Projekt Open-ESB

Business Integration mit Open Source

Daniel Adelhardt

Software & Java Architekt

Sun Microsystems GmbH

Agenda

- Open-ESB – Einführung
  - Projekt Zielsetzung
  - Java Business Integration als Infrastruktur Layer
  - GlassFish v2 – Die Runtime von Open ESB
- Open-ESB – Aufbau und Architektur
  - Überblick über Engines und Komponenten
  - BPEL 2.0 Engine
  - Java EE Engine
  - Scripting Engine
- Tooling
  - Service Erstellung und Orchestration
  - Demo, Demo,Demo

Projekt Open-ESB Überblick

- Open ESB implementiert einen Enterprise Service Bus auf Basis der Java Business Integration Spezifikation
  - Infrastruktur für den Betrieb von Composite Applications
  - Umfangreiches Tooling für die rasche Entwicklung von Composite Applications
- Open ESB ist eine offene Community von Sun, ISVs and Open Source Vertretern
  - Open ESB schafft einen ***offenen*** Marktplatz für Integrationskomponenten
- Open ESB ist ein Open Source Projekt
  - Lizenz derzeit CDDL (Common Development and Distribution License)
  - Geplant: GPLv2 + Classpath Exception (wie bei OpenJDK, GlassFish et al)
  - Source, CVS, Issuetracking, Forum gehostet auf http://open-esb.dev.java.net

Open ESB

The *Open* Enterprise Service Bus

Composite Applications?

Composite application - From Wikipedia, the free encyclopedia

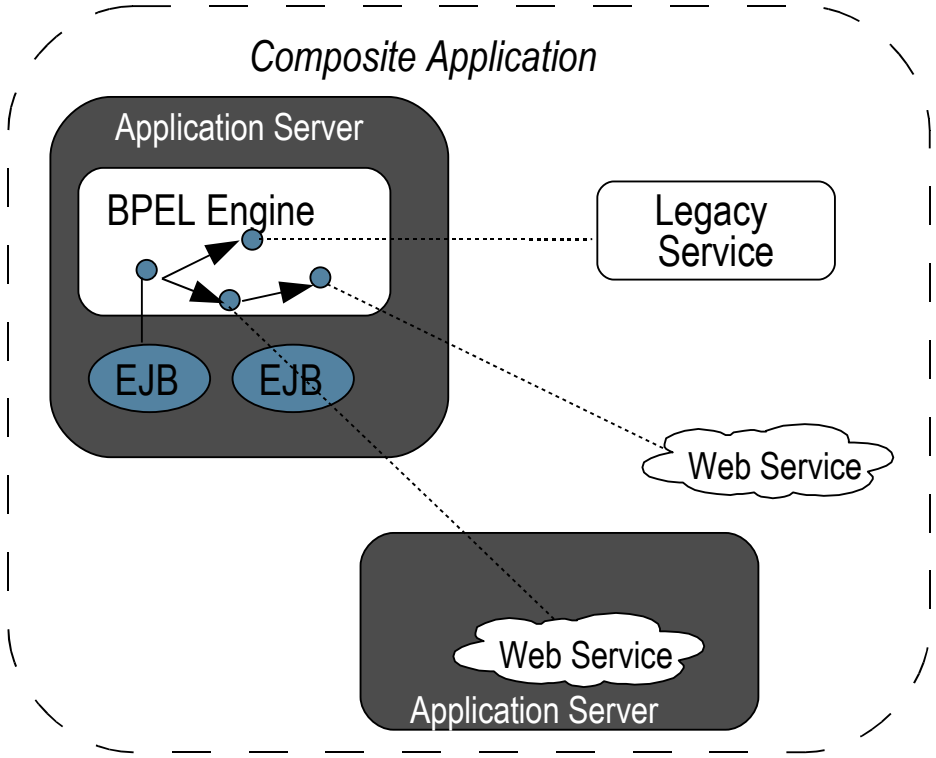
... the term composite application expresses a perspective of software engineering that defines an application built by combining multiple services. A composite application consists of functionality drawn from several different sources within a service oriented architecture (SOA). The components may be individual web services, selected functions from within other applications, or entire systems...

Composite Applications?

- Wo machen Composite Applications Sinn?
  - Nicht jede klassische Java EE Applikation muss als Composite Application realisiert werden...
- Composite Applications adressieren
  - Heterogene Systeme
  - Klassische Anwendungsintegrationsszenarien
  - Erstellung von “SOA” Architekturen
  - Systeme mit hohen Anforderungen bzgl. Flexibilität und hoher Änderungshäufigkeit

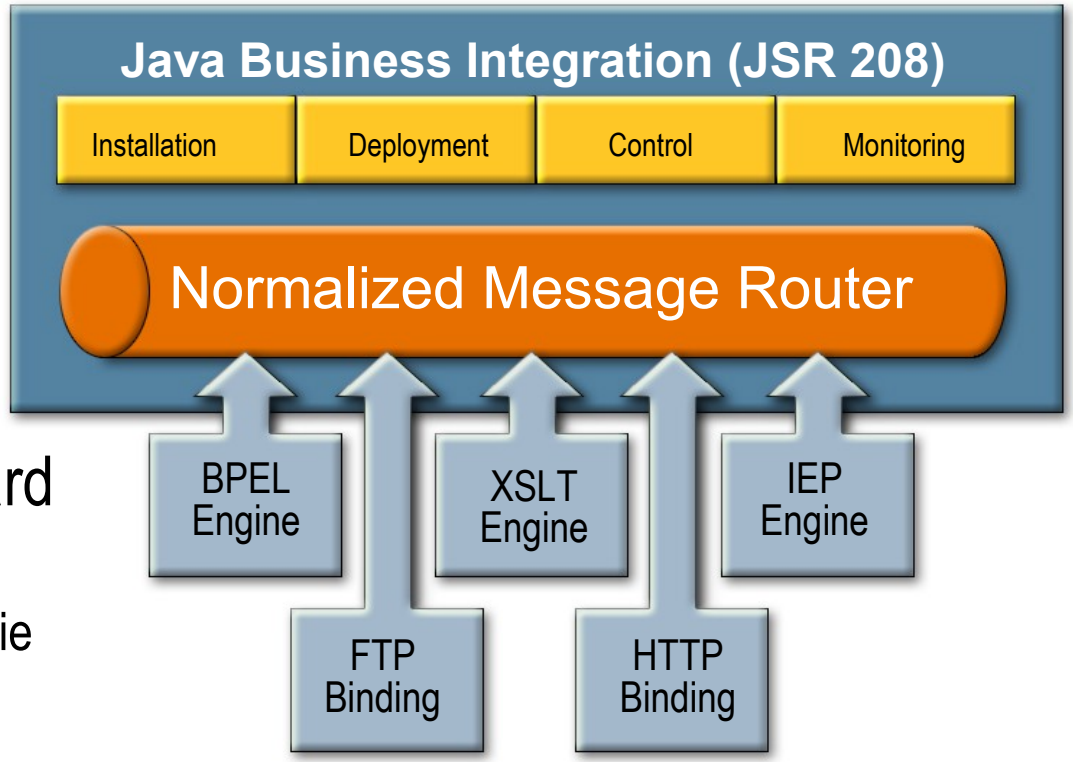
Beispiel für eine Composite Application

- Composite Applications sind ein Verbund unterschiedlichster Technologien, Runtimes, Sprachen, die als Ganzes eine Funktion oder Prozess realisieren
  - > Beispiel: BPEL Prozess, der Web Services, EJBs, Legacy Systeme koppelt
- Fragestellung für Composite Applications
  - > Wie erfolgt das Packaging der Applikation?
  - > Wie erfolgen Management und Deployment?
  - > Welche Standards gibt es für die Infrastruktur?



