WS-BPEL 2.0

Language Constructs Overview

Dieter König

IBM Senior Technical Staff Member

dieterkoenig@de.ibm.com

WS-BPEL Language Constructs

- WS-BPEL process definition
- Partner links
- Variables
- Variable properties
- Correlation sets
- Basic and structured activities
- Scopes
- Compensation handling

WS-BPEL Process Definition

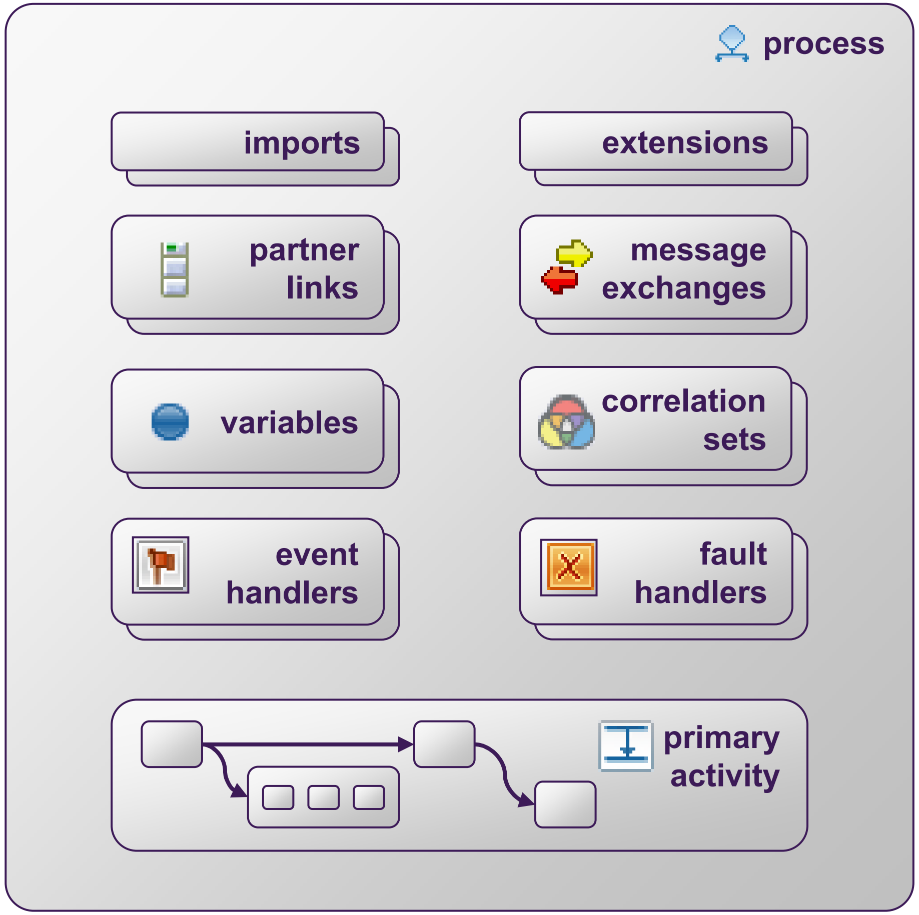
Declare dependencies on external XML Schema or WSDL definitions

