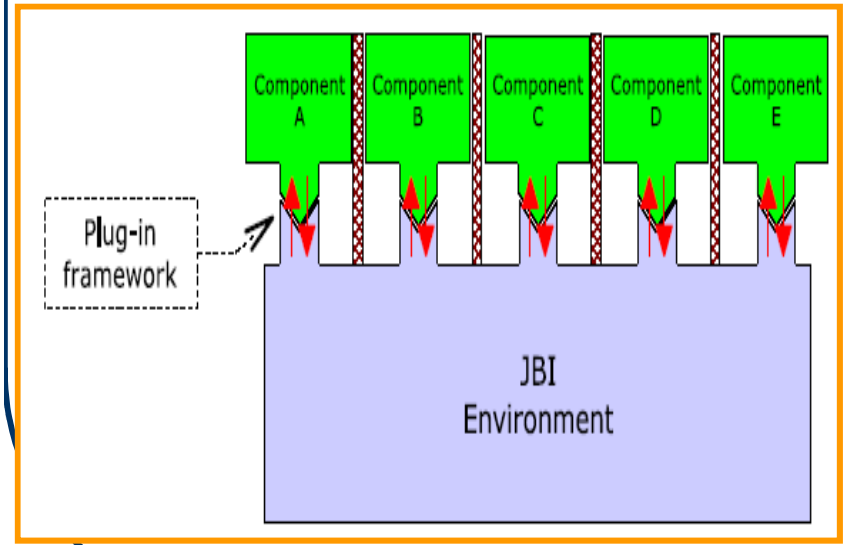
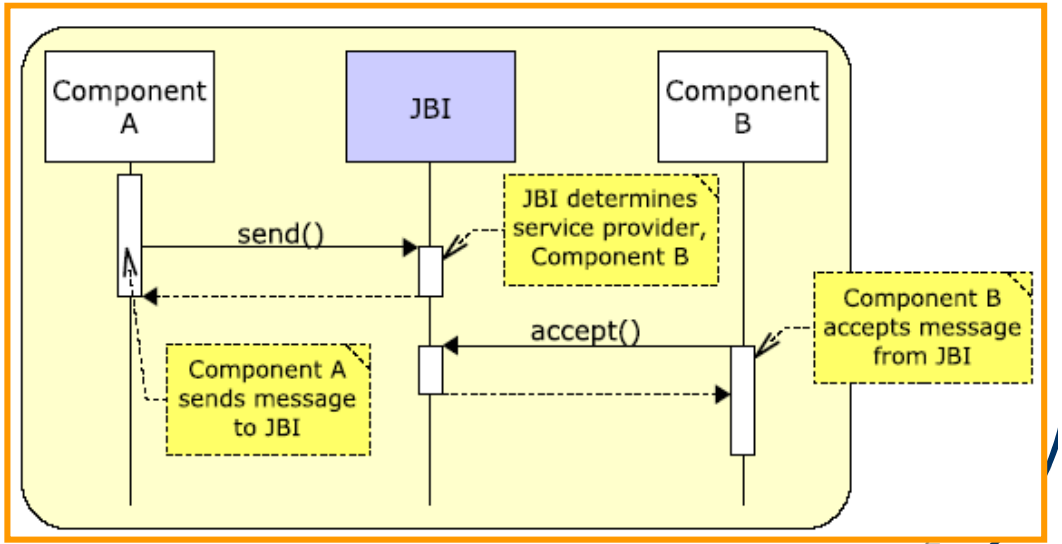
Message Exchange Patterns in JBI

JBI

- **Java Business Integration (JBI)** is a *specification* developed under the Java Community Process (**JCP**) for an approach to implementing a service-oriented architecture (SOA).
  - Creating a standards-based architecture for integration solutions
  - A suitable standard technology instead of proprietary vendor solution

- JBI defines an architecture that allows the construction of integration systems from **plug-in** components, that interoperate through the method of mediated **message exchange**.

