

Integration between services and mediation

Oxford University
Software Engineering
Programme
April 2021



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



Through 2020, integration work will account for 50% of the time and cost of building a digital platform.

Use a Hybrid Integration Approach for Digital Transformation

© 2018 Gartner, Inc. and/or its affiliates. All rights reserved.

Smarter With Gartner®



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Seven Cases of Integration (from Gartner)

API Management

API

Application Integratio

SAP

B2B Integration



Data Integration



Digital Integration Hub

APP
API

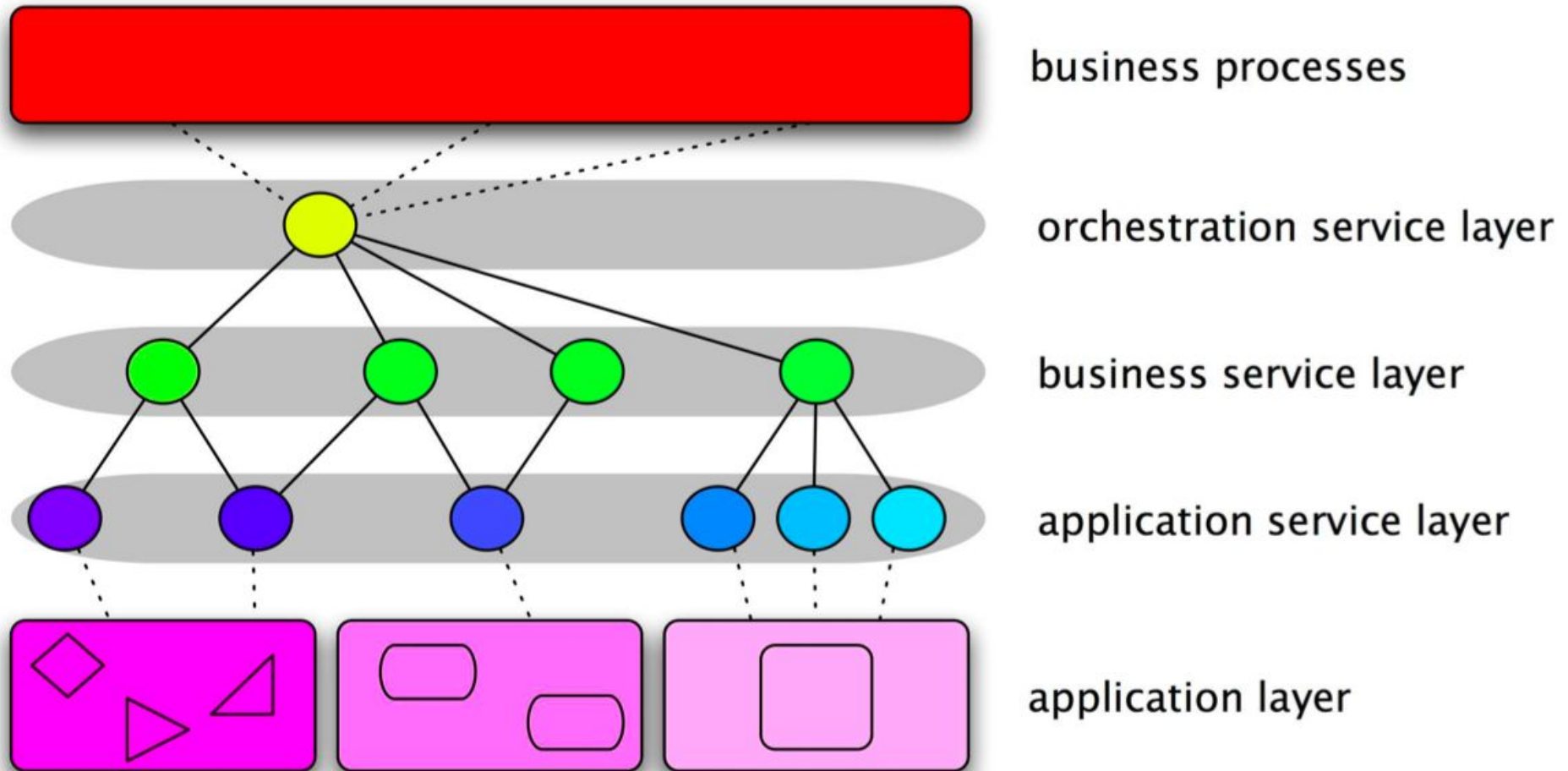
Event Streaming



IOT Integration



Recap on SOA model

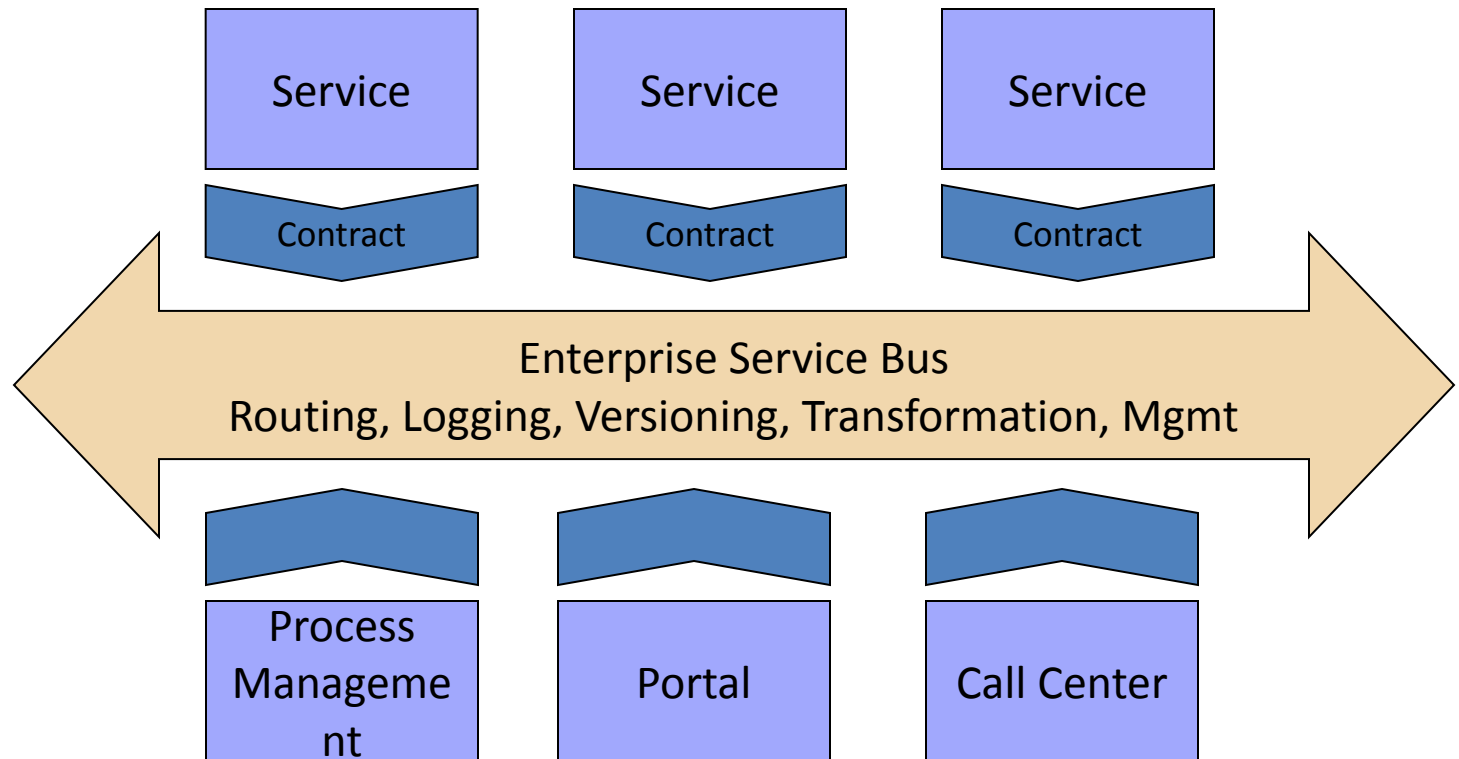


Enterprise Service Bus (ESB)

- A software architecture
 - A logical intermediary through which every message flows
 - Offers a policy based approach to decide what to do to each message or interaction
- The benefits of the gateway model
 - Without a physical hub and spoke
- Many vendors offer ESB products
 - Often a layer over an existing messaging framework



ESB as an implementation of SOA



Enterprise Integration Patterns

Home - Enterprise Integratio x


www.eaipatterns.com

Offline Mail Inbox (124,820) - p 100+ WSO2 WSO2, Inc. - Calend + bitmark Shorten with bit.ly

**Enterprise
Integration
Patterns**

Home

[HOME](#) • [PATTERNS](#) • [RAMBLINGS](#) • [ARTICLES](#) • [TALKS](#) • [DOWNLOAD](#) • [LINKS](#) • [BOOKS](#) • [CONTACT](#)