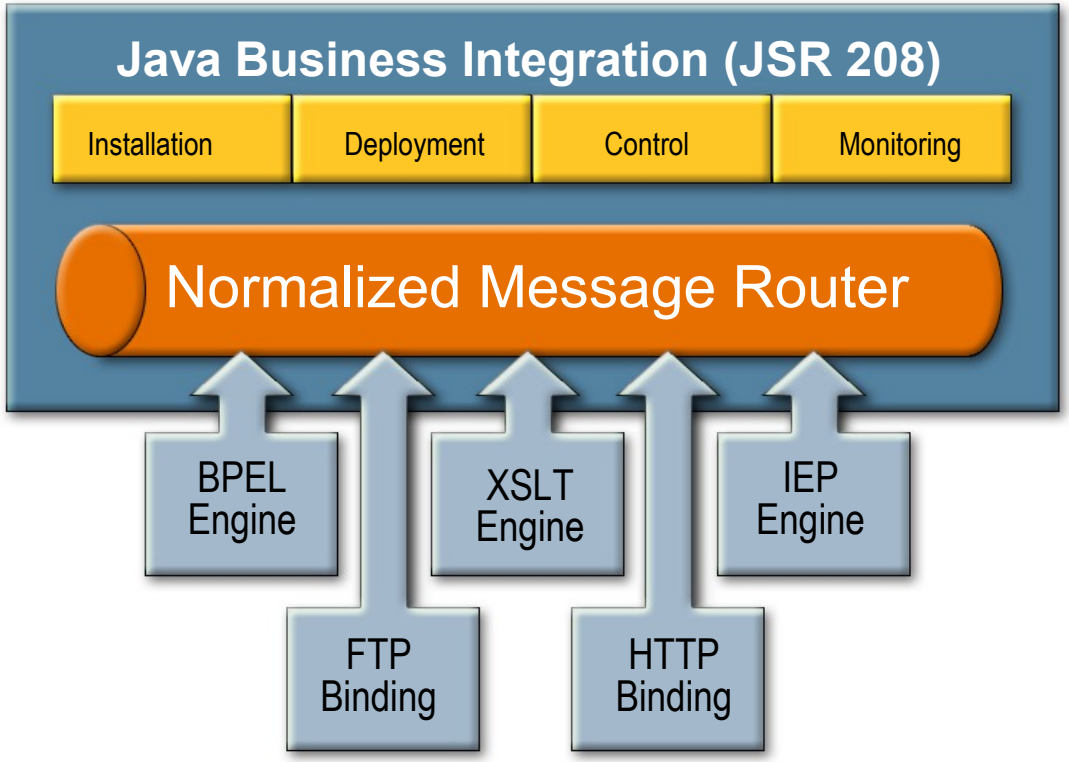
Java Business Integration (JBI - JSR-208)
Die Grundlage von Open ESB

- JBI spezifiziert eine Meta Container Infrastruktur
  - Lifecycle, Deployment & Management Services für Composite Applications
  - Normalisiertes Routing von Nachrichten
  - Standardisierte Plugin Schnittstellen für Service Engines und Binding Components
- JBI definiert *keine*
  - Programmiermodelle
  - APIs für Service Entwickler
- JBI definiert einen Standard
  - Für ESBs Plugins
  - Für die Infrastrukturtechnologie eines ESBs

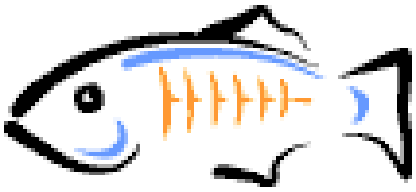


Java Business Integration (JBI - JSR-208)
Die Grundlage von Open ESB

- Die Entwicklung von Services und Logik erfolgt gegen die Programmiermodelle der Service Engines
  - Java EE, BPEL, Java Script,...
  - JBI stellt einen Weg bereit Deploymentartefakte als Service Assemblies auf die einzelnen Engines und Binding Komponenten zu verteilen