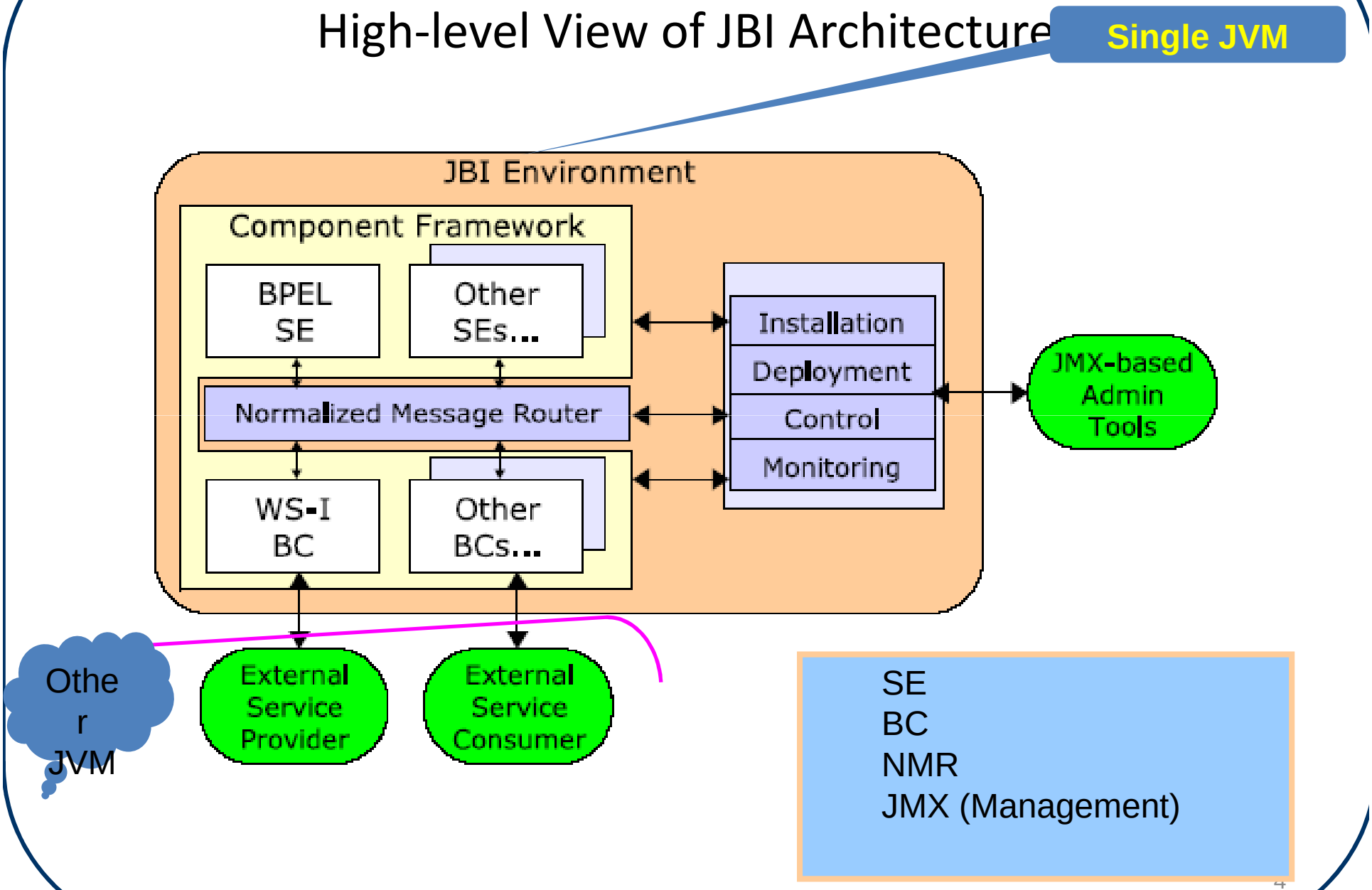




JBI

- Java Business Integration
- JBI is a messaging-based plug-in architecture
- Defines an approach to implement a SOA and underlying structure supporting WS communications
- JBI allows anyone to create JBI compliant integration plug-in component and integrate them dynamically into the JBI infrastructure
- JBI is an architecture for integration systems specifying plug-in components that interoperate by exchanging messages
  - This decoupling increases flexibility because each component needs to know how to interact with the JBI bus only and not with *n* number of other components.
- JBI components provide services, consume services, or sometimes both.

High-level View of JBI Architecture



JBI

- Key pieces of the JBI environment
  - Service Engines (SE)
    - SEs provide business logic and transformation services
  - Binding Components (BC).
    - BCs provide connectivity for applications that are external to the JBI.
  - Normalized Message Router (NMR)
    - Provides mediated message exchange between components
  - JBI runtime environment (JBI meta container)
- The separation of business and processing logic from communication logic makes the implementation of components much less complex.

JBI

- Service engines and binding components interact with JBI in two ways:
  - **SPIs:** Interfaces implemented by a binding or engine
  - **APIs:** Interfaces exposed to bindings or engines by the framework
- An external service consumer sends a service request across a specific protocol and transport to a binding component.
- The binding component converts the request to a normalized message.
- The binding component then creates a message packet called a message exchange (ME) and sends it across the binding component's delivery channel to the NMR for delivery to a service provider.