 **Ramblings**

My ongoing thoughts about the present and future of integration, SOA and Web services. [\[see all\]](#)

[DDD - Diagram Driven Design](#)
(March 22, 2010)

[What Does It Mean to Use Messaging?](#)
(Feb 17, 2010)

[A Chapter a Day...](#)
(Feb 1, 2010)

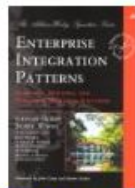
 **Upcoming Events**

Articles & Interviews

[Conversations Between Loosely Coupled Services](#)
(Video on [InfoQ](#))

[Developing in a Service-oriented World](#)
(Video on [InfoQ](#))

[SOA Patterns - New Insights or Recycled Knowledge?](#)
(Whitepaper)



[Enterprise Integration Patterns](#)

Gregor Hohpe, Bobby...

Best Price \$23.99 or Buy New \$50.74

Patterns and Best Practices for Enterprise Integration

This site is dedicated to making the design and implementation of integration solutions easier. The solutions and approaches described here are relevant for integration tools and platforms such as IBM WebSphere MQ, TIBCO, Vitria, SeeBeyond, WebMethods, or BizTalk, messaging systems such as JMS, WCF, or MSMQ, ESB's such as Sonic, Fiorano, ServiceMix, Mule, Apache Synapse, or WSO2, and SOA and Web-service based solutions.

All content on this site is original and is maintained by [Gregor Hohpe](#). I have been building integration solutions for large clients for many years and enjoy sharing my findings with the community. I hope you find this material insightful and useful. Please [contact me](#) if you have suggestions or feedback.

Enterprise Integration Patterns - The Book

Enterprise integration remains harder than it really should be. While integration is inherently complex, I felt that one of the major stumbling blocks is the lack of a common vocabulary and body of knowledge around asynchronous messaging architectures used to build integration solutions. Under the guidance of Martin Fowler and Kyle Brown, I teamed up with Bobby Woolf to create such a language in the form of 65 [integration patterns](#) (see the pattern links on the right).

The book [Enterprise Integration Patterns](#) provides a consistent vocabulary and visual notation to describe large

Integration Patterns

[Integration Patterns Overview](#)

[Table of Contents](#)

[Revision History](#)

Introduction

[Preface](#)

[Introduction](#)

[Solving Integration Problems using Patterns](#)

Integration Styles

[Introduction](#)

[File Transfer](#)

[Shared Database](#)

[Remote Procedure Invocation](#)

[Messaging](#)

Messaging Systems

[Introduction](#)

[Message Channel](#)

[Message](#)

[Pipes and Filters](#)

[Message Router](#)


[Message Translator](#)

[Message Endpoint](#)

Messaging Channels

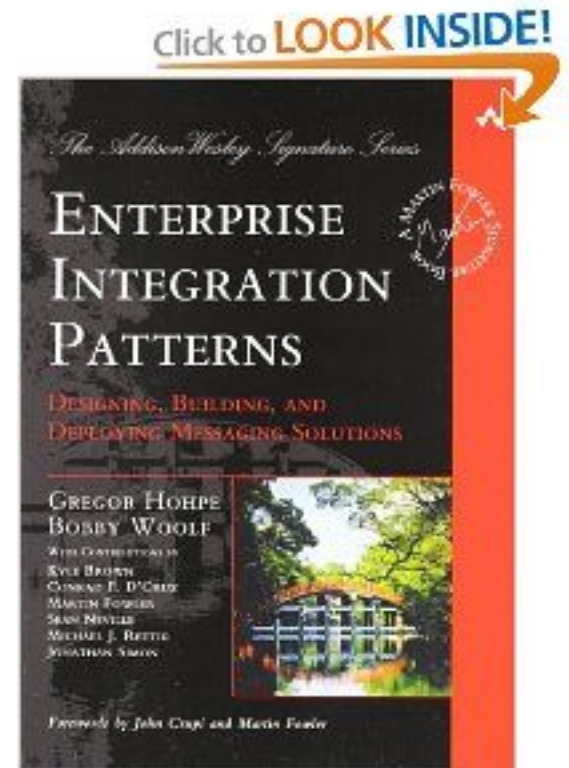
[Introduction](#)

[Point-to-Point Channel](#)



Enterprise Integration Patterns

- <http://www.eaipatterns.com/>
- The book
 - Enterprise Integration Patterns
 - Gregor Hohpe, Bobby Woolf