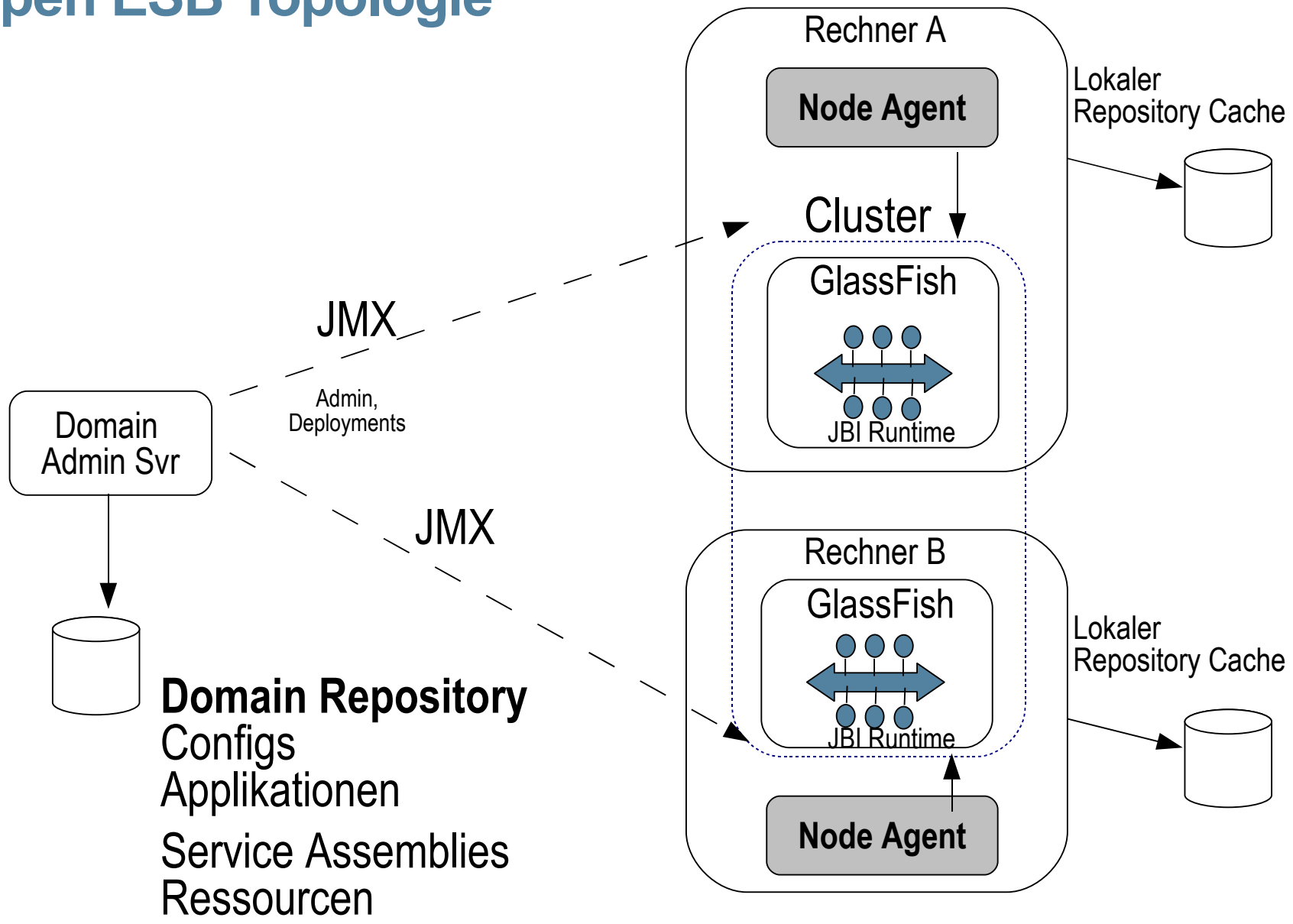


Bestandteile von Open ESB

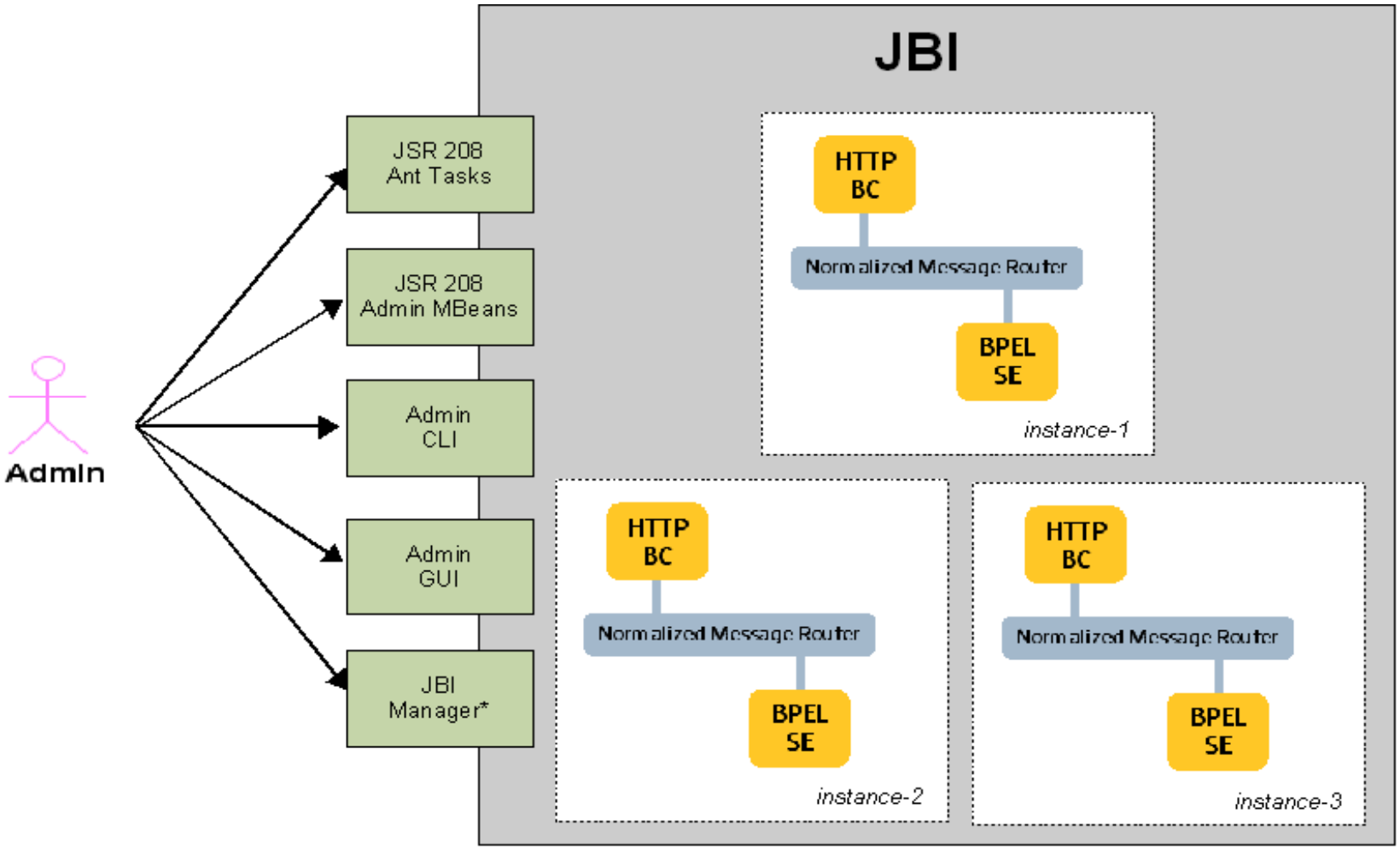


- Open ESB ist eine Erweiterung des GlassFish Application Servers
  - GlassFish ist der Open Source Java EE Server von Sun http://glassfish.dev.java.net
  - GlassFish v2 integriert die komplette Infrastruktur für Java Business Integration in den Java EE Container
  - Administration, Monitoring, Logging für JBI integriert in den Application Server
  - Clustering, Loadbalancing und Failover des Application Servers werden mitgenutzt
- WSIT aka Project Tango in OpenESB/GlassFish beinhaltet
  - WS-Security, WS-ReliabileMessaging, WS-AtomicTransaction, WS-Coordination, ...
- Das Open ESB Projekt erweitert den GlassFish JBI Container um Engines und Binding Componenten für Applikationen

Open ESB Topologie



Topologie im Detail

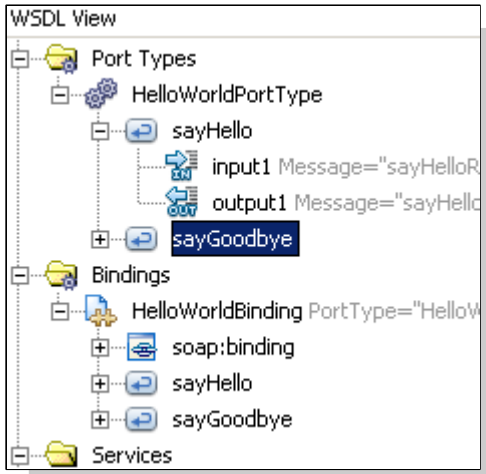


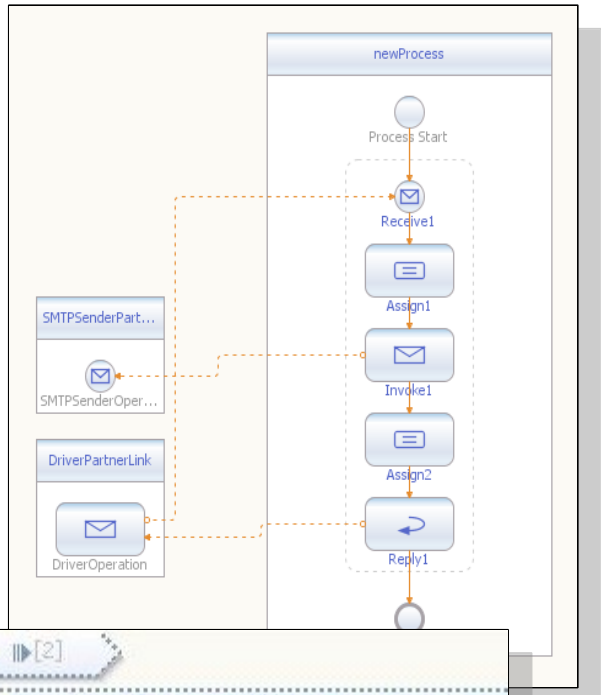
Open ESB Komponenten

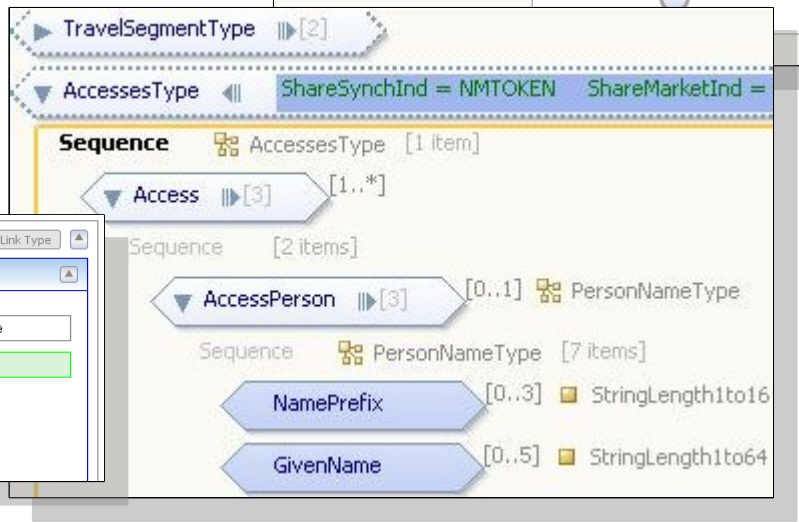
- Open ESB Service Engines (Auswahl)
  - Java EE Service Engine, BPEL 2.0 Service Engine
  - Scripting Service Engine, XSLT Transformation Engine
  - Intelligent Event Processing Engine, Data Mashup Engine
  - (Human) Workflow Engine (WLM SE)
- Open ESB Binding Components
  - Kommunikation: TCP/IP, HTTP, LDAP,SMTP, SOAP,FTP,DCOM,
  - Application: MQ-Series, SAP, CICS, SWIFT, MSMQ, IMS
  - Telco: SIP, XMPP, CORBA, SNMP
- Liste aller Komponenten unter https://open-esb.dev.java.net/Components.html