JBI

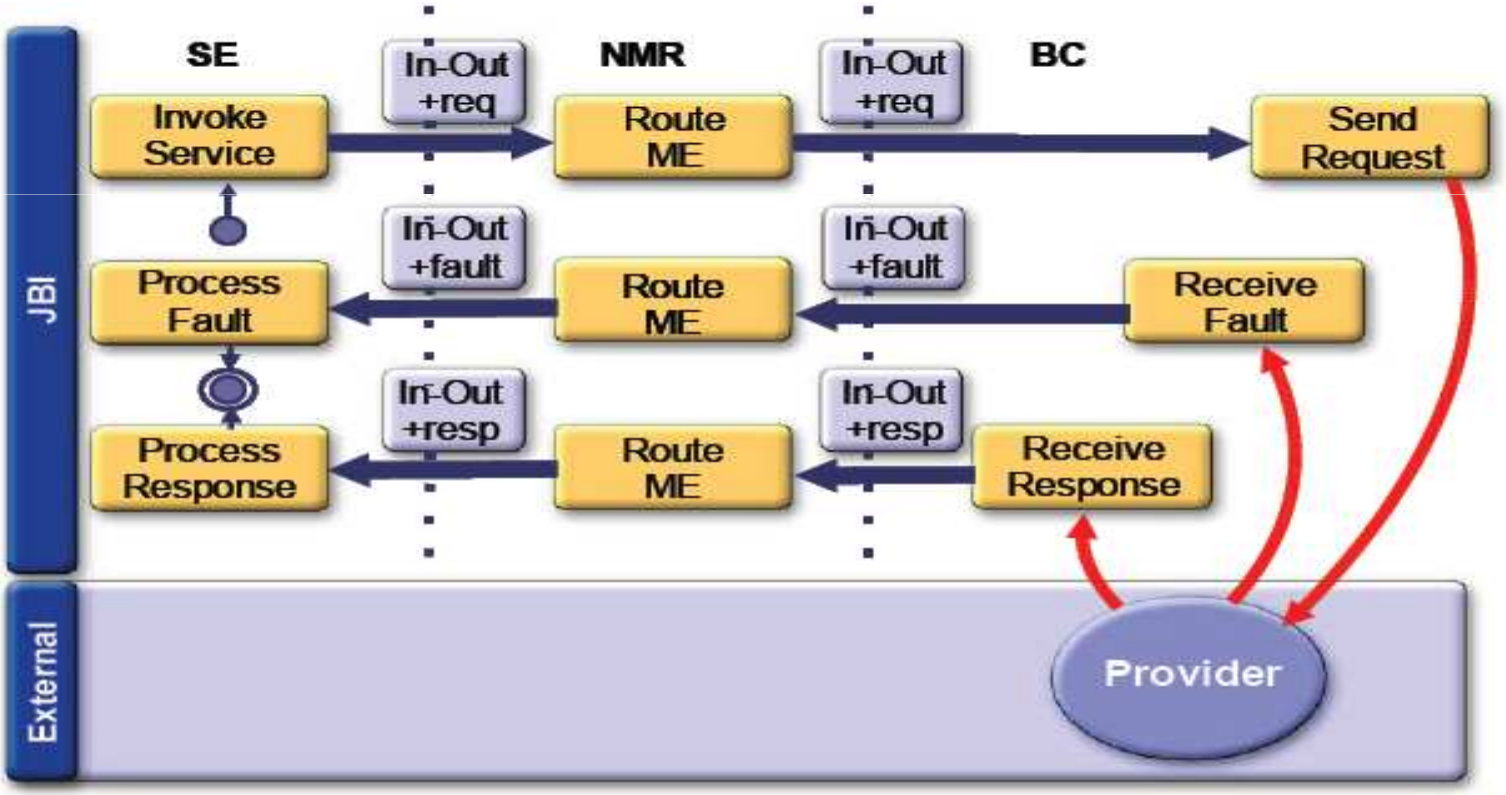
- JBI integrates with legacy systems, binary transports, document-oriented transports, and RPC (remote procedure call) systems.
- Message normalization is the process of mapping context-specific data to a context-neutral abstraction to transport data in a standard format.
- All messages handled by the NMR are normalized.
- The NMR APIs include:
  - JBI Message API
  - JBI Service API
  - JBI Message Exchange Factory API
  - Service Description SPI
  - Message Exchange Patterns API
  - Endpoint Reference API

JBI - NMR

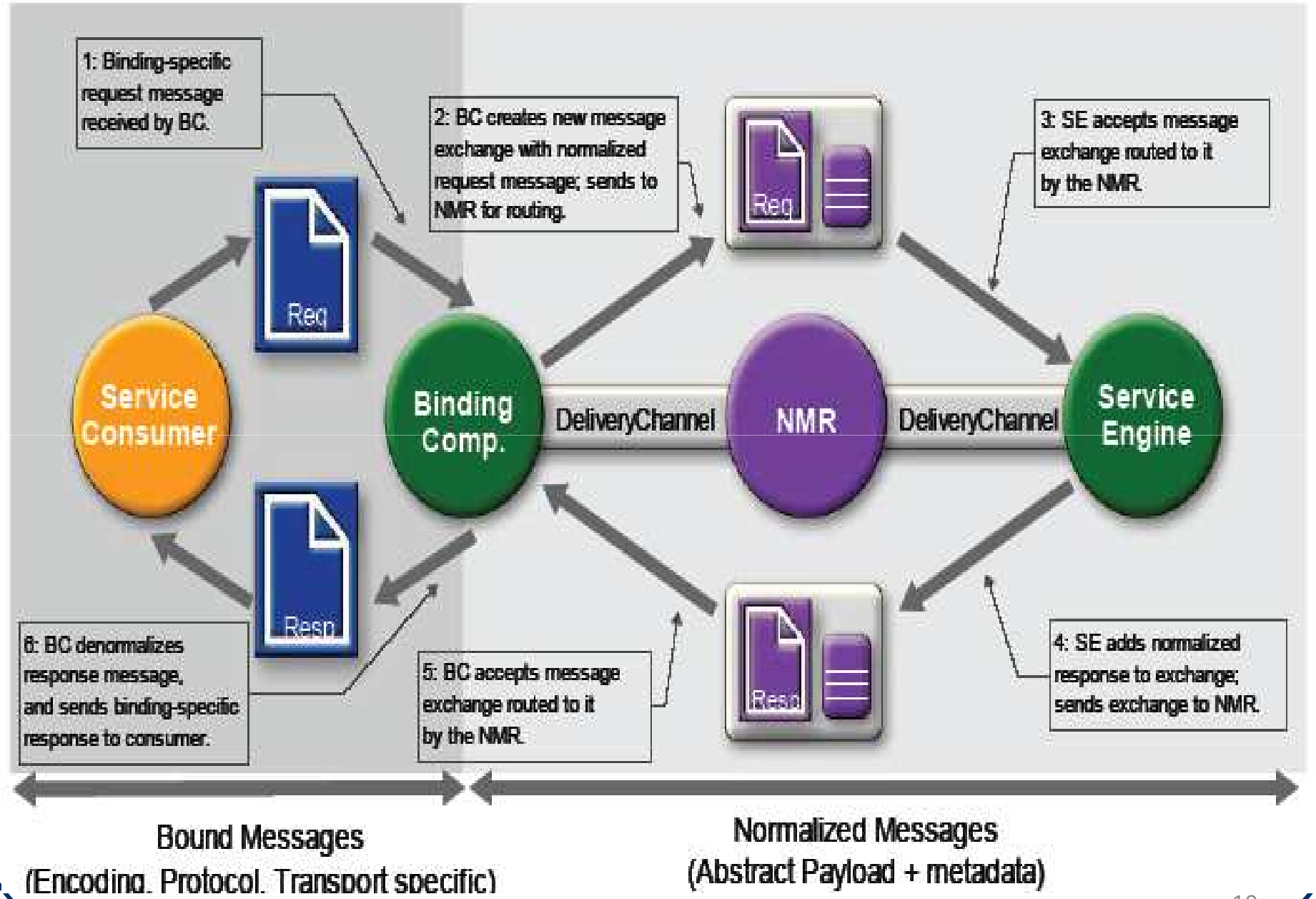
- **Normalized Message**
  - Abstract Message (payload) + Message Properties (metadata)
  - WSDL Abstract Message
    - WSDL interface operation message definition
  - Properties (metadata)
    - Protocol-supplied context information
    - Security tokens
    - Transaction information
    - Data other components may recognize
- Messages flowing into the JBI, via binding components, are translated into a normalized (neutral) format, then routed to their destination.
- Prior to final delivery, the normalized message is translated into the appropriate format for the recipient.
- A message can be routed through several JBI components depending on what processing is needed.

