EA IT - Studies General Purpose - Technical Event Data Model

Summary

Stakeholders involved

Role	Person
Product owner	
IT Area Lead	
Enterprise architect	
Domain architect	
Information architect	
Business analyst	
Feature engineer	

Objective of the study

Define a standard data model for a subcategory of Technical Events which enables *Event Driven Conversation Patterns* between parties.

Introduction

As applications grow in scale and complexity, the way we design and implement these communications can significantly impact system performance, scalability and maintainability.

The **Event Data Model** provides a framework for structuring and managing these interactions, particularly in distributed and microservices-based systems.

This document aims to define the **Event Data Model**, its components, and its implementation in various architectural patterns.

By understanding this model, developers and architects can make informed decisions about how to structure their system's communications, leading to a more robust, scalable and efficient applications.

Glossary

First of all we need to create a common ground about the definitions.

What is an Event?

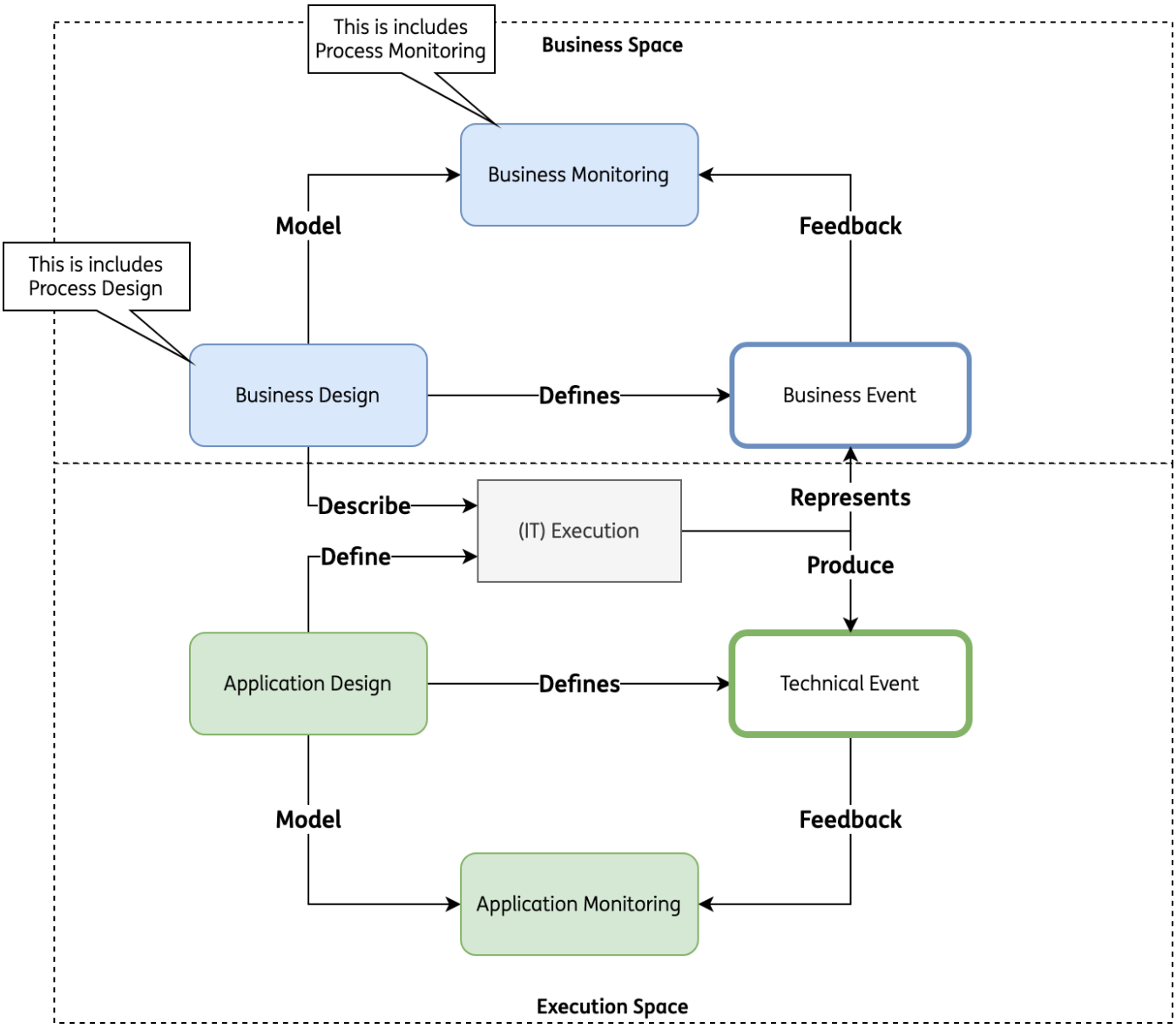
Event is a message which contains or depict something that occurs. In general it helps to decouple processes and systems by providing an asynchronous and data driven integration, providing a loosely coupled and more flexible architecture, where an evolving and heterogeneous technical landscape does not inhibit to achieve responsiveness and integration.