Relationships that a WS-BPEL process will employ in its behavior

Data holding state of a business process or exchanged with partners

Concurrently process inbound messages or timer alarms

Perform the process logic – any number of activities may be recursively nested



process

imports

extensions

partner links

message exchanges

variables

correlation sets

event handlers

fault handlers

primary activity

Declare namespaces of WS-BPEL extension attributes and elements

Relationship between inbound and reply message activities

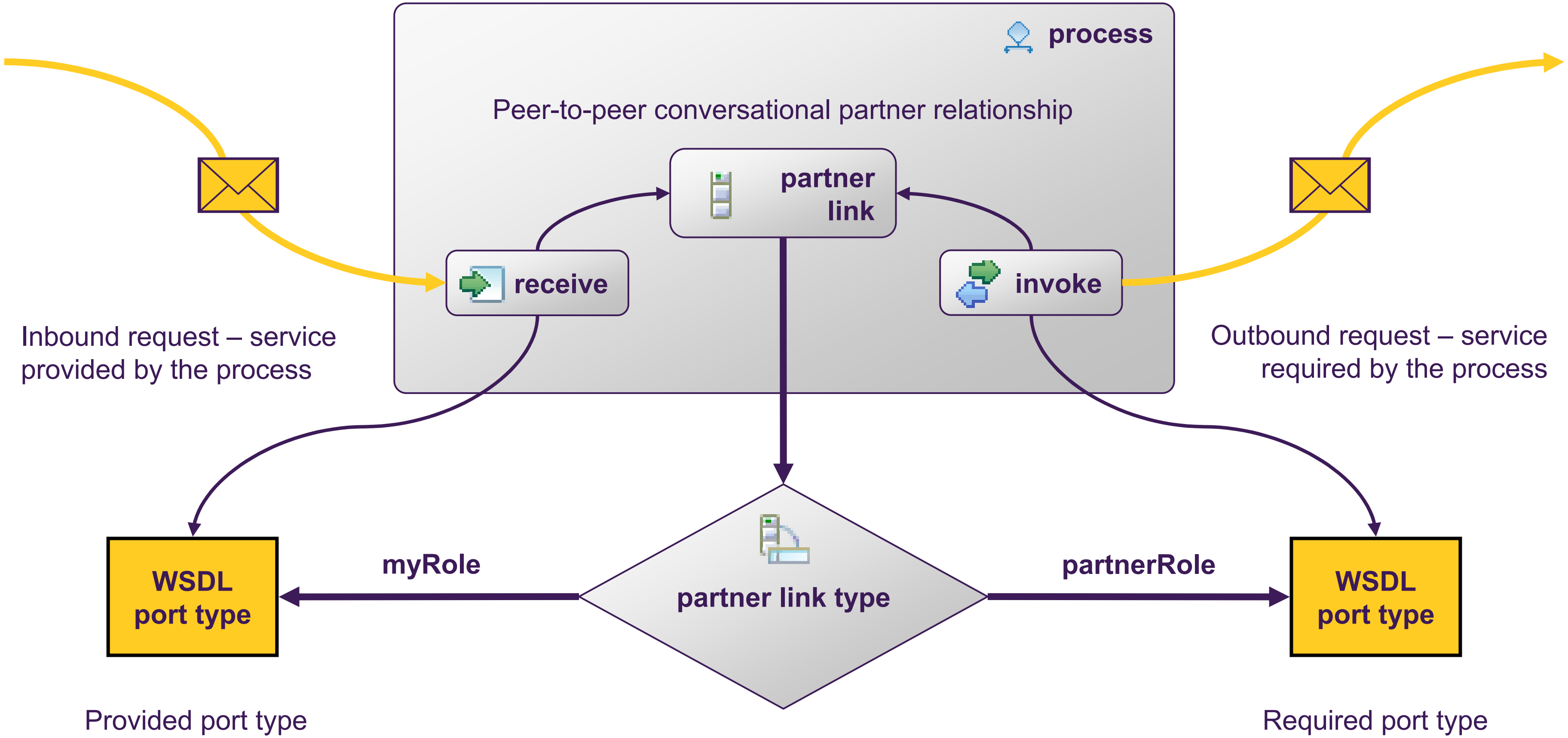
Application data fields that together identify a conversation