FUSE - LoanBroker/eip/credit_response.eip_diagram - FUSE

File Edit Diagram Navigate Search Project Run Window Help

Segoe UI 9 B I A 100%

Project Explorer

- LoanBroker
 - Spring Elements
 - src
 - JRE System Library [jre1.6]
 - build
 - eip
 - credit_response.eip
 - credit_response.eip.d
 - null

credit_response.eip_diagram

```

graph LR
    Input(( )) --> Check[Check...]
    Check --> Router{ }
    Router --> Accept[Accept]
    Router --> Reject[Reject]
    Accept --> ProcessReq[Process Req]
    ProcessReq --> InformCust[Inform Cust]
    Reject --> InvalidReq[Invalid Req]
    InformCust --> Output(( ))
    InvalidReq --> Output
  
```

Palette

- Patterns
 - Resequencer
 - Routing Slip
 - Splitter
 - Throttler
- Endpoints
 - CXF
 - Direct
 - File
 - Generic
- Camel Proces...
 - Bean
 - Catch
 - Convert Body To
 - Finally

Problems Tasks Properties Console Servers

Check Processor

Properties

General

Destination Parameters

*Type queue

*Name checkResponse

Anypoint Studio

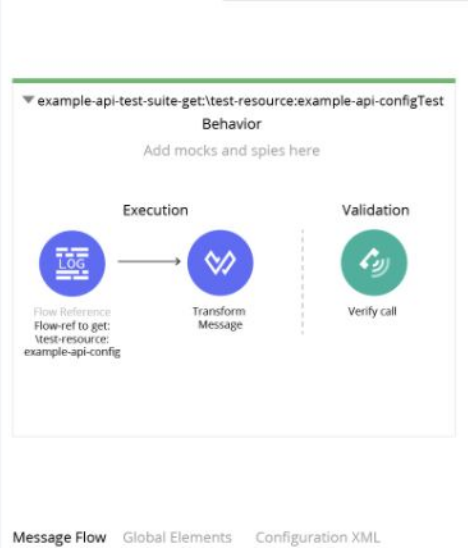
Package Explorer



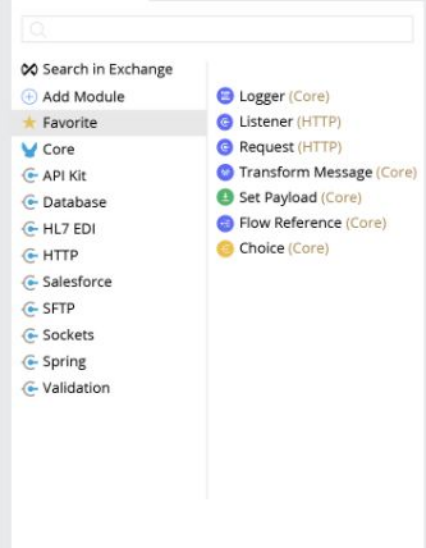
example-api



example-api-test-suite



Mule Palette



Outline

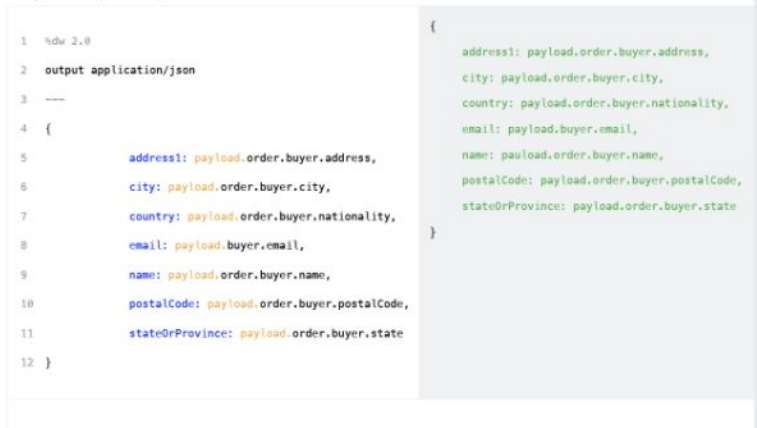


MUnit

Transform Message



Input Output Payload



workspace22wl - Integration Studio Template Dashboard - Integration Studio

Quick Access

Getting Started

Integration Studio

Getting Started

Start

New Integration Project
Open Project...

New Data Service Configs
New Data Source Configs

New BPMN Project
New Human Task Project

New Mediator Project
New Maven Multi-Module Project

Add Server

Try out a Sample

Select the sample and follow the instructions to get started.

Category
Search by sample or mediator name

Hello World Service
This sample demonstrates a simple HTTP service.

API Testing
This sample demonstrates unit testing of a REST API artifact using WSO2 Integration Studio.

Content Based Routing
This sample demonstrates content-based message routing in an integration solution.