Business vs System Events

Events can be splitted into two macro categories (ref: RA Business Event Categories Endorsed v02): **Business Event** and **System Event**

Event Type	Definition
Business Event	A business event is an identifiable, significant happening in a business scenario.
System Event	is a point in the execution of a program at which an identifiable computing task takes place.

The following picture depicts how the two event categories belongs to different "worlds"



In particular we are going to focus on a subcategory of System Events: the System Process Events.

System Process Events

this kind of events enables the inter-processes conversation patterns and they can be seen as an API endpoint, in a parallel with the HTTP/Rest communication

Synchronous vs. Asynchronous Interaction

Before exploring the **System Process Event Data Model,** it's essential to understand the fundamental paradigms of intercomponent communication: synchronous and asynchronous interactions. This understanding forms the foundation upon which the **Event Data Model** is built.

Synchronous Interaction

In synchronous communication, which occurs typically via microservices, the sender waits for the receiver's response before proceeding. This creates a direct, real-time interaction.

Pros:

- Immediate feedback
- Easier to manage and reason about
- Simplifies error handling
- Optimized for low latency interaction

Cons:

- Can lead to blocking and reduced performance
- Limited scalability
- Potentials for timeouts in log running operations

Asynchronous Interaction

In asynchronous communication, which occurs works by message exchange between components, the sender doesn't wait for the receiver response and can continue processing other tasks.

Pros:

- Improved performance and responsiveness
- Better scalability
- Supports log-running operations without blocking

Cons:

- More complex to implement and reason about
- Require additional
- Requires additional mechanisms for error handling and state management

The need for a well defined Data Model

The distinction between synchronous microservices and asynchronous event-driven systems highlights the critical need for a well-defined data model.

This model serves two primary purposes:

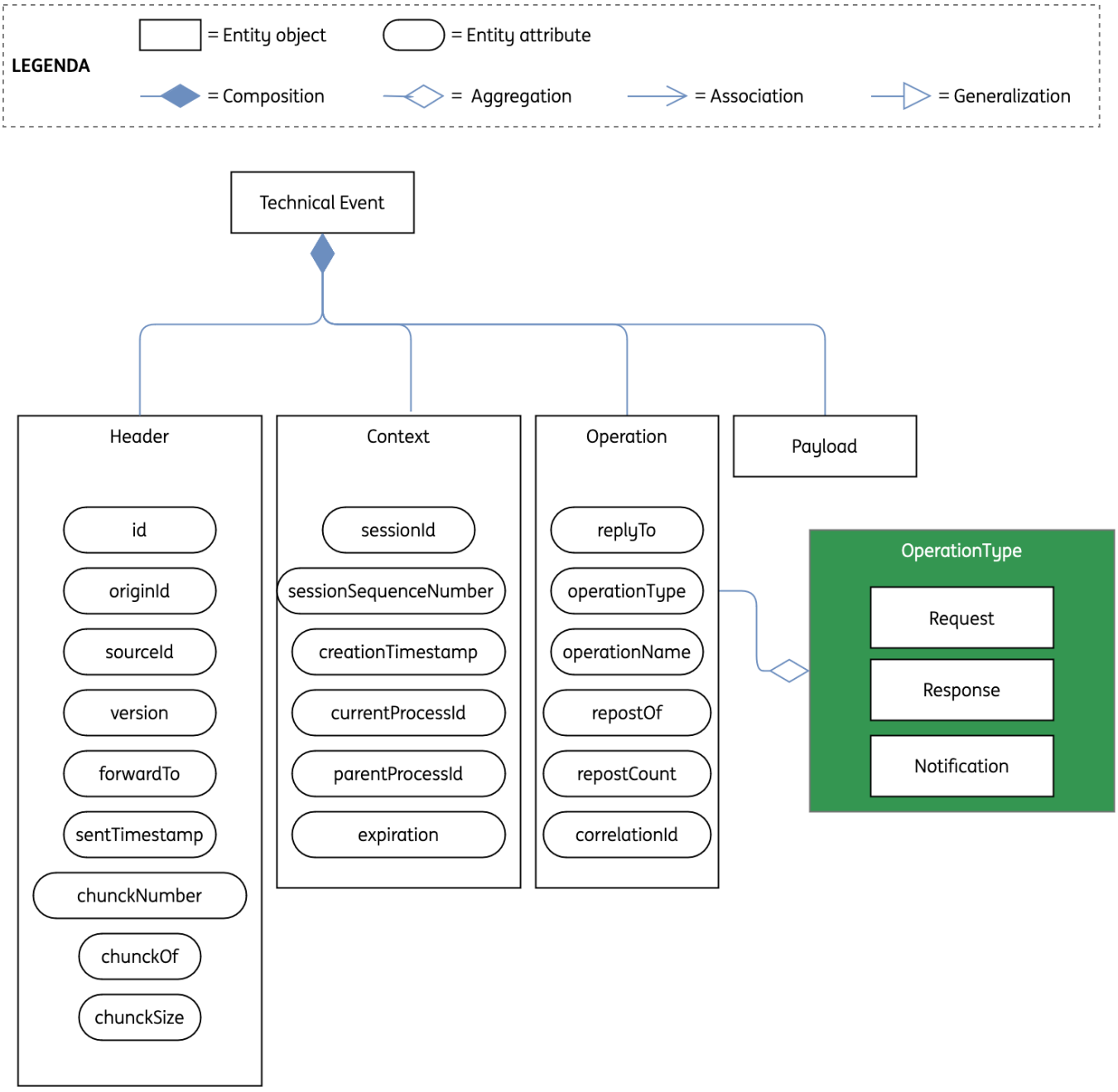
- **Supporting Interaction Mechanisms**
  - In event-driven system messages often need to carry metadata about how they should be processed, routed or responded to; for example a *replyTo* header in a message indicates where responses should be sent, enabling asynchronous request-response patterns
- **Enabling Application Monitoring and Observability**

The System Process Event Data Model

Naming convention (... Proposal)

Term	Description
Request	Any *System Event* that delivers a request execution command; a request is always tied to a Response or a set of possible Responses
Response	Any *System Event* that delivers a reply to an execution command request; a response is always associated to a Request.
Notification	Any *System Event* that delivers a status information from one system component to another, without expecting and response or Reply.

Entity Relation Diagram



Header

This dictionary contains additional information which are not strictly content related. These attributes should be used for troubleshooting and monitoring reason.

Due to the fact we are defining a standard model for all the business event the headers have to be included into Kafka message and Avro schema.