Normalized Message Router

- Mediated Message Exchange
- Message Exchange Pattern
  - Support for simple communications primitives
  - Message exchange patterns are defined from the provider's perspective



Normalized Message Router



Message Exchange

- A Message Exchange (ME) serves as a "**container**" for the normalized messages that are part of a service invocation.

- JBI supports a fixed set of **message exchange patterns**
  - a well-defined sequence of message exchanges between the consumer and the provider.

- A **message exchange pattern (or MEP)** no matter what contents (or type) of the messages themselves. By supporting a limited set of MEPs, the JBI standard ensures that components have a simple, well known interaction model to implement, ensuring **interoperability**.

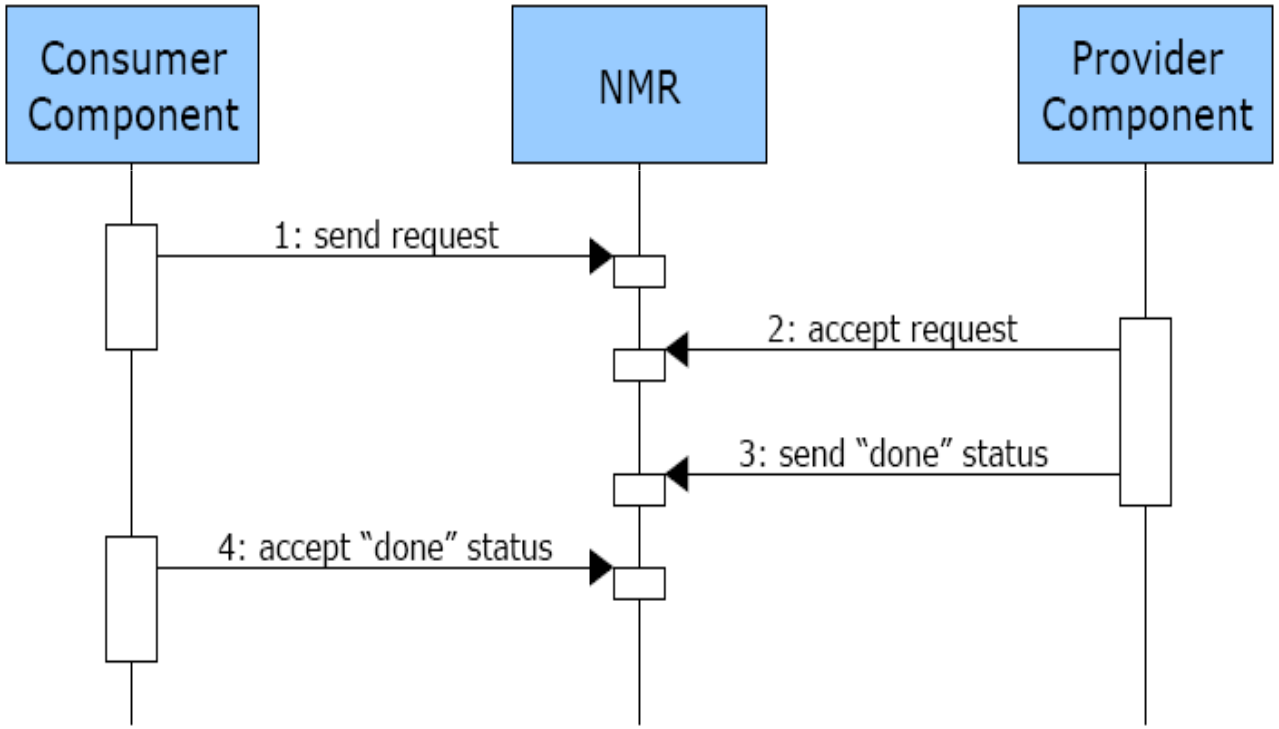
Message Exchange Patterns (MEPs)

- An MEP defines the sequence, direction, and cardinality of all messages that occur in the course of invoking and performing the operation.
- All message exchanges between consumer and provider are mediated by the NMR.
- The components interact with the NMR, using their individual DeliveryChannels
  - Sending **MessageExchange** instance
  - Accepting **MessageExchange** instance
- **MEPs**
  - In-only Message Exchange Pattern
  - Robust In-Only Message Exchange Pattern
  - In-Out Message Exchange Pattern
  - In-Optional-Out Message Exchange Pattern