Database Polling
This sample demonstrates how to periodically poll data from a database and then trigger a message flow.

Email Service
This sample demonstrates how to send and receive emails using WSO2 Email Connector.

Exception Handling
This sample demonstrates the exception handling capabilities of WSO2 Enterprise Integrator.

Fetch & Act

This sample how to test Salesforce

Help

Documentation
GitHub
Slack
Video Tutorials

☒ Always show on startup

workspace - HelloWorldSampleConfigs/src/main/synapse-config/api/HelloWorld.xml - Integration

Getting Started HelloWorldSample/pom.xml HelloWorld.xml

Project Explorer

- HelloWorldSample
 - HelloWorldSampleCompositeExporter
 - HelloWorldSampleConfigs
 - src
 - main
 - synapse-config
 - api
 - HelloWorld.xml
 - endpoints
 - inbound-endpoints
 - local-entries
 - message-processors
 - message-stores
 - proxy-services
 - sequences
 - tasks
 - templates
 - test
 - pom.xml
 - ReadMe.html
 - pom.xml

Design Source Swagger Editor

Palette

API

Mediators

Call

CallTre

Drop

Log

Loop

Prope

Prope

Respx

Send

EndPoints

Defined S

Defined E

Email Con

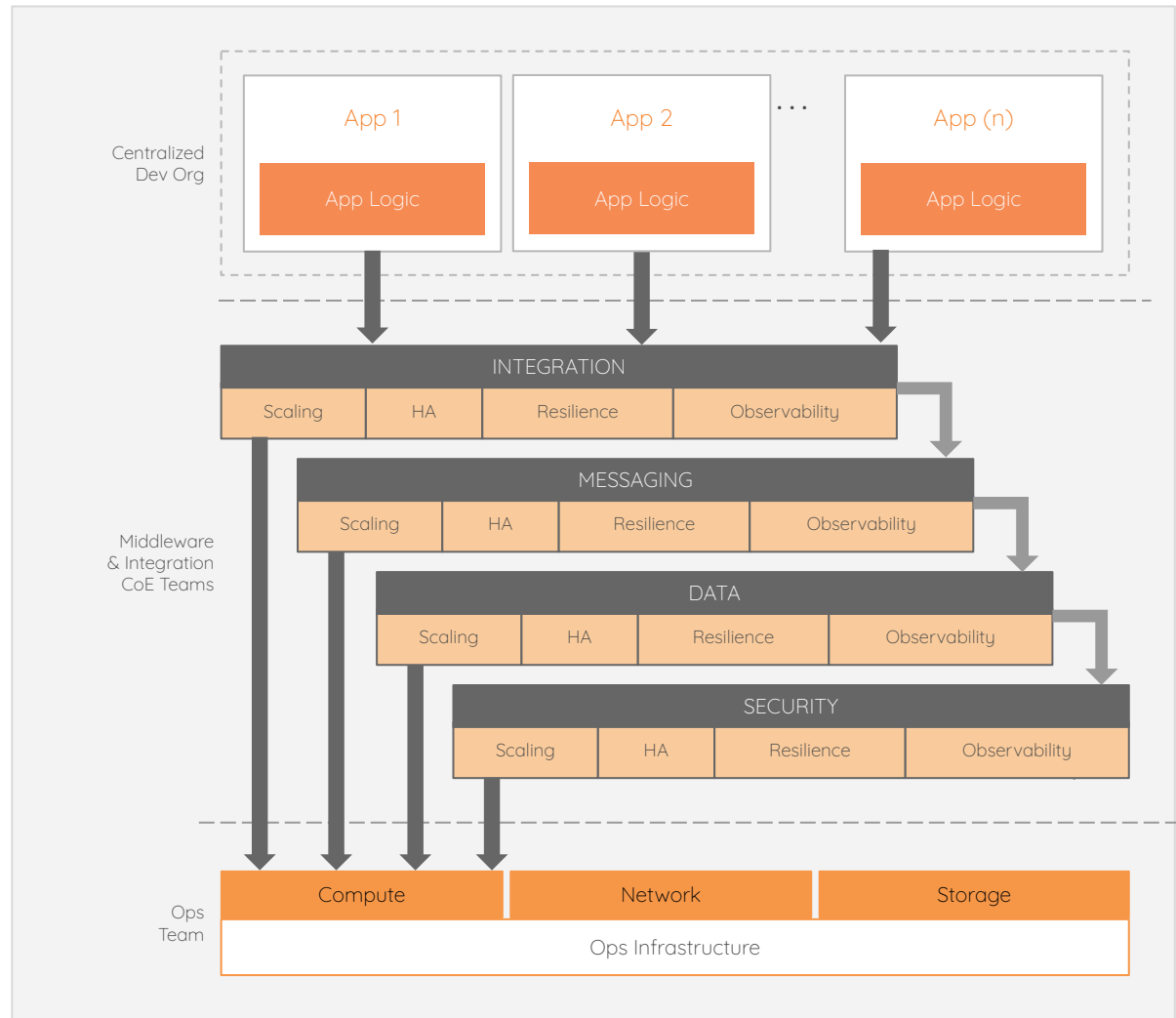
Salesforc

Fileconne



Fast Waterfall

“Wagile”
“Fagile”



How does mediation / integration fit into Microservices / Containers?

“Smart Endpoints and Dumb Pipes”





