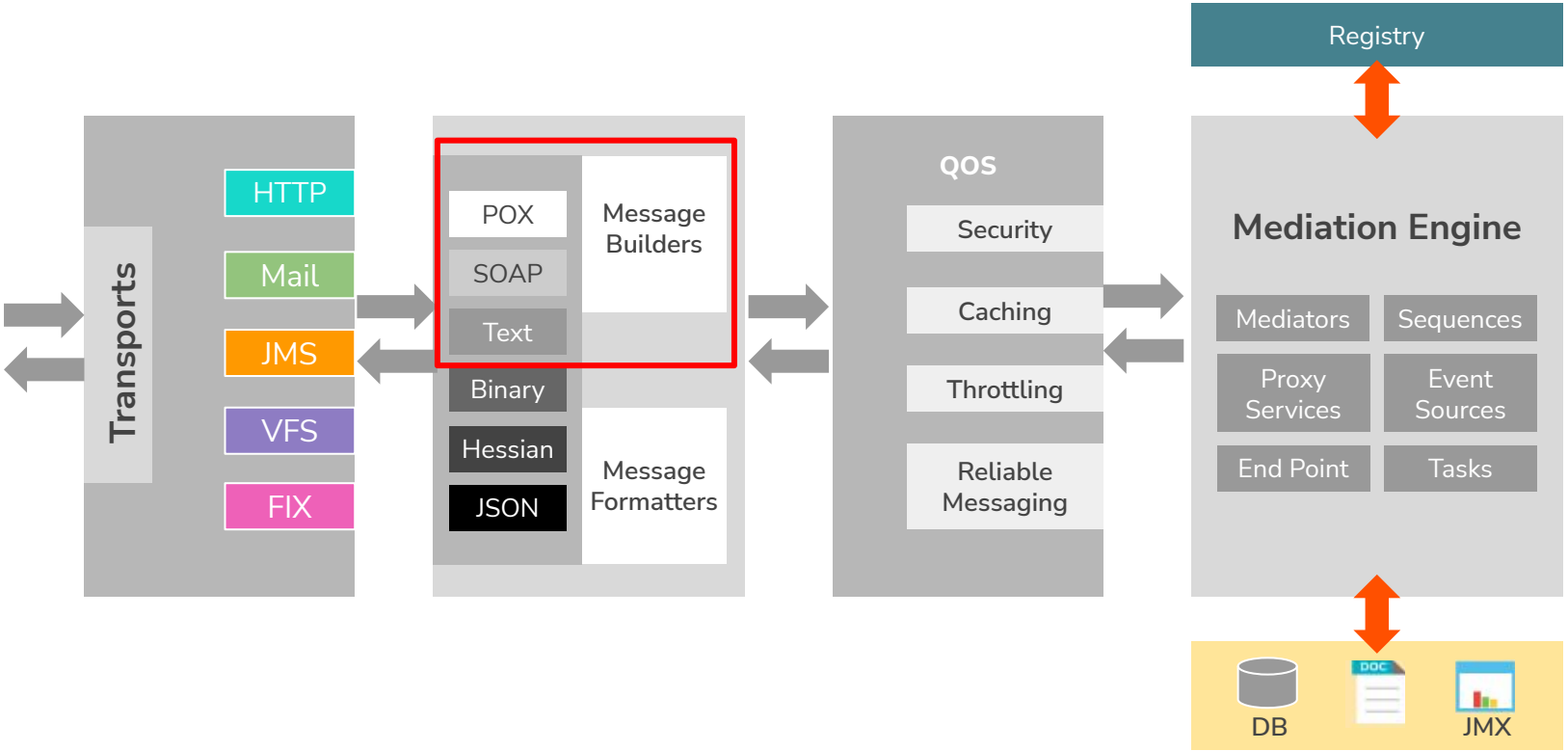


JMS Transport

WSO2 Micro Integrator as a JMS Consumer and JMS Producer.