Deal with exceptional situations in a process

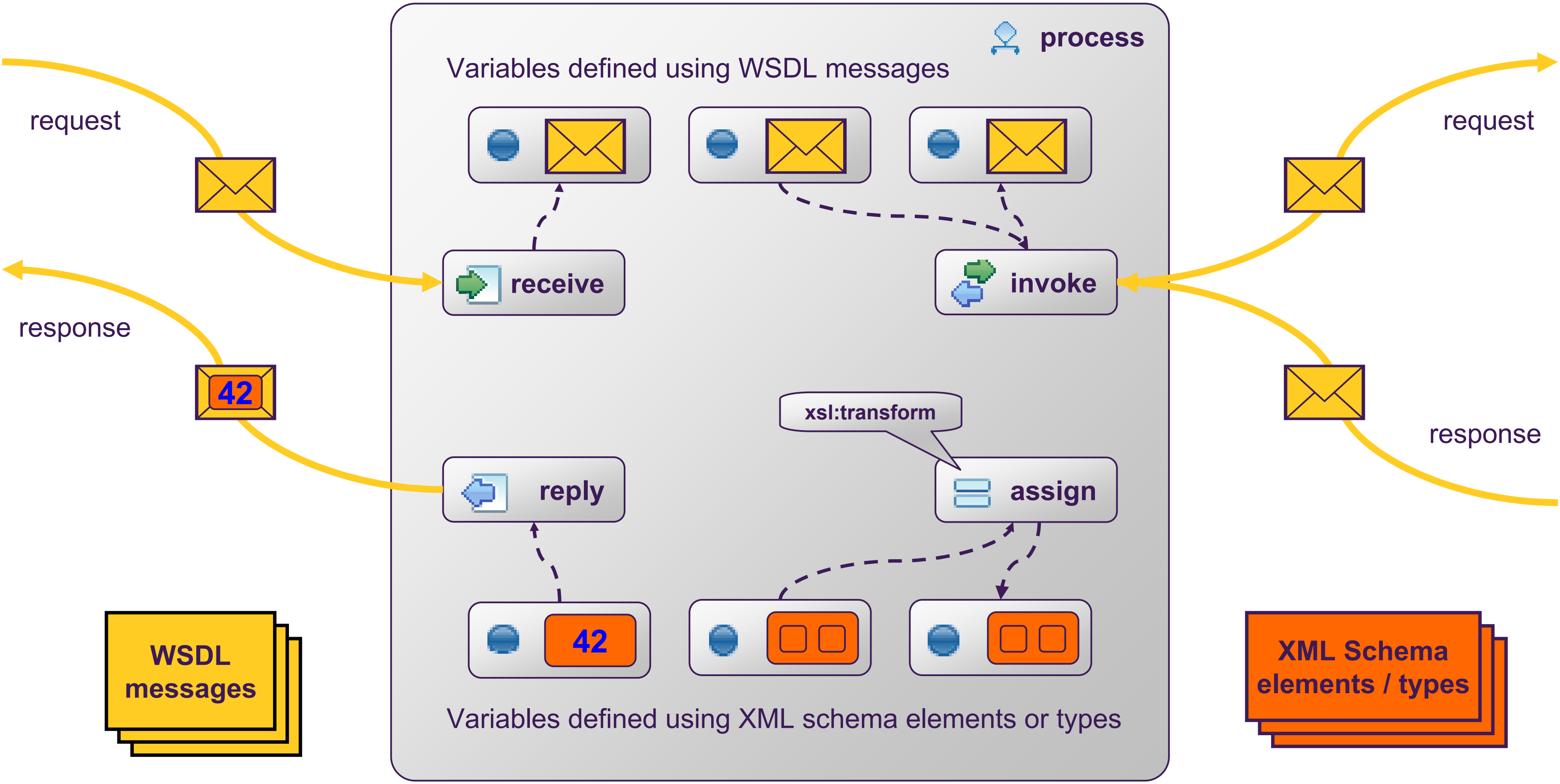
WSDL definitions

XML schemas

Partner Links

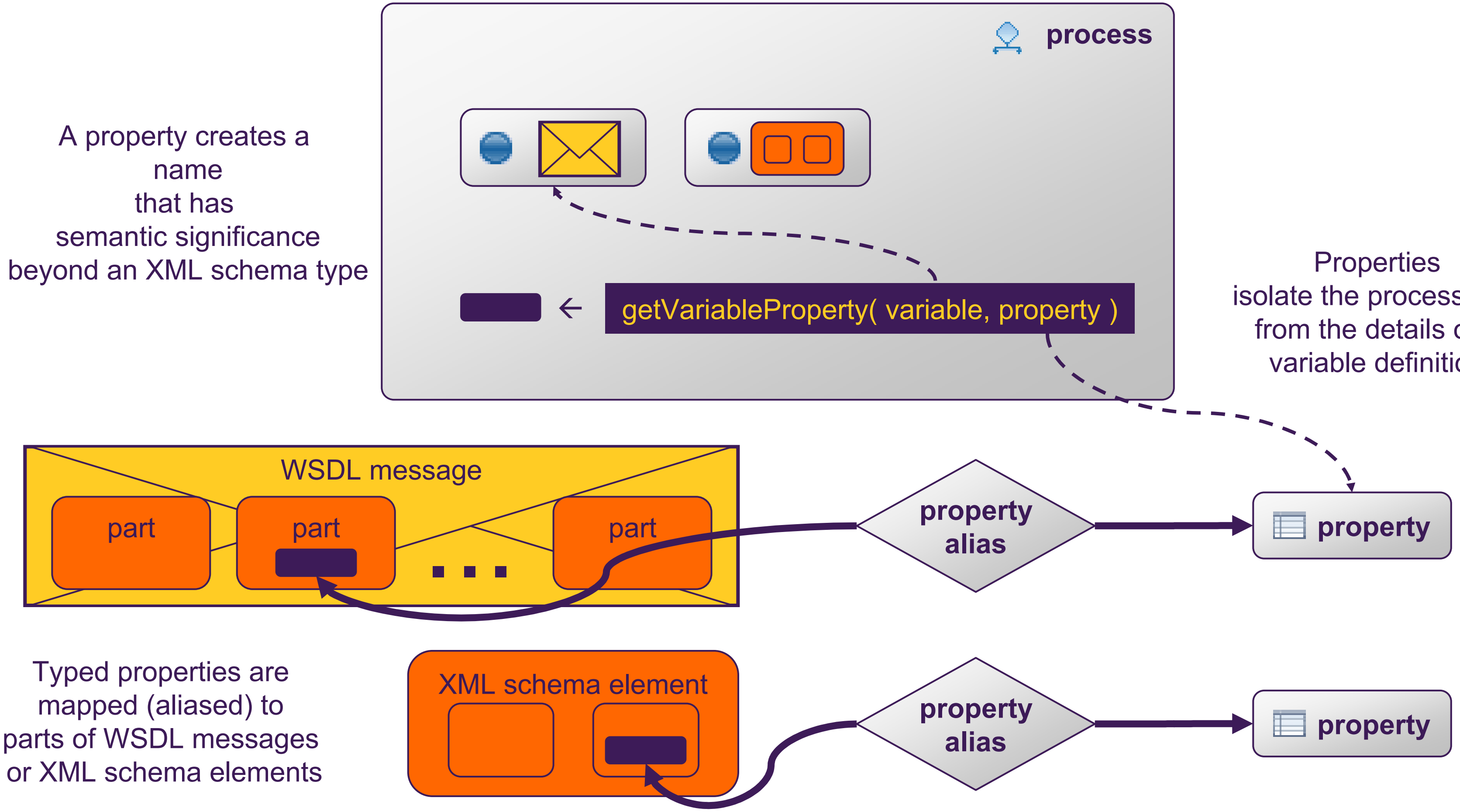

process
Peer-to-peer conversational partner relationship
partner link
receive
invoke
Inbound request – service provided by the process
Outbound request – service required by the process
WSDL port type
myRole
partner link type
partnerRole
WSDL port type
Provided port type
Required port type

Variables


process
Variables defined using WSDL messages
request
response
42
receive
invoke
request
response
xsl:transform
reply
assign
42
WSDL messages
XML Schema elements / types
Variables defined using XML schema elements or types