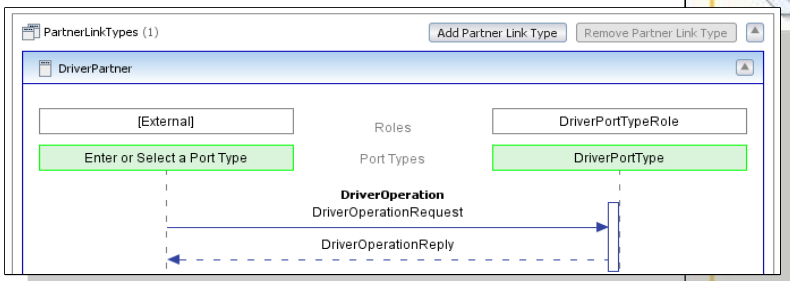
Umfassendes Tooling per NetBeans

- WS-BPEL
  - Editing
  - Deploying
  - Debugging
  - Testing
- > XML
  - XML Schema
  - WSDL Creation and Visualization









Ein Hello World mit Open ESB

- Ziel: Erzeugung eines ersten lauffähigen Open ESB Projektes mit Web Services, BPEL
- Bestandteile:
  - BPEL Prozess
  - XML Schema
  - Unit Tests
- Vorgehensweise
  - Erstellung eines XML Schemas
  - Erstellung eines BPEL Prozesses
  - Wrapping als JBI Assembly
  - Test

DEMO

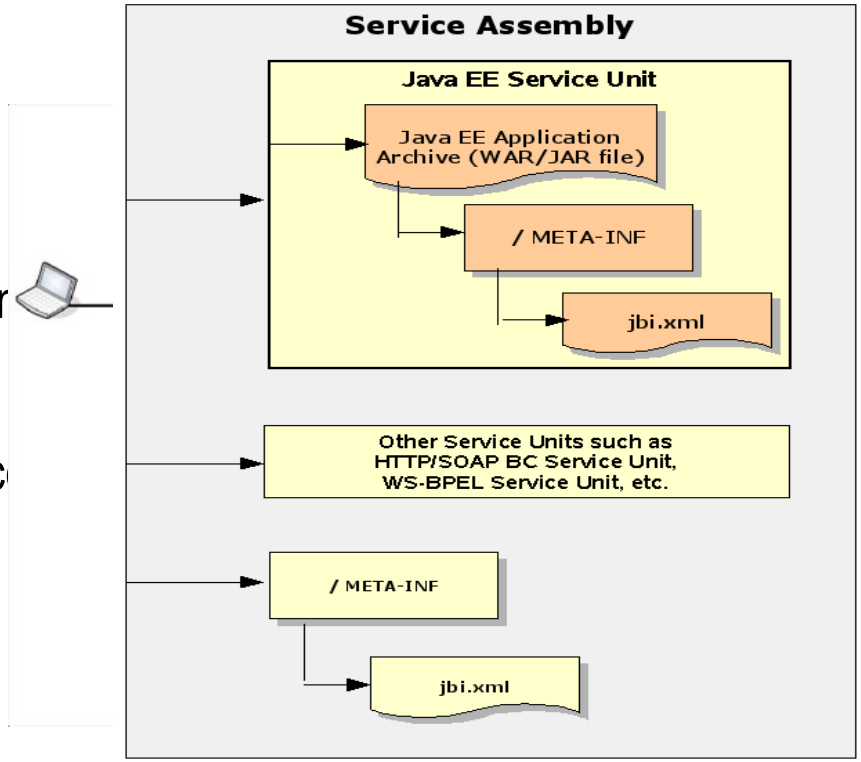
Live Demo der Service Erstellung

Open ESB und Java EE

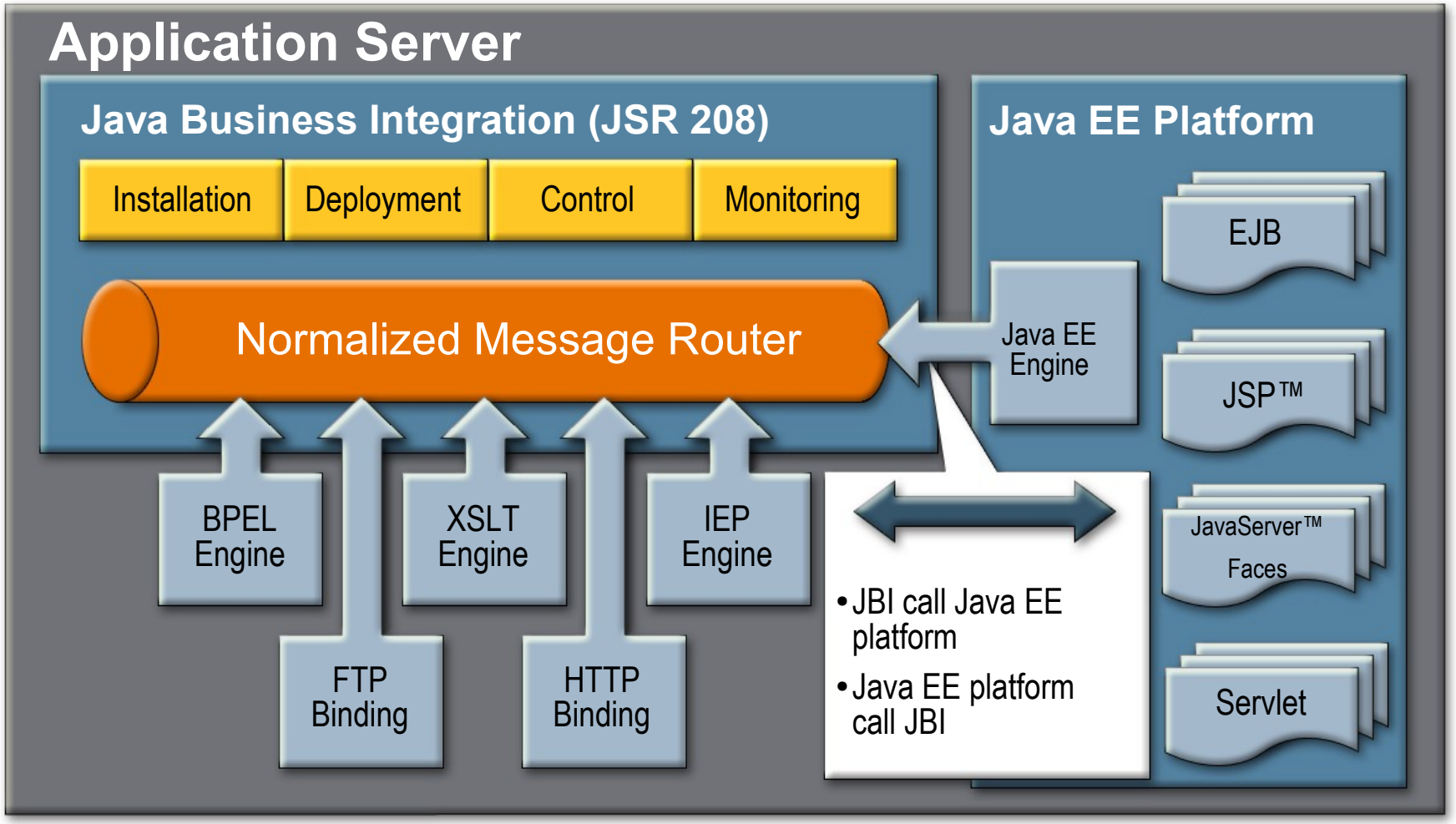
- Welche Rolle spielt Java EE in einer JBI/ESB Welt???
  - JBI definiert kein Programmiermodell für Services!
  - Entwickler implementieren Business Logik und Services für Engines
  - Java EE ist “nur” eine mögliche Engine in einem JBI Container
- Java Enterprise Edition 5 ist eine gute Plattform für Services
  - EJB 3.0 und Java Persistence API sind einfach zu nutzende APIs für die Implementierung von Business Logik
  - JAX-WS 2.x API als Standard API für Web Services definiert ein einfaches Programmiermodell für Web Services
- Wir betrachten Java EE / JBI als ideale Kombination für die Implementierung von Composite Applications
  - SCA setzt auf anderem Level auf