Service Invocation	Message Exchange Pattern (Provider View)
One-Way	In-Only
Reliable One-Way	Robust In-Only
Request-Response	In-Out
Request Optional-Response	In Optional-Out

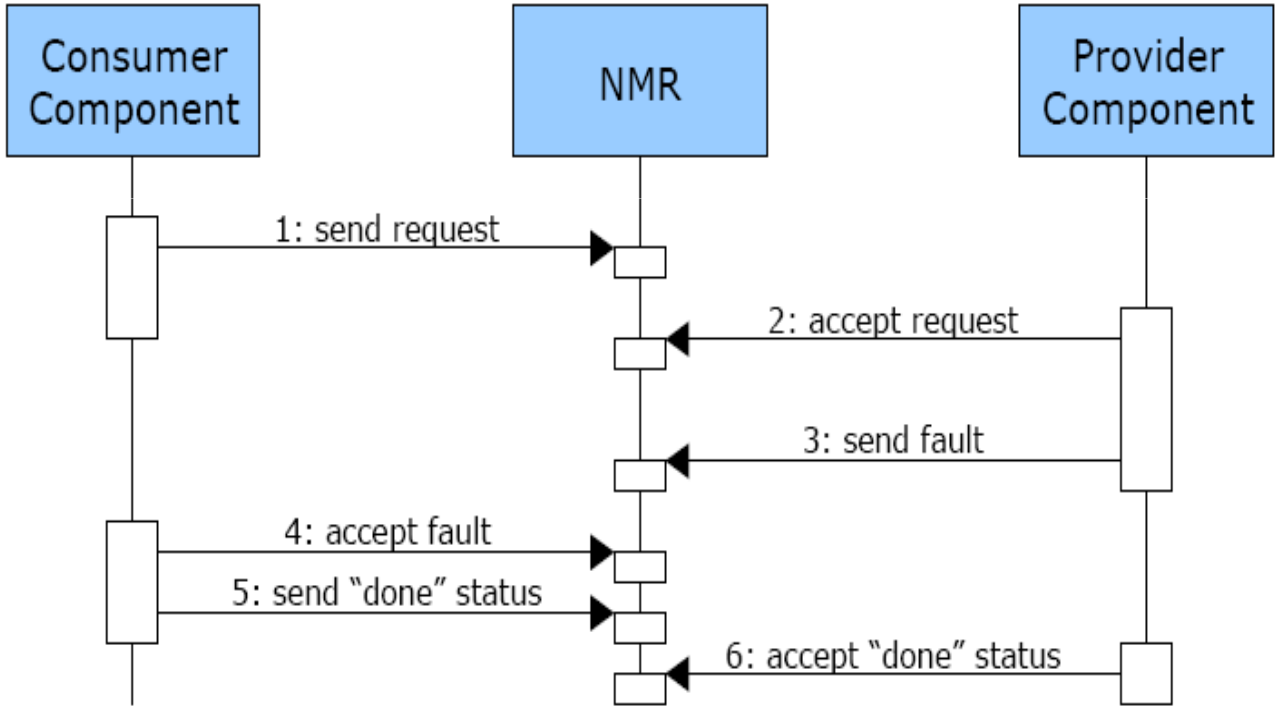
In-only Message Exchange Pattern

- Describing a one-way messaging pattern



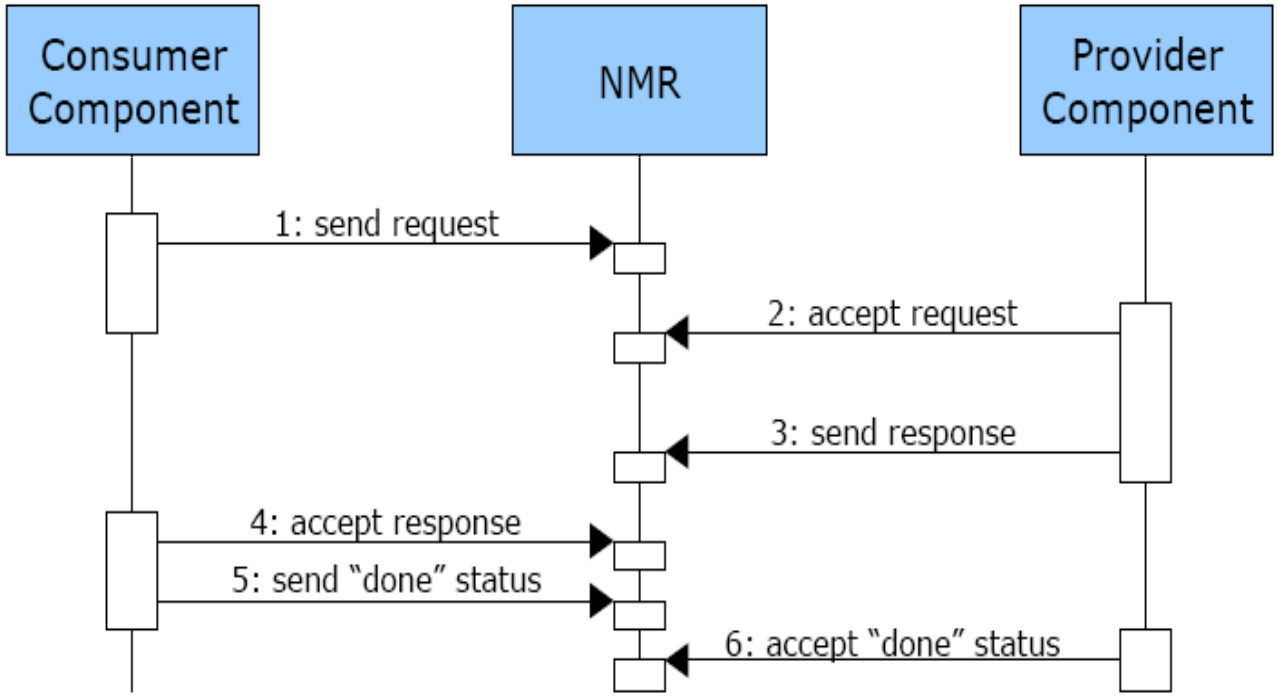
Robust In-Only Message Exchange Pattern (fault scenario)