Brownfield

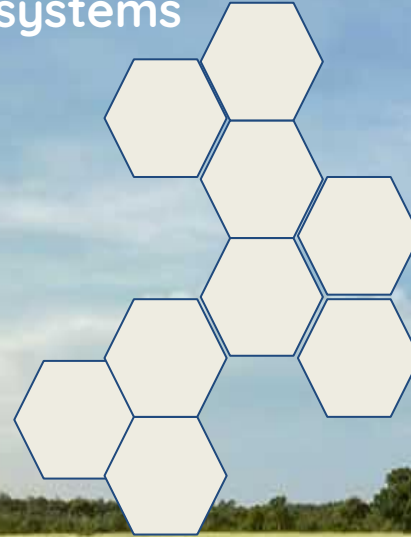
Greenfield

Core
Systems

Core
Systems

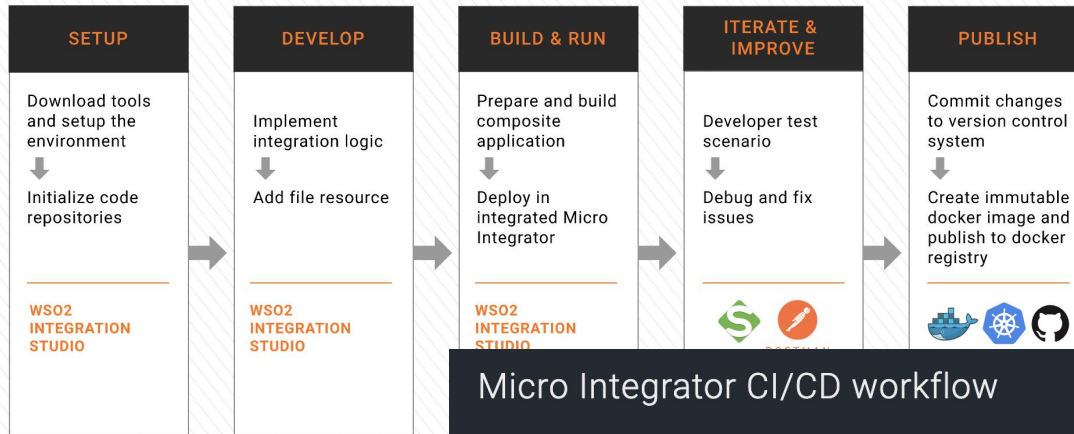
Core
Systems

Disaggregated
systems

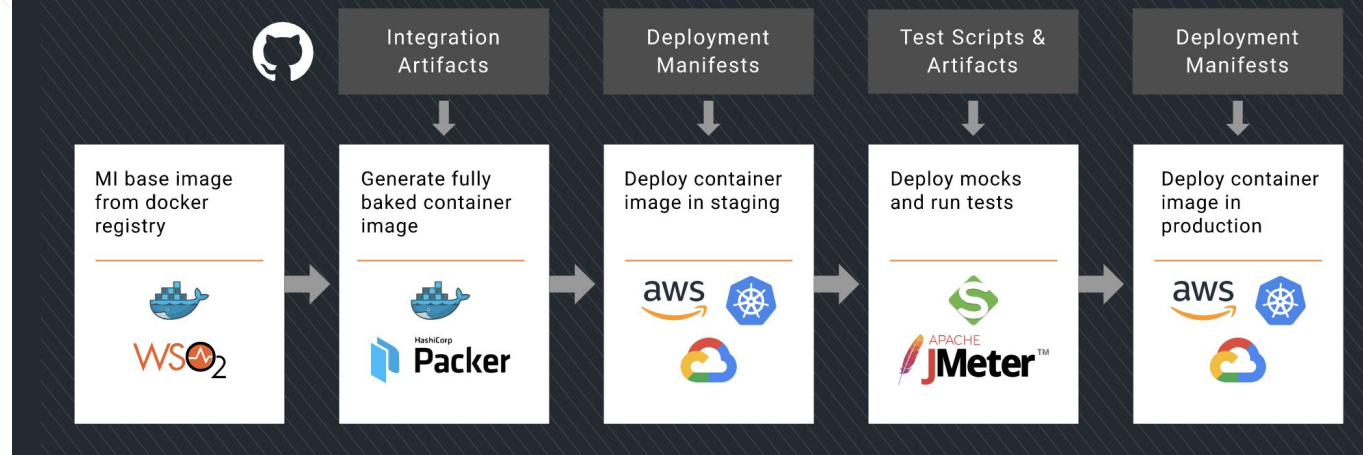