Die GlassFish/OpenESB Java EE Service Engine

- Nicht immer sind alle Artefakte im ESB verteilt
  - Java EE Service Engine bietet direkten Zugriff auf Java EE Komponenten aus dem JBI Container
  - Übliche Weg wäre Servlet/EJB per SOAP/HTTP anzusprechen
- Benefits der Java EE Engine
  - Transaction und Security Support
  - Direkter Aufruf von Service Endpoints ohne SOAP Overhead
  - Co-Packaging von JavaEE/JBI Kompor
- Usecases
  - BPEL Prozess ruft JavaEE Web Servic
  - MDB/Servlet ruft BPEL Prozess auf
  - Java EE Komponenten rufen Web Services per SMTP/JMS Binding



GlassFish/OpenESB Integration

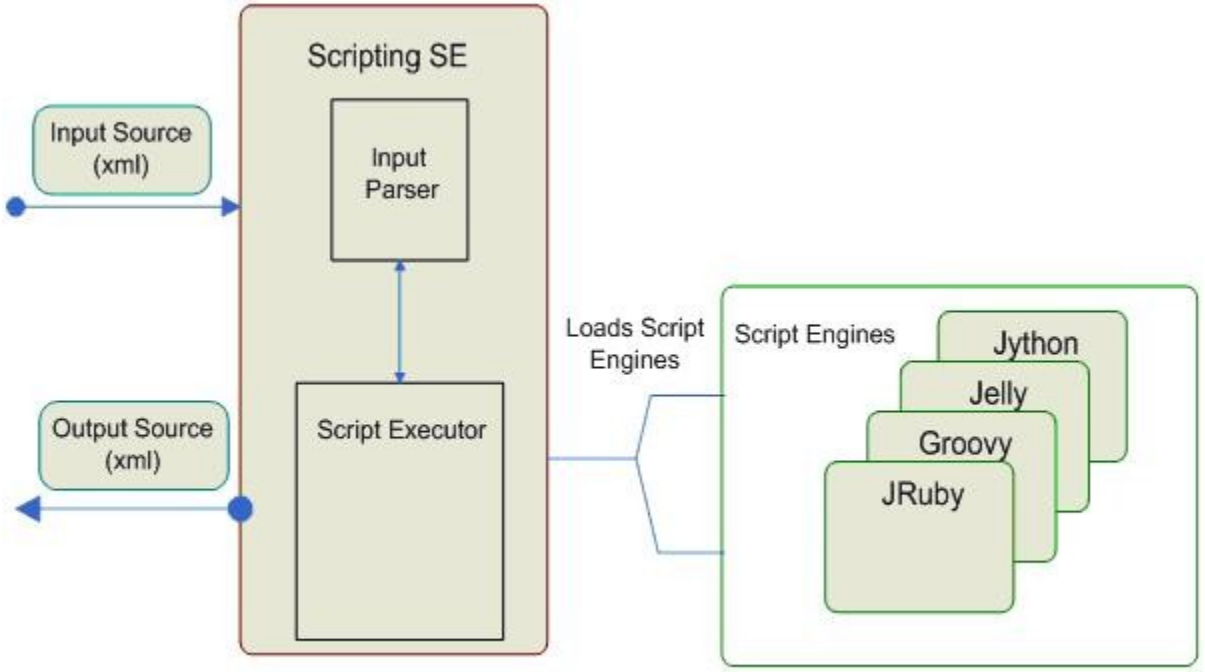


Open ESB BPEL Engine

- BPEL Engine ist eine WS-BPEL 2.0 Implementierung als JBI Service Engine
  - Komplette Unterstützung für alle WS-BPEL 2 Konstrukte
  - Unterstützung für multithreaded Ausführung
  - Debugger Support
  - Persistenz von Business Prozessen (optional) über Standard Application Server DataSource
  - Load Balancing und Clustering per Infrastruktur
- Kein Human Workflow!
  - Separate Workflow Service Engine verfügbar
- Tooling per NetBeans BPEL Designer