Variable Properties

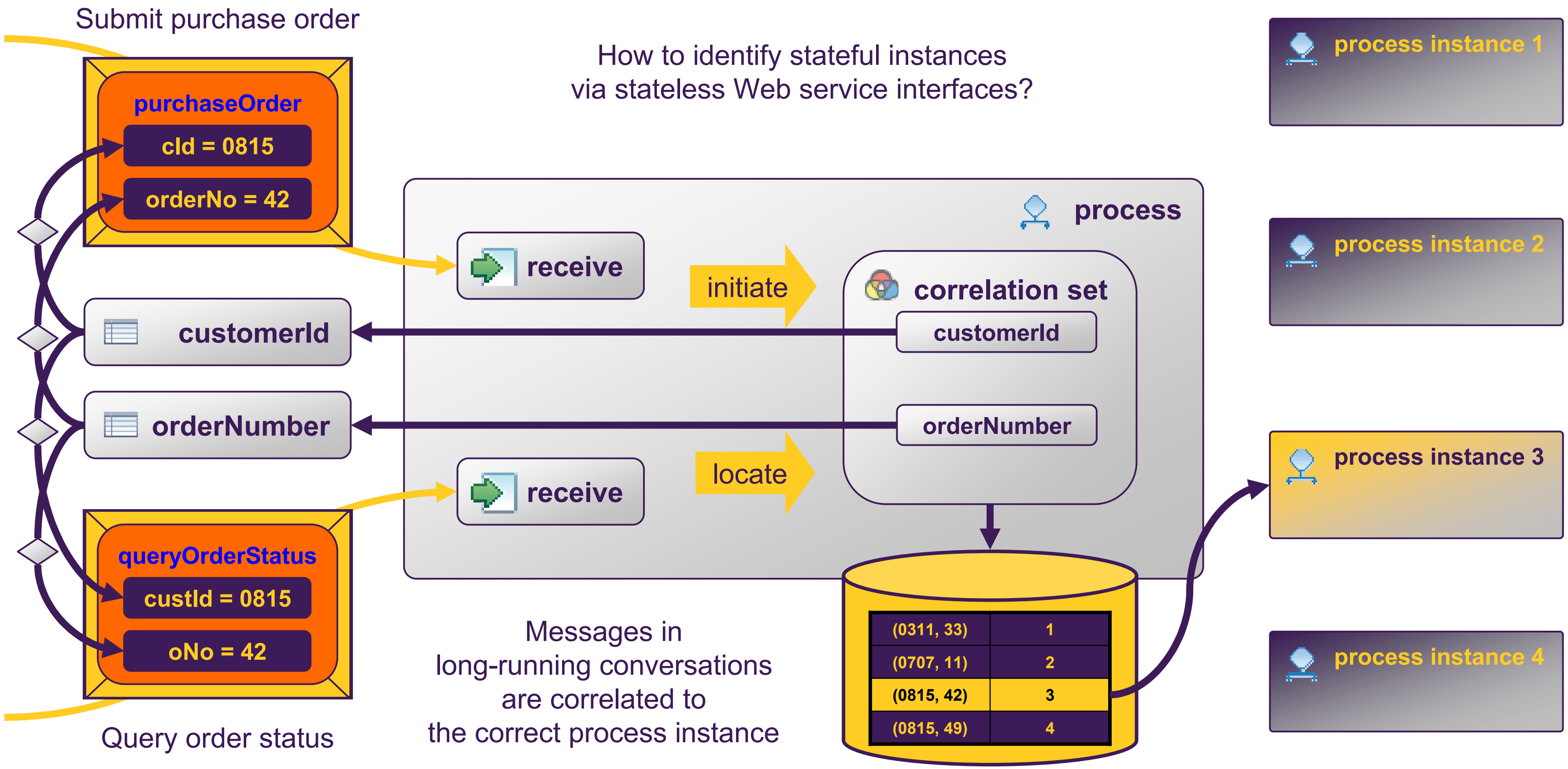
A property creates a
name
that has
semantic significance
beyond an XML schema type



Properties
isolate the process logic
from the details of a
variable definition

Typed properties are
mapped (aliased) to
parts of WSDL messages
or XML schema elements

Properties and Correlation Sets



How to identify stateful instances
via stateless Web service interfaces?

(0311, 33)	1
(0707, 11)	2
(0815, 42)	3
(0815, 49)	4

Messages in
long-running conversations
are correlated to
the correct process instance

Basic Activities

Do a blocking wait for a matching message to arrive / send a message in reply

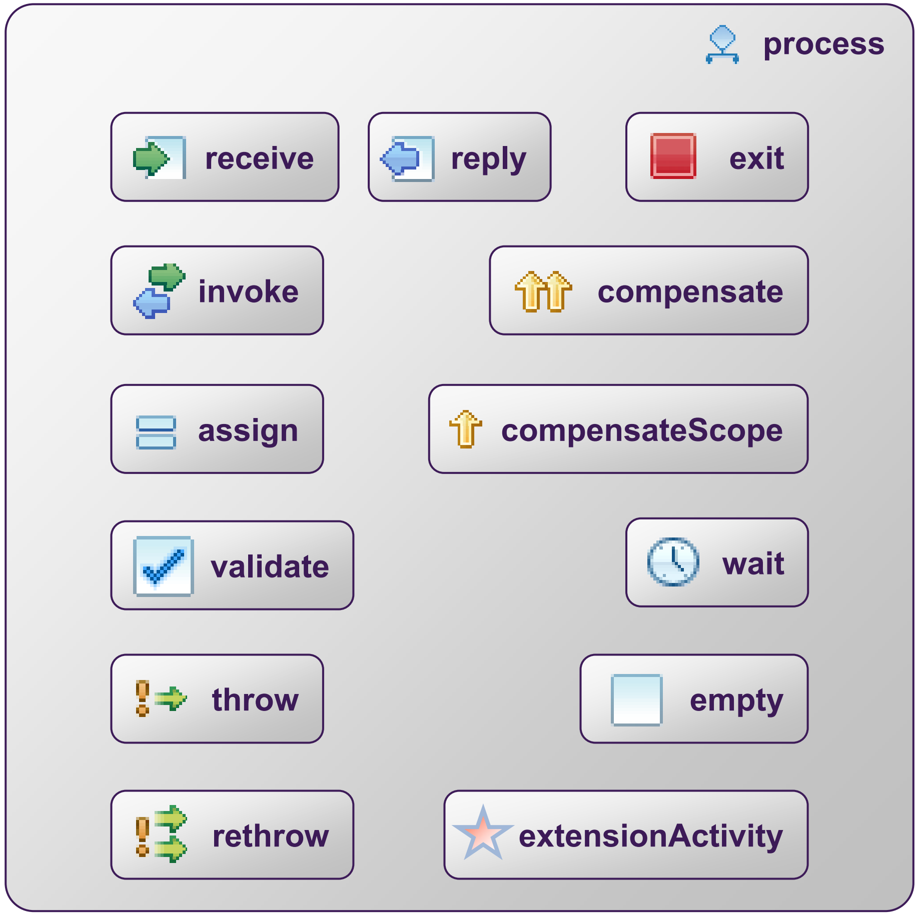
Invoke a one-way or request-response operation