Message Builders



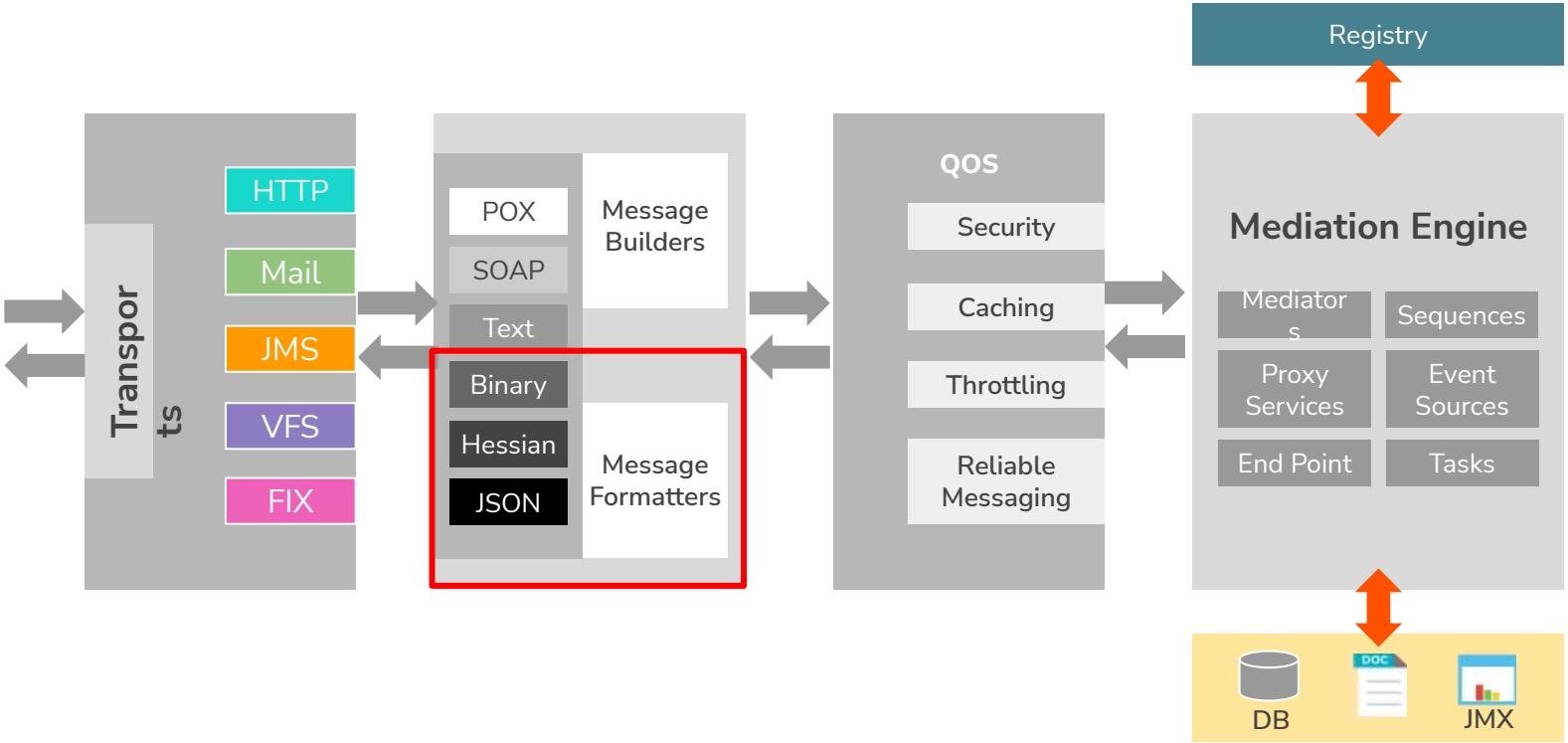
Message Builders

- Convert incoming messages to SOAP.
- Reads incoming messages based on the message content type.
- Deserializes the messages.

Default Message Builders

```
application_xml = "org.apache.axis2.builder.ApplicationXMLBuilder"
form_urlencoded = "org.apache.synapse.commons.builders.XFormURLEncodedBuilder"
multipart_form_data = "org.apache.axis2.builder.MultipartFormDataBuilder"
text_plain = "org.apache.axis2.format.PlainTextBuilder"
application_json = "org.wso2.micro.integrator.core.json.JsonStreamBuilder"
json_badgerfish = "org.apache.axis2.json.JSONBadgerfishOMBuilder"
text_javascript = "org.apache.axis2.json.JSONBuilder"
octet_stream = "org.wso2.carbon.relay.BinaryRelayBuilder"
application_binary = "org.apache.axis2.format.BinaryBuilder"
```

Message Formatters



Message Formatters

- Converts AXIOM messages to on-the-wire messages.
- Converts based on the content type of the outgoing message.
- Serializes the message.

Default Message Formatters

```
form_urlencoded = "org.apache.synapse.commons.formatters.XFormURLEncodedFormatter"
multipart_form_data = "org.apache.axis2.transport.http.MultipartFormDataFormatter"
application_xml = "org.apache.axis2.transport.http.ApplicationXMLFormatter"
text_xml = "org.apache.axis2.transport.http.SOAPMessageFormatter"
soap_xml = "org.apache.axis2.transport.http.SOAPMessageFormatter"
text_plain = "org.apache.axis2.format.PlainTextFormatter"
application_json = "org.wso2.micro.integrator.core.json.JsonStreamFormatter"
json_badgerfish = "org.apache.axis2.json.JSONBadgerfishMessageFormatter"
text_javascript = "org.apache.axis2.json.JSONMessageFormatter"
octet_stream = "org.wso2.carbon.relay.ExpandingMessageFormatter"
application_binary = "org.apache.axis2.format.BinaryFormatter"
```

Message Relay

- Enables the Micro Integrator to pass messages without building or processing unless specifically requested.
- Replace the message builders and formatters with the following class implementations:
  - Binary relay builder

    ```
    org.wso2.carbon.relay.BinaryRelayBuilder
    ```

  - Binary relay formatter

    ```
    org.wso2.carbon.relay.ExpandingMessageFormatter
    ```

Let’s try it out!

Micro Integrator as a JMS Consumer and Producer