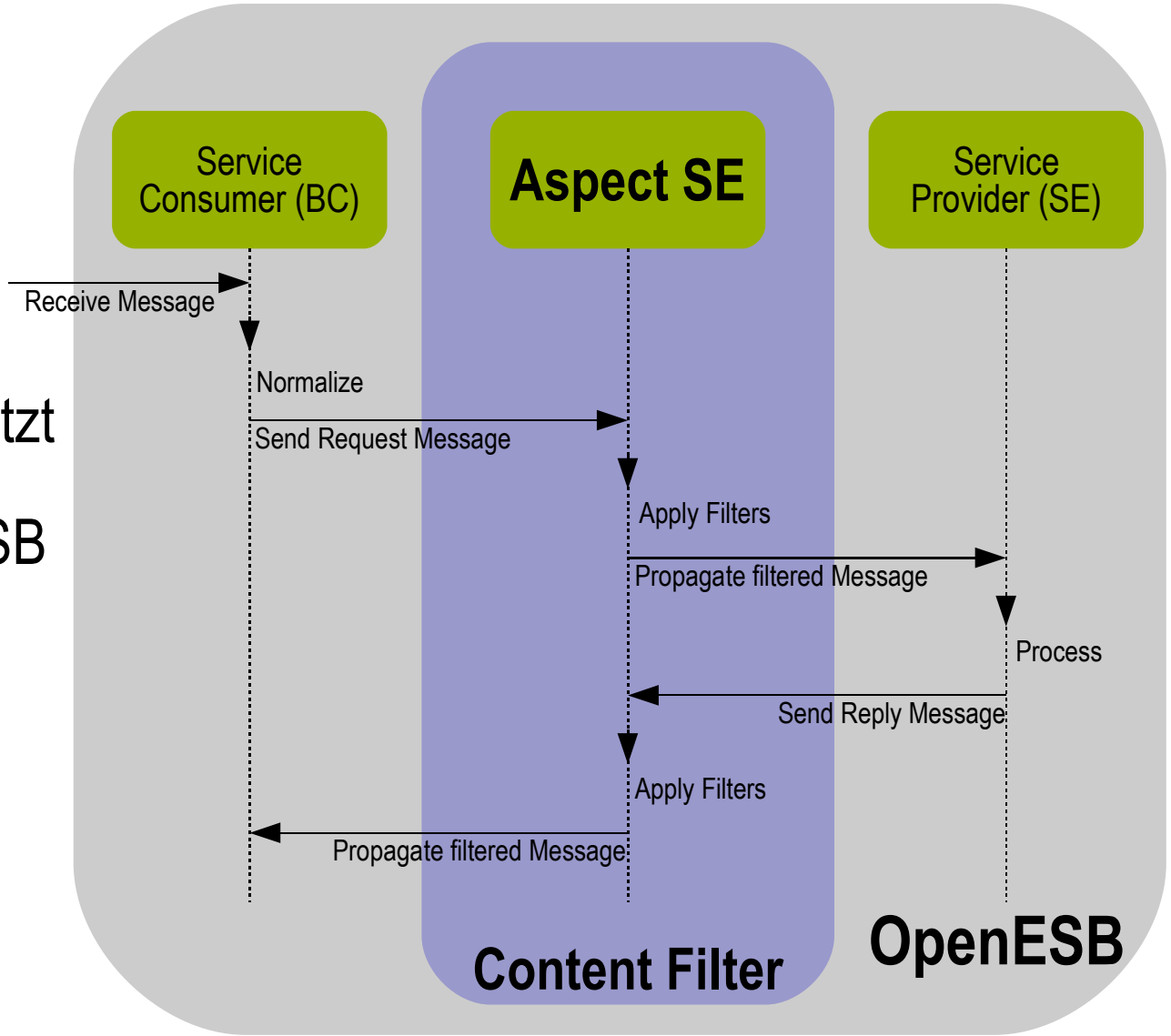
Scripting Service Engine

- Die Scripting SE ermöglicht die Implementierung von Business Logik in beliebigen Script Sprachen
  - > Verwendung von JSR 223 (Java Scripting Integration)
  - > Input XML enthält die Parameter für die Script Ausführung



Aspect Service Engine

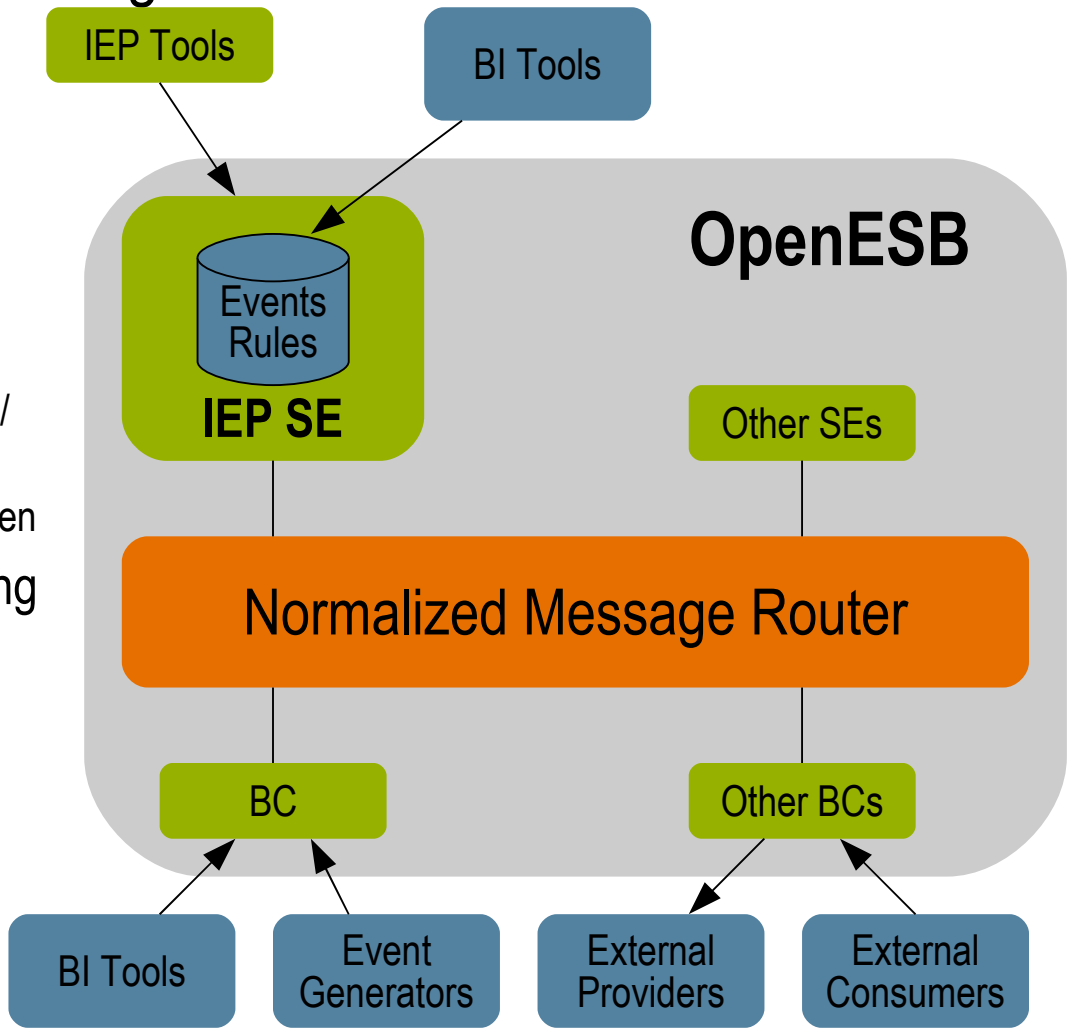
- Aspect SE unterstützt Einbringen von Aspekten in den ESB Nachrichtenfluss
  - > Policies
  - > Common Services
  - > Compliance



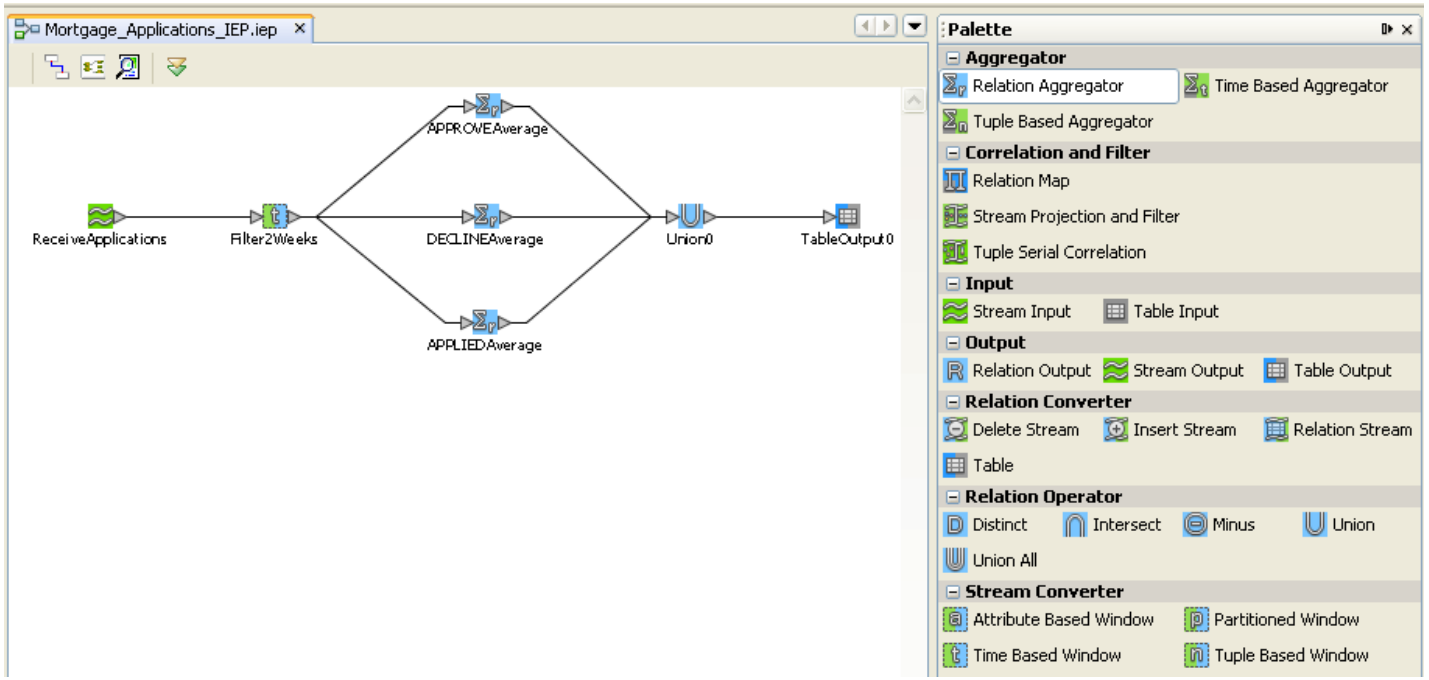
* interaction with OpenESB NMR intentionally left out