Update the values of variables or partner links with new data

Validate XML data stored in variables

Generate a fault from inside the business process

Forward a fault from inside a fault handler



process

receive

reply

exit

invoke

compensate

assign

compensateScope

validate

wait

throw

empty

rethrow

extensionActivity

Immediately terminate execution of a business process instance

Invoke compensation on all completed child scopes in default order

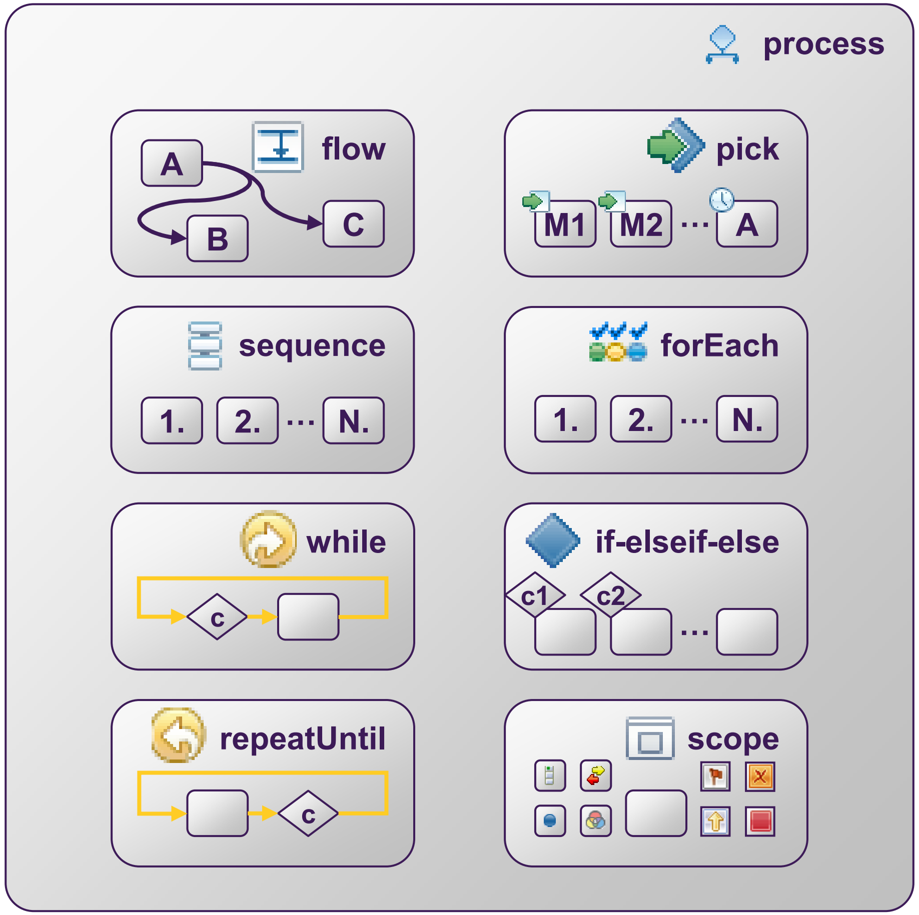
Invoke compensation on one completed child scope

Wait for a given time period or until a certain time has passed

No-op instruction for a business process

Wrapper for language extensions

Structured Activities



Contained activities are executed in parallel, partially ordered through control links

Contained activities are performed sequentially in lexical order

Contained activity is repeated while a predicate holds

Contained activity is repeated until a predicate holds

Block and wait for a suitable message to arrive (or time out)