WSO2 Micro Integrator

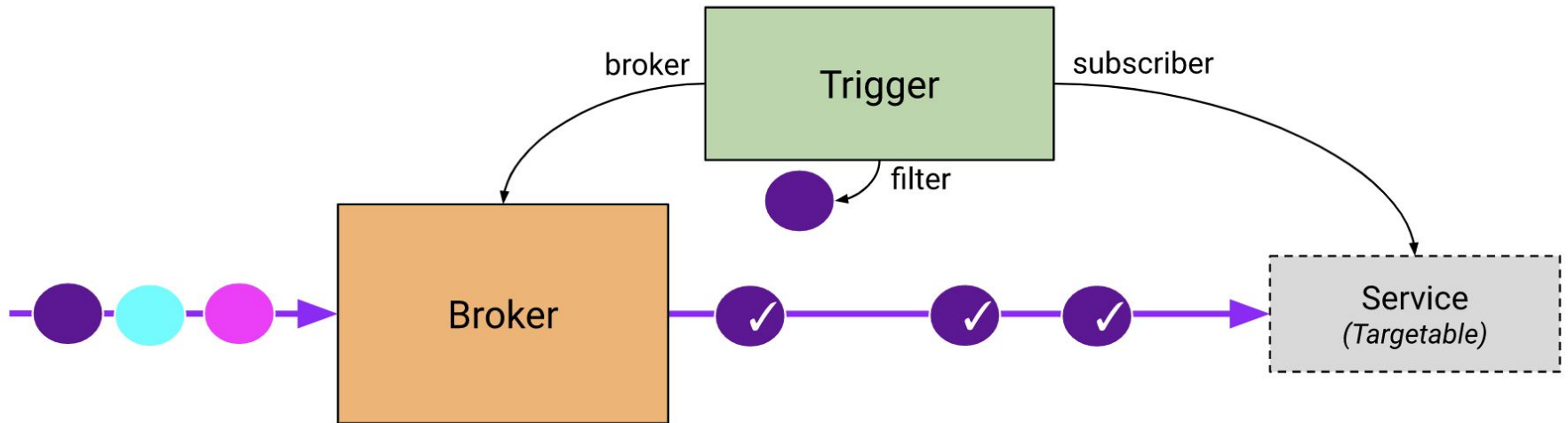
Micro Integrator Developer Workflow



Micro Integrator CI/CD workflow



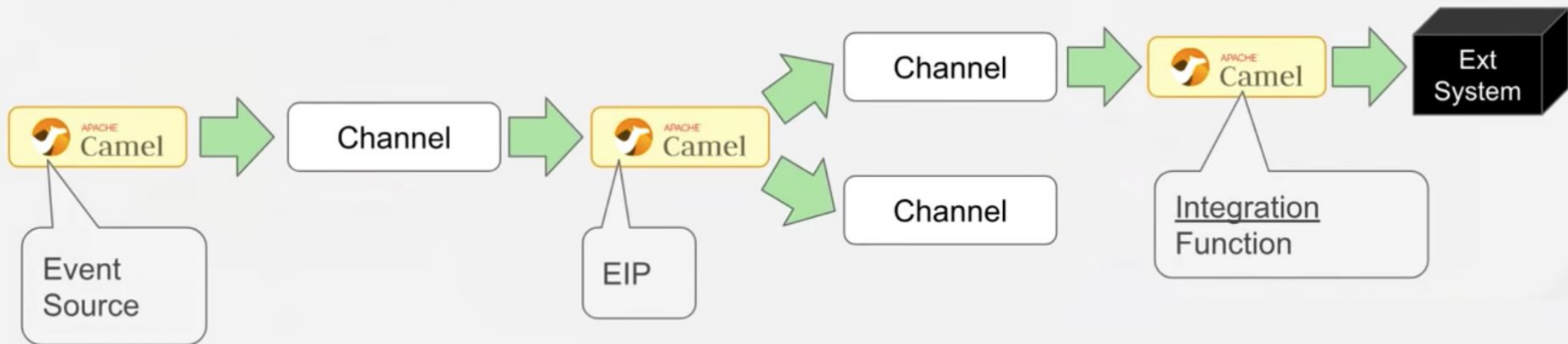
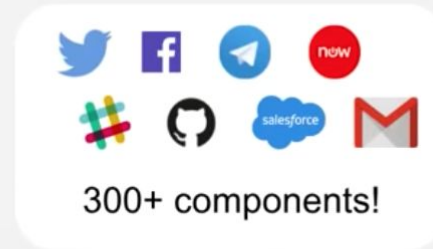
knative