Intelligent Event Processing Service Engine

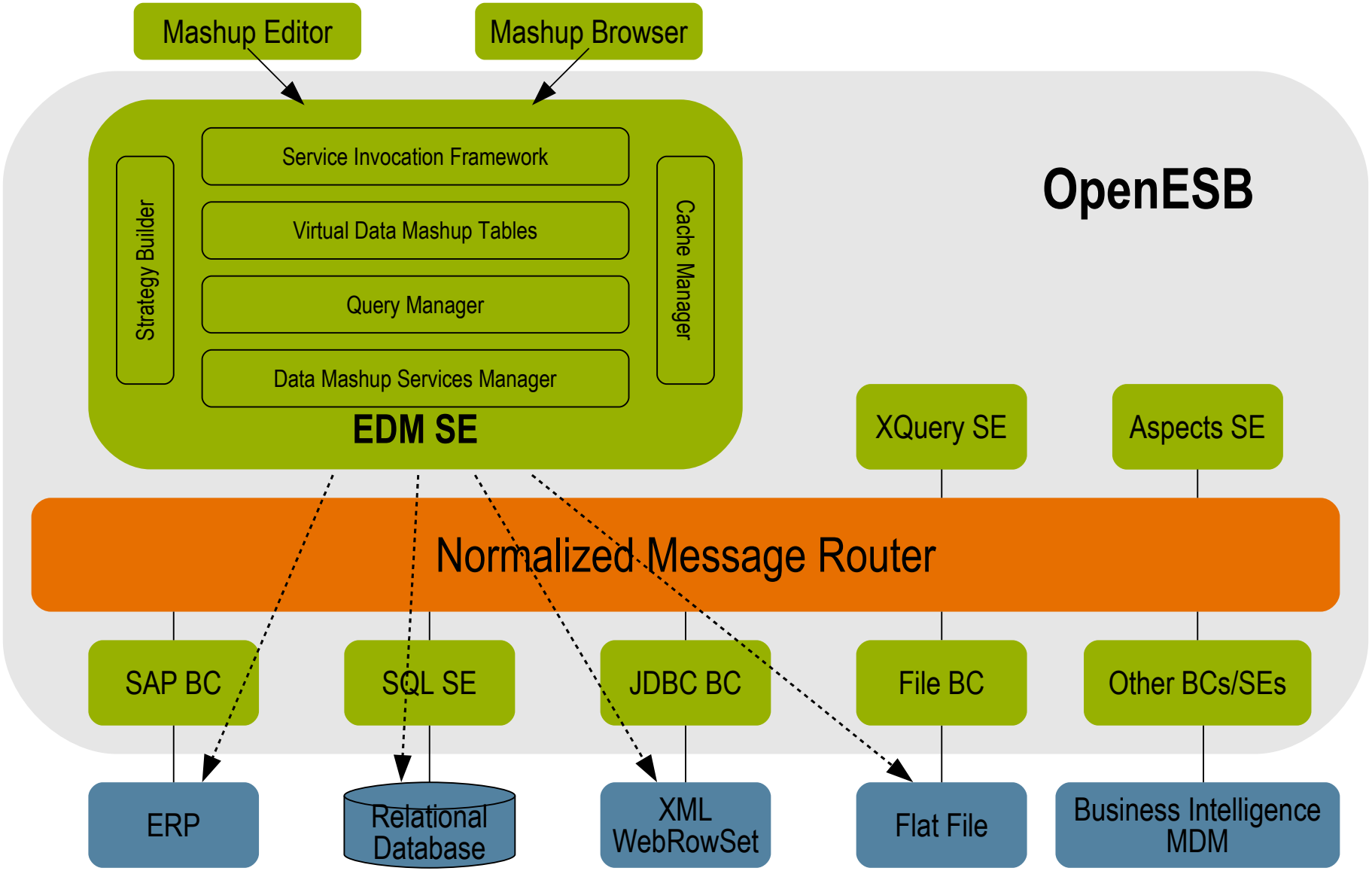
- Die Event Processing SE ermöglicht die Event getriebene Verarbeitung von Datenströmen
  - > Anwendungsszenarien:
    - Real Time Datenanalyse im Finanzumfeld
    - Auditing von Handelsvorgängen / Compliance Prüfung
    - Steuerung von Produktionsflüssen
  - > Realtime Collection, Processing Analyse und Notification
  - > Bereitstellung von Operatoren auf Daten:
    - Aggration, Relationenbildung, Korrelation