- Allowing the provider to easily return error responses to in-only message exchange



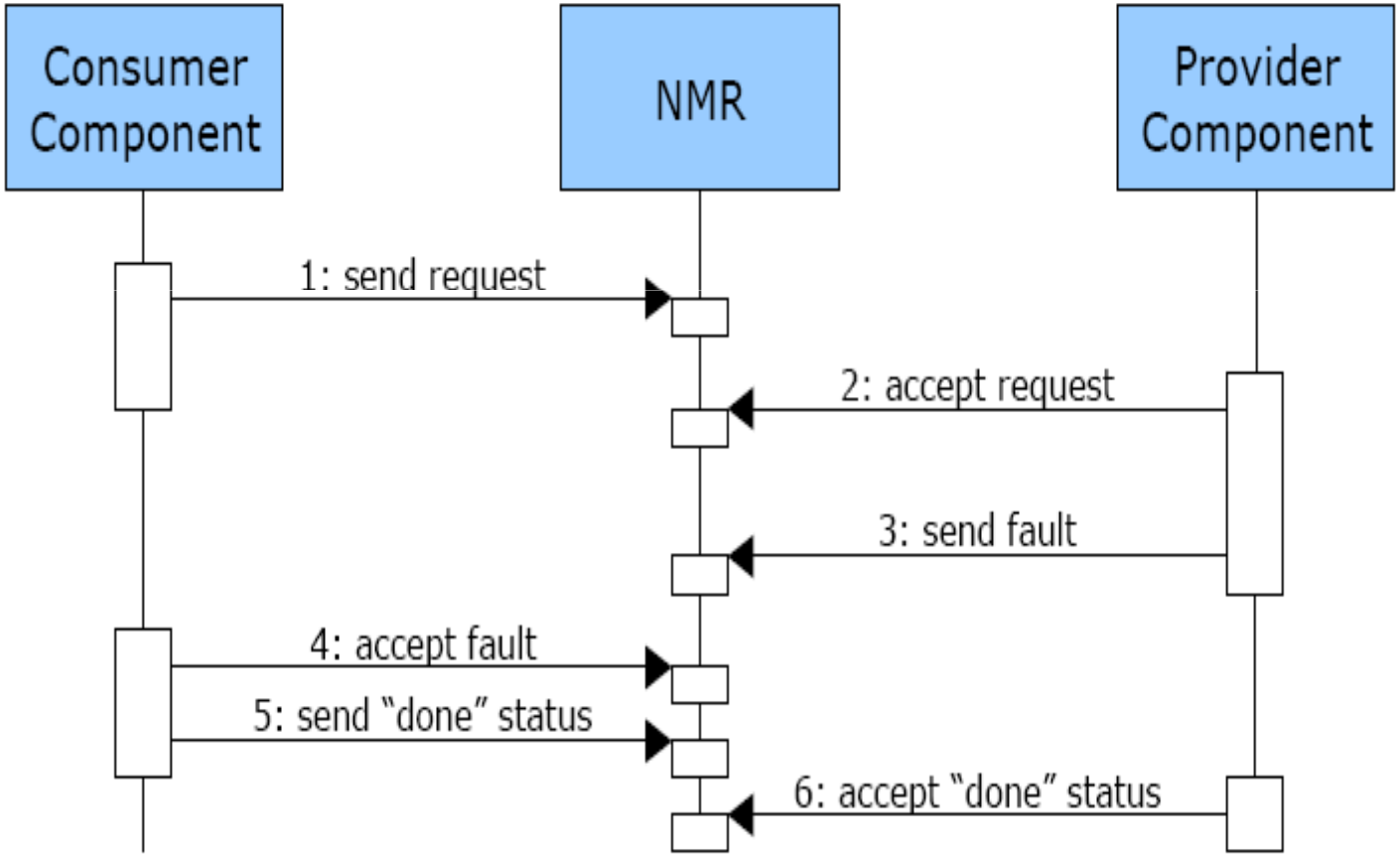
In-Out Message Exchange Pattern

- Normal response



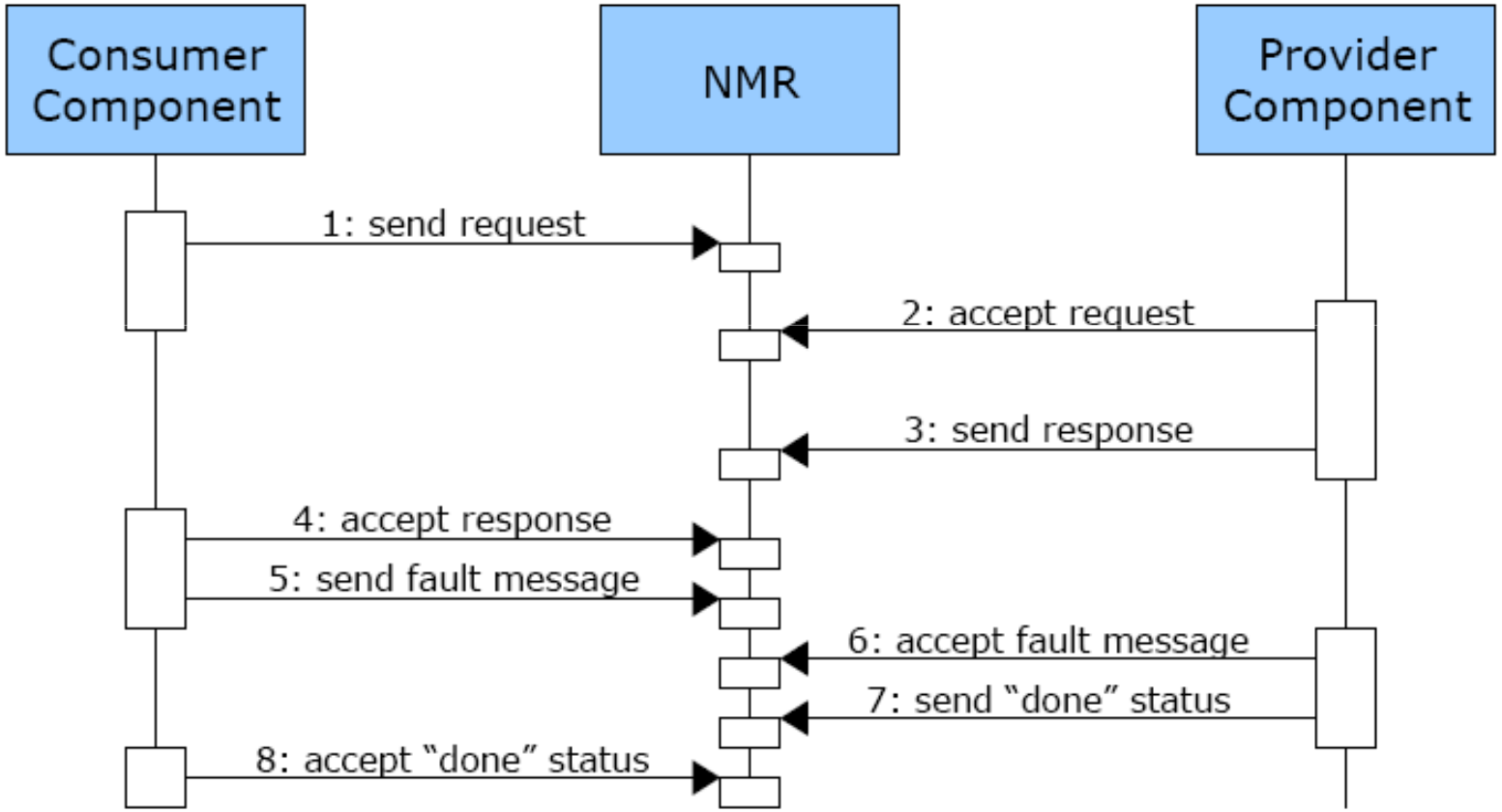
In-Out Message Exchange Pattern (Cont'd)