camel-k

Knative

Camel K: same model for different purposes



Ballerina Language

- A new integration language and framework for Microservices, RESTful SOA
- Based on Swagger and Sequence Diagrams
- Textual and graphical are 100% interchangeable
- <https://ballerina.io>



Ballerina diagram and language

The screenshot displays the Ballerina IDE interface. The left pane shows the source code for `demo.bal`, which defines an HTTP service named `hello` with a `listener` binding. The service has a `ResourceConfig` with `path: "/"`, `methods: ["POST"]`, and `body: "js"`. The `hi` endpoint, accessible via `caller`, receives an `http:Request` and a `json js`. It extracts `title` and `content` from the JSON, creates a `github4:Issue` object, and calls `git->createIssue` with the issue details. A response object `resp` is constructed and sent back to the `caller` via `respond(resp)`.

The right pane, titled "Ballerina Diagram", shows a sequence diagram for the `hi` endpoint. It illustrates the interaction between the `caller`, the `default` service boundary, and the `git` external service. The sequence of messages is: `request.js` from `caller` to `default`, followed by a call to `createIssue("pzfreo", "btest", title, content, [], [])` on the `git` service, and finally `respond(resp)` from `default` back to `caller`.

At the bottom of the IDE, the status bar indicates the current position: `Ln 33, Col 13`, with settings for `Spaces: 4`, `UTF-8`, `LF`, and the `Ballerina` language.

Choreo

(currently private beta)

The screenshot displays the Choreo web interface. At the top, there's a navigation bar with a menu icon, the Choreo logo, and links for 'Integration list', 'Help Center', 'About Choreo', and a user profile for 'Paul Fremantle'. The main workspace is divided into two panels. The left panel shows a workflow diagram on a grid. The workflow starts with a 'WEBHOOK' trigger, followed by 'release...', 'message', and 'releaseT...' steps. It then enters a 'For Each' loop containing a 'Variable' step and a 'releas...' step. After the loop, there's a 'newMessa...' step, a 'slackClient' connector, and a 'postMessage' action. The workflow ends with an 'END' node. The right panel shows the source code for the workflow, which is a Kotlin script. The code defines a service 'githubListener' with a function 'onReleased' that processes a 'ReleaseEvent' and sends a message to a Slack channel. The code includes comments and uses various Kotlin constructs like 'remote function', 'string', 'foreach', and 'slackClient'.

Integration list

GitHub New Release to Slack Channel Message

Help Center About Choreo Paul Fremantle

Develop Test Go Live Observe

RUN & TEST

```
37     auth: gitHubTokenConfig
38   }
39 }
40 service / on githubListener {
41   remote function onReleased(webhook:ReleaseEvent event) returns error? {
42     webhook:Release releaseInfo = event.release;
43     string message = "There is a new release in GitHub ! \n";
44     [string,string][] releaseTuples = [[VERSION_NUMBER, RELEASE_TAG_NAME],
45     message += "<" + releaseInfo.get(RELEASE_URL).toString() + ">\n";
46     foreach var releaseTuple in releaseTuples {
47       var [description,keyFromMap] = releaseTuple;
48       if (releaseInfo.containsKey(keyFromMap)) {
49         message += description + SEMICOLON + releaseInfo.get(keyFromMa
50       }
51     }
52     slack:Message newMessage = {
53       channelName: slackChannelName,
54       text: message
55     };
56
57     slack:Client slackClient = check new (slackConfig);
58     var result = check slackClient->postMessage(newMessage);
59   }
```



Resources

- Wikipedia!
 - http://en.wikipedia.org/wiki/Enterprise_bus
- Books
 - David Chappell: ESB
 - Open Source ESBs in Action
- Open Source
 - synapse.apache.org
 - wso2.com/products/enterprise-service-bus
 - servicemix.apache.org