Intelligent Event Processing Tooling

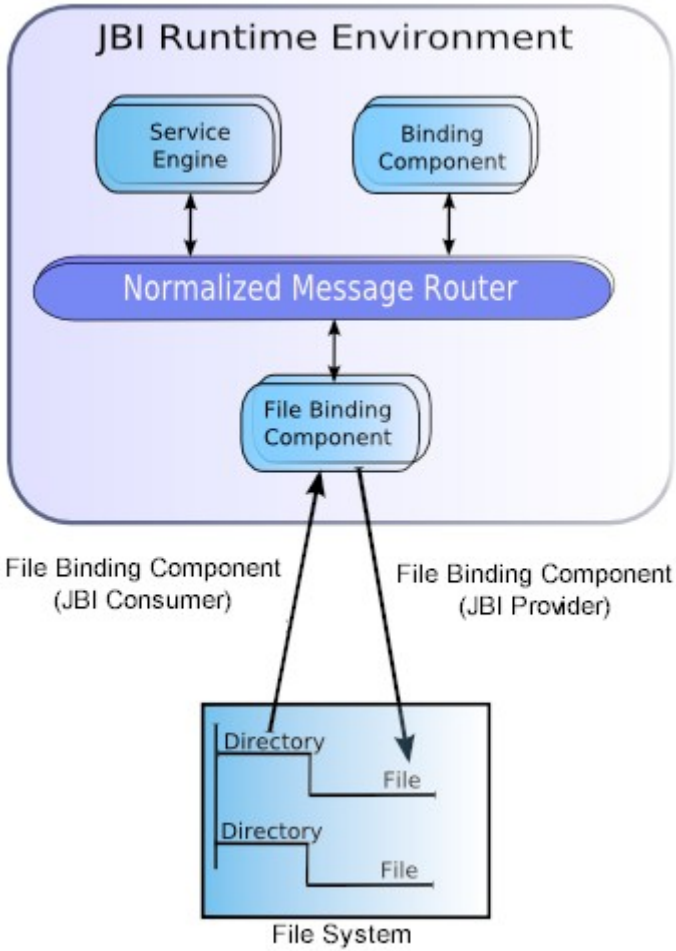


Enterprise Data Mashup Service Engine



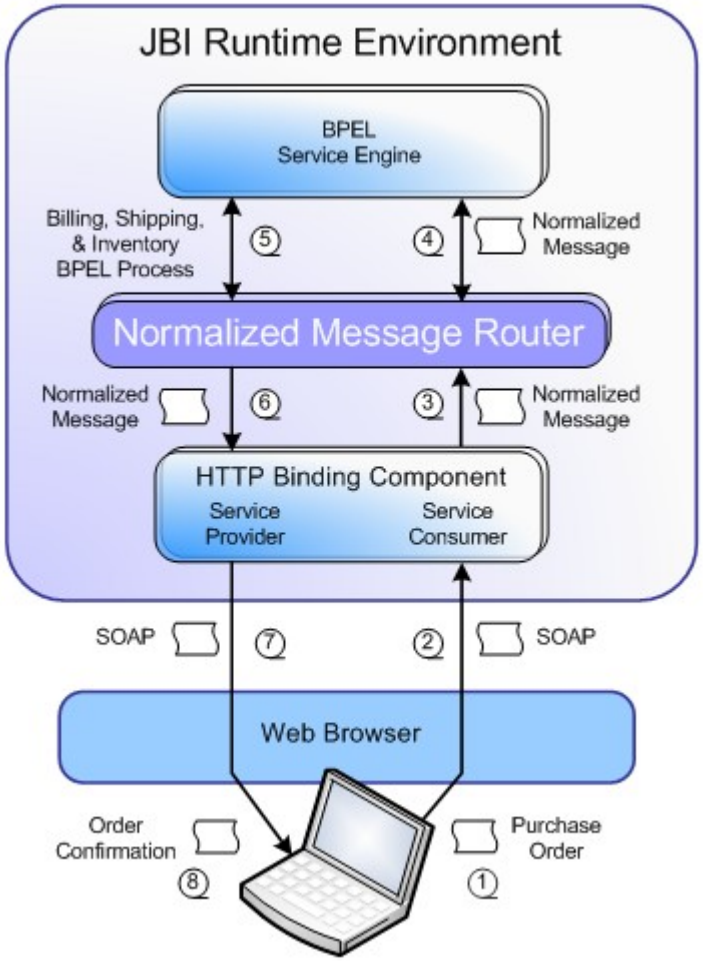
File Binding Component

- Die File BC kapselt Interaktion mit File Systemen für Dateiimport/export
  - > Konfigurierbarkeit bezüglich
    - Verzeichnissen
    - Dateinamen
    - Appendverhalten
    - Locking
    - Pollingverhalten
    - Uvm.
  - > Patternmatching
  - > Newline/CRLF Behandlung
  - > Encoding
  - > Single Message / Multiple Message Behandlung p Steuer/Trennzeichen



SOAP-HTTP Binding Compont

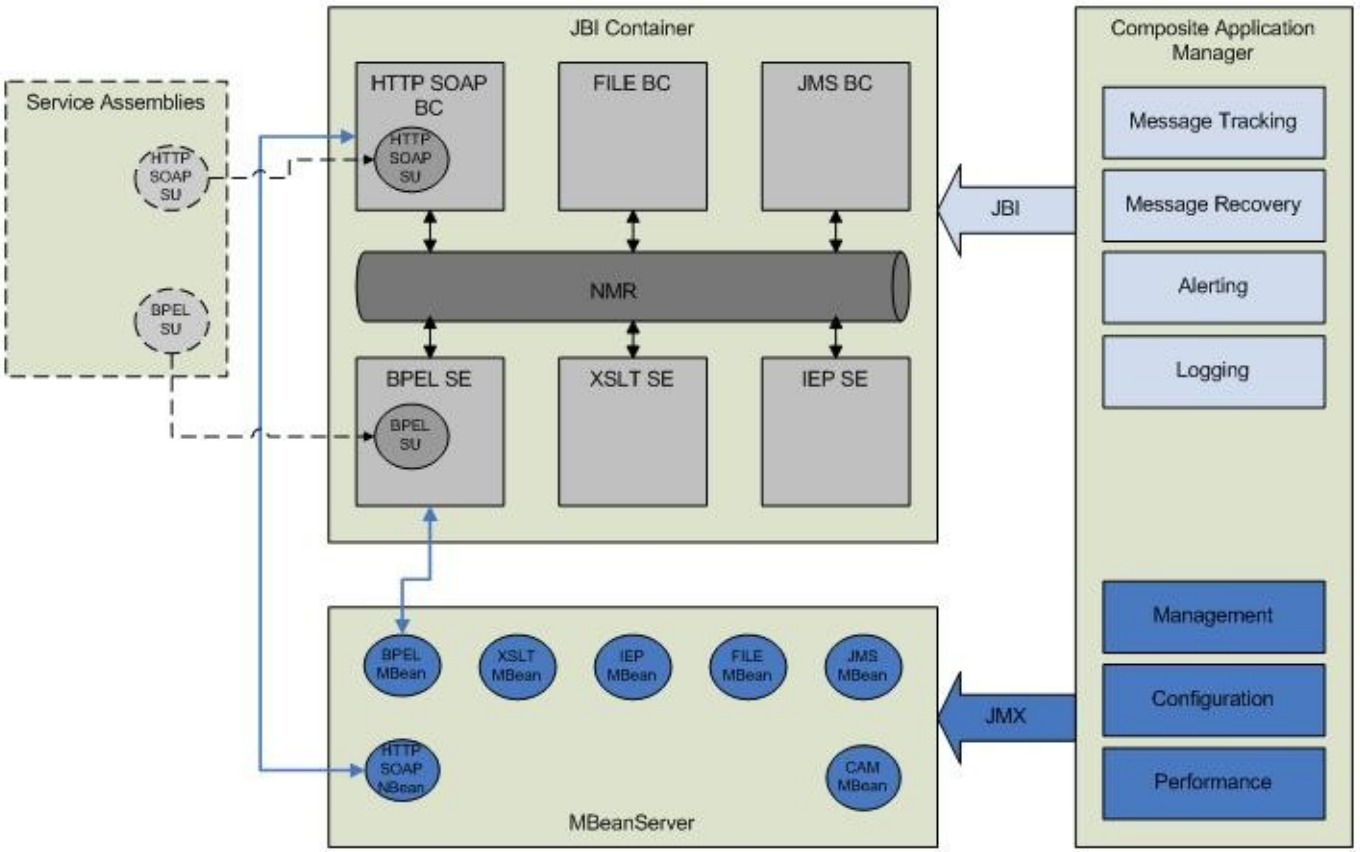
- SOAP-HTTP BC baut auf dem Grizzly NIO Framework auf
  - > Stellt SOAP/HTTP Connectivity für JBI bereit
  - > SOAP 1.1, WS-I BP, Security Profile
  - > Document Literal, RPC Encoded, RPC/Literal
- Unterstützung für Loadbalancing und Clustering
  - > Verwendung des GlassFish Loadbalancing Plugins für HTTP(s)



Weitere Binding Komponenten

- Generische JMS BC
  - Für Pluggability mit beliebigen JMS Providern
- Spezielle MOM BC für MQ Series, MS MQ
- FTP Binding Component
  - Um Input Daten per FTP zu holen bzw. Output per FTP zu verschicken
- Generische JDBC Binding Component
  - Für Polling auf Datenbanken bzw. Write in Datenbanken
- 3rd Party Komponenten für CICS, CORBA, IMS,...

Future: Composite Application Manager



System Architecture Overview

Zusammenfassung

- Open ESB 2.0 zeigt die Leistungsfähigkeit und Vorteile der Java Business Integration Spezifikation
  - Plug & Play für Integrationskomponenten
- JBI & Java EE in Kombination ermöglichen die einfache und portable Entwicklung von Composite Applications
- Die Entwicklung als Open Source ermöglicht und fördert die Integration und Entwicklung von 3rd Party Erweiterungen
- Tooling ist essentiell für rasche Service Entwicklung

Weitere Informationen

- Open ESB Projekt
  - https://open-esb.dev.java.net/
- Verfügbare Komponenten
  - https://open-esb.dev.java.net/Components.html
- Tutorials
  - https://open-esb.dev.java.net/Documents.html
- GlassFish
  - https://glassfish.dev.java.net/

DEMOS, DEMOS, DEMOS

Projekt Open-ESB

Business Integration mit Open Source

Daniel Adelhardt

Software & Java Architekt

Sun Microsystems GmbH