Contained activity is performed sequentially or in parallel, controlled by a specified counter variable

Select exactly one branch of activity from a set of choices

Associate contained activity with its own local variables, partner links, etc., and handlers

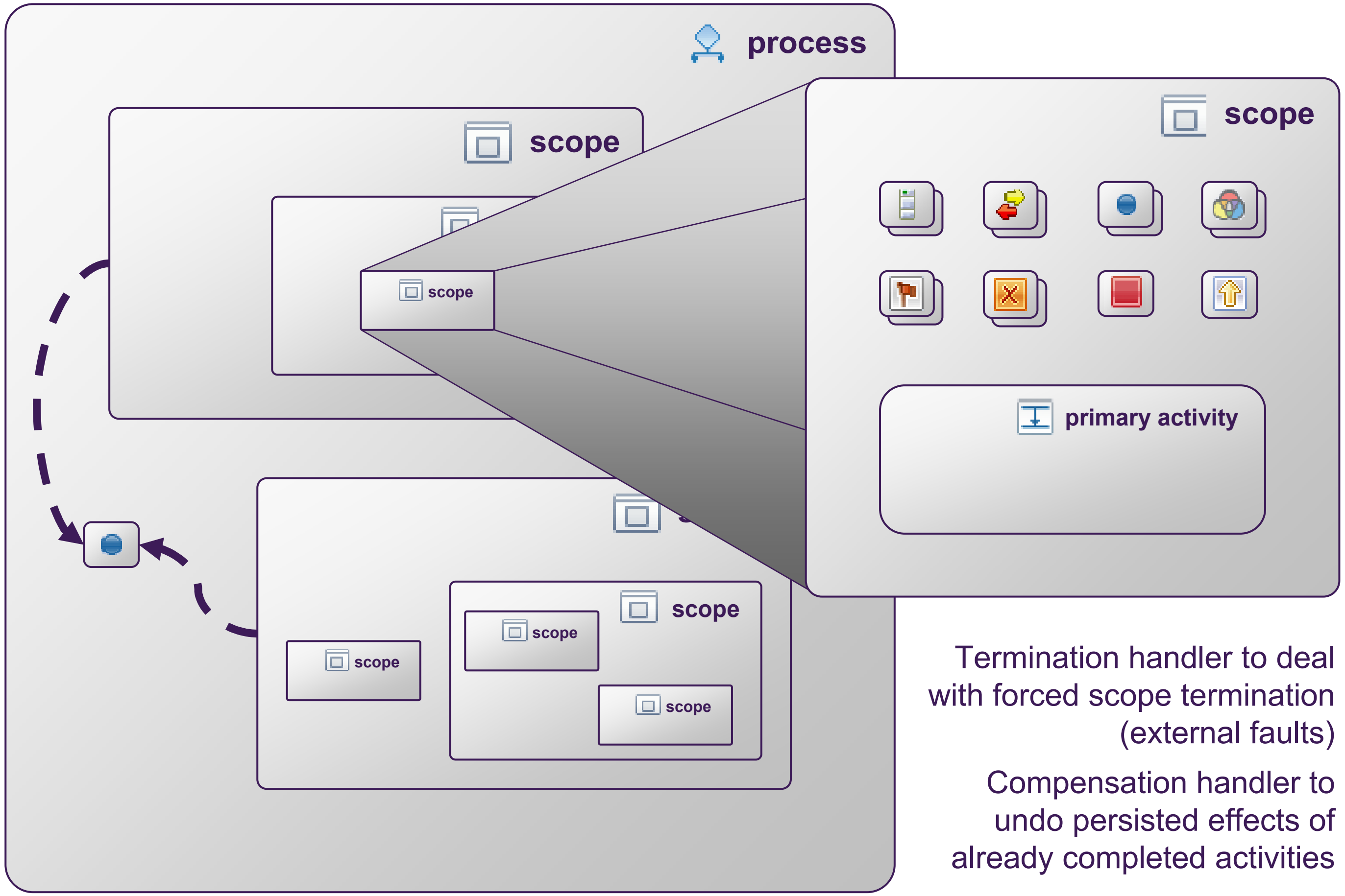
Scopes

Scopes provide a context which influences the execution behavior of its enclosed activities

Local declarations – partner links, message exchanges, variables, correlation sets

Local handlers – event handlers, fault handlers, a termination handler, and a compensation handler