- Fault response

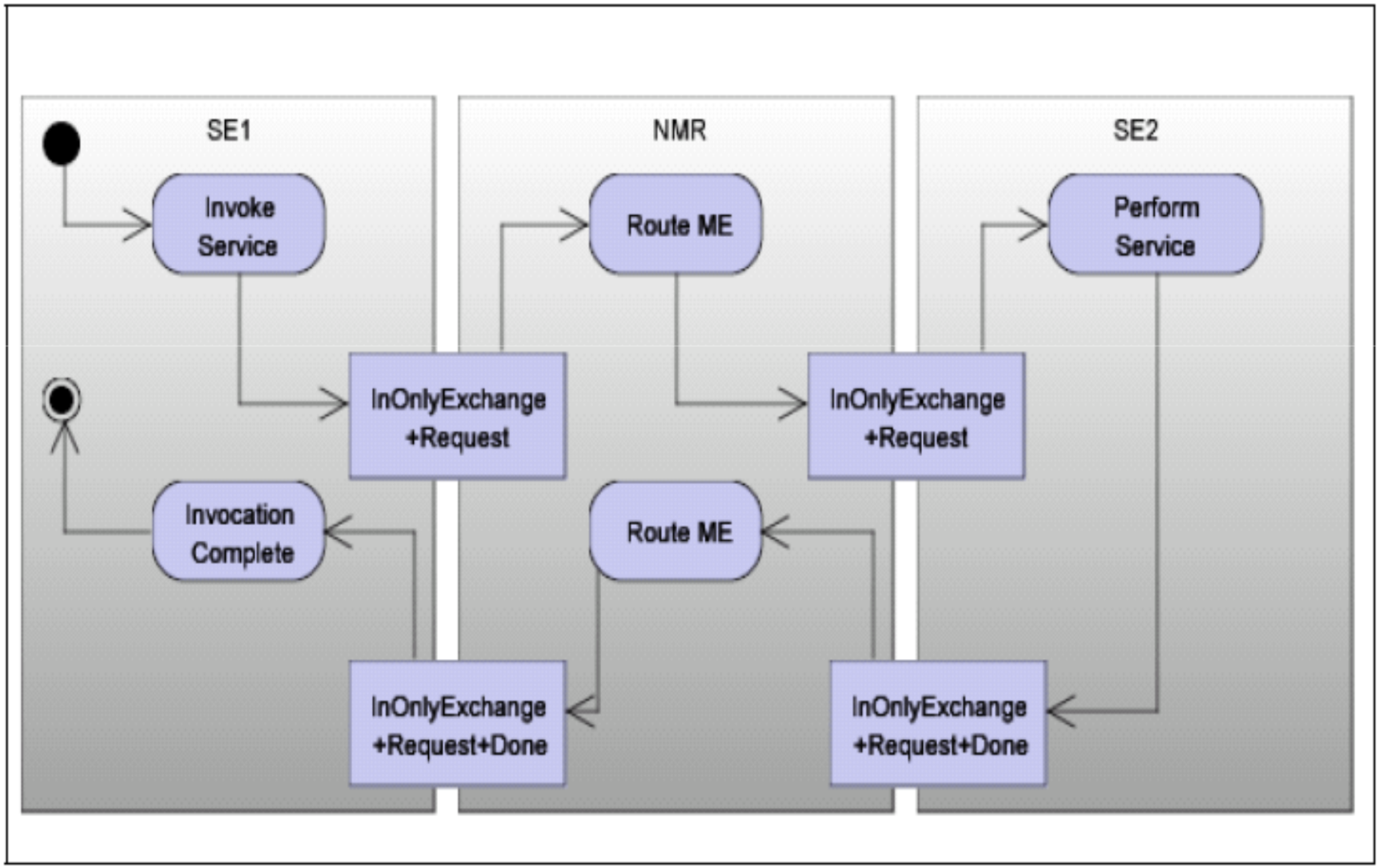


In-Optional-Out Message Exchange Pattern (Cont'd)

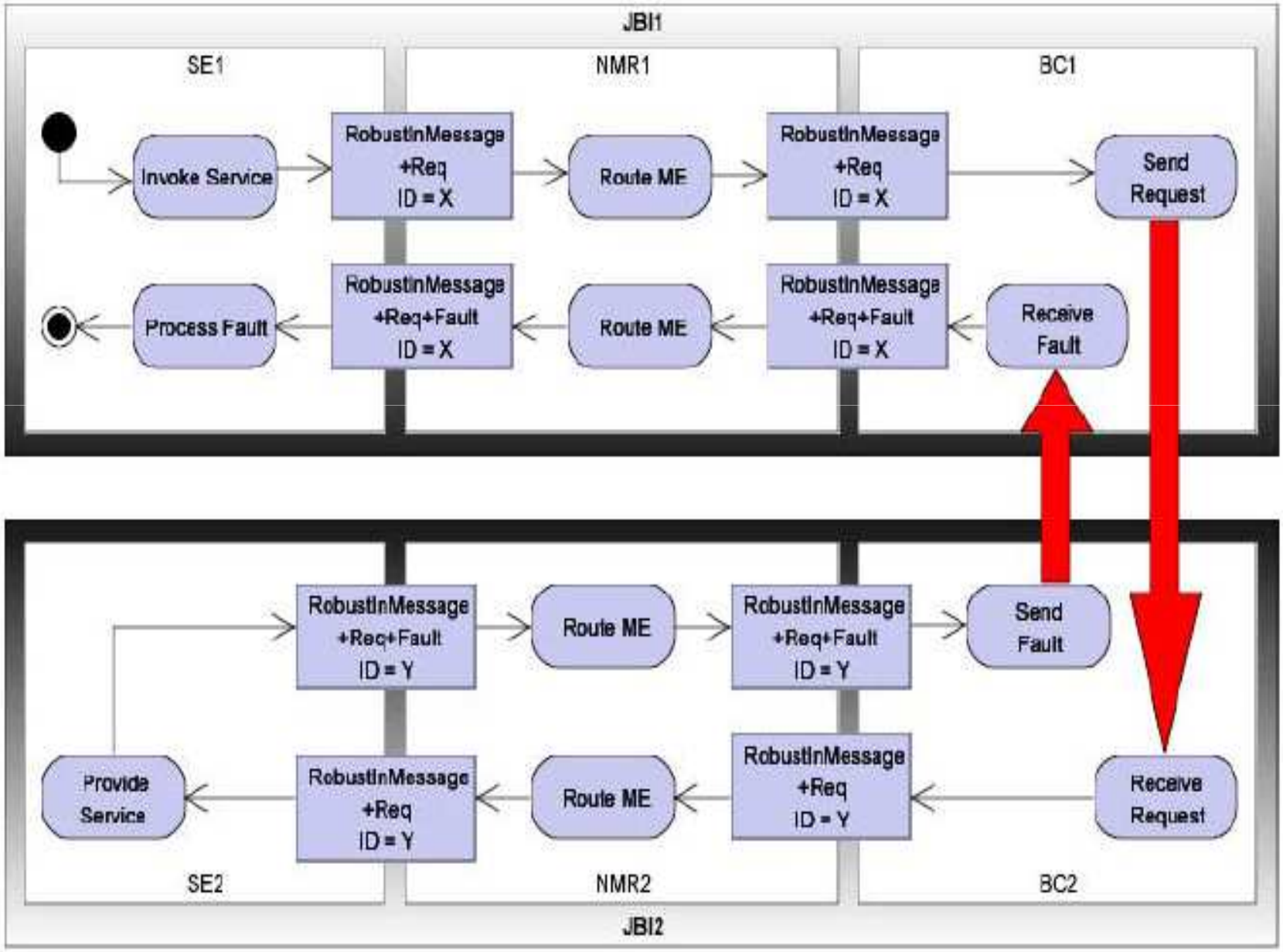
- Normal response 2



One-way Service Invocation Between two Service Engines



SE Invokes Service Provided by Remote JBI Instance: Robust In with Fault



SE Invokes Remote Service