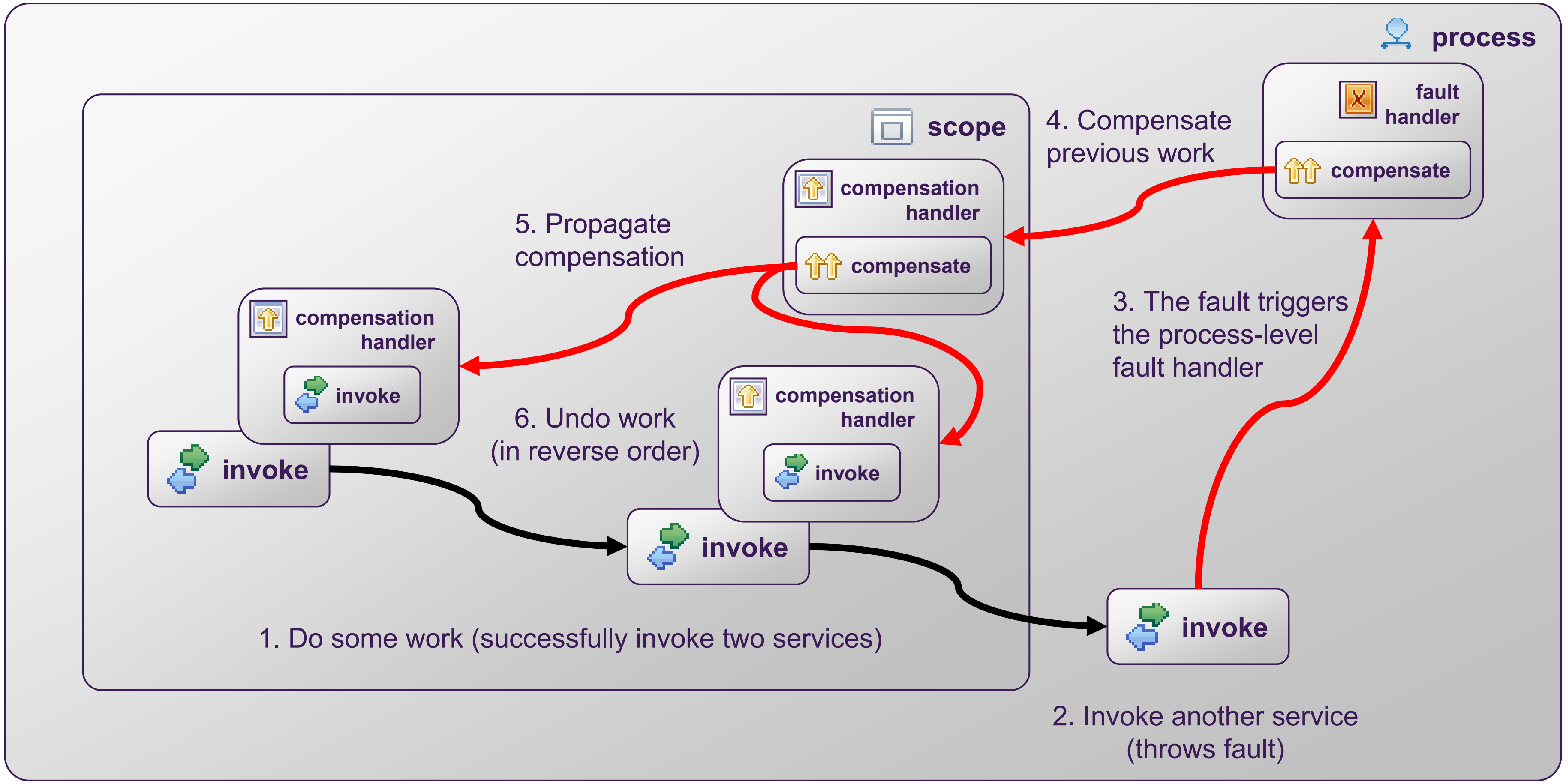
Isolated scopes provide control of concurrent access to shared resources



Termination handler to deal with forced scope termination (external faults)

Compensation handler to undo persisted effects of already completed activities

Compensation Handling



process
fault handler
compensate
scope
4. Compensate previous work
compensation handler
compensate
5. Propagate compensation
compensation handler
invoke
compensation handler
invoke
3. The fault triggers the process-level fault handler
6. Undo work (in reverse order)
invoke
invoke
invoke
1. Do some work (successfully invoke two services)
2. Invoke another service (throws fault)