Name	Description	Format	Example	Mandatory	Default Value
id	Message Unique identifier. It is NOT context related	UUID	Random UUID	Y	

originId	It logical initiator of the event conversation. It should trace information related to ▪ BUSINESS DOMAIN ▪ ORIGINATING API ▪ ORIGINATING SERVICE ▪ ORIGINATING_PROCESS_ID It can be useful for tracing event stream which "connects" different domains. It's logically acts as the traceid in synchronous http requests	Domain. OriginatingAPI. OriginatingService.	Party.ITAPega. Offboard. [ProcessID]	Y	
sourceId	Producer application identifier. It should be structured with this info: ▪ ApplicationID (applicationName)	ApplicationID		Y	
version	Event schema version	major.minor.patch	1.0.0	Y	
forwardTo	This field could contains Topics where the message can be forwarded as it is. This can be usefull for monitoring or troubleshooting reasons	TopicName(; TopicName)*		N	
sentTimestamp	Event production timestamp (this is when the event has been sent not generated).In some scenarios this timestamp can be equal to context. Timestamp	date-time standard format	2020-10-24 12.00.00	Y	Current timestamp
chunckPosition	Considering the usecase where is needed to split a big business event into smallest ones, this field declare the position of this specific chunck into the full list	digit		N	
chunckOf	Considering the usecase where is needed to split a big business event into smallest ones, this field declare the expected total number of chuncks.	digit		N	
chunckSize	Considering the usecase where is needed to split a big business event into smallest ones, this field declare the expected chunck size in bytes	digit		N	

Context

This object contains content related information. These attributes should be used for business purposed and application monitoring/troubleshooting

sessionId	This is the overall session identifier. This field must contains the profileSession (see ESA - Auditing Logging TPA in references) or a process session identifier when the process is not "started" by customer/employee	Unique alphanumeric string (UUID or HASH)		Y	
sessionType	This field declare the sessionid origin type (AT = AccessToken, BT= BatchToken)	AT|BT		Y	
sessionSequenceNumber	This is incremental number of messages that have been sent in the same session	Sequence Number		Y	
creationTimestamp	Event generation timestamp (this is when the event has been generated not sent). In some scenarios this timestamp can be equal to sentTimestamp	date-time standard format		Y	
currentProcessId	This is the unique identifier of the current process and it has the same structure of origin • BUSINESS DOMAIN • CURRENT API • CURRENT SERVICE • CURRENT_PROCESS_ID This is equals to the originatingProcessId at the first level	Domain.CurrentAPI. CurrentService. [ProcessID]	Lending. ITAPersonalLoan. CloseLoan. [ProcessID]	N	
parentProcessId	This is the related parent-process identifier and it has the same structure of origin. It could be blank when there is no parent-process • BUSINESS DOMAIN • PARENT API • PARENT SERVICE • PARENT_PROCESS_ID	Domain.ParentAPI. ParentService. [ProcessID]	Party.ITAPega. Offboard.[ProcessID]	N	
expiration	This field sets the required processing timeout. It can be useful in setting up recovery or retry policies	date-time standard format		N	

Operation

This object contains behavioral information

Name	Description	Format	Mandatory	Default Value
replyTo	This field contains the reply topics list	TopicName(;TopicName)*	N (M only for Request-Response)	

operationType	This is the required action type. ▪ **Notification**: event which informs that the current message is a notification concerning the current system state (it includes also ack) ▪ **Request**: event which informs that the current message is an execution request and it is mandatory to receive a response (also with status) ▪ **Response**:event which informs that the current message is an execution response it is always related to a request	NTFCTN|RQST|RSPNS	Y	
OperationName	This is the specific actionid known by producer and consumer. For example "ChangeAccountRequest or CloseAccountRequest or ChangeAccountNotification"	An action known by producer and consumer	N	
correlationID	This is the identifier which binds request and response	UUID	N (M only for Request-Response)	
repostOf	This is a reference to the original id	UUID	N	
repostCount	This a retry-counter	Digit	N	

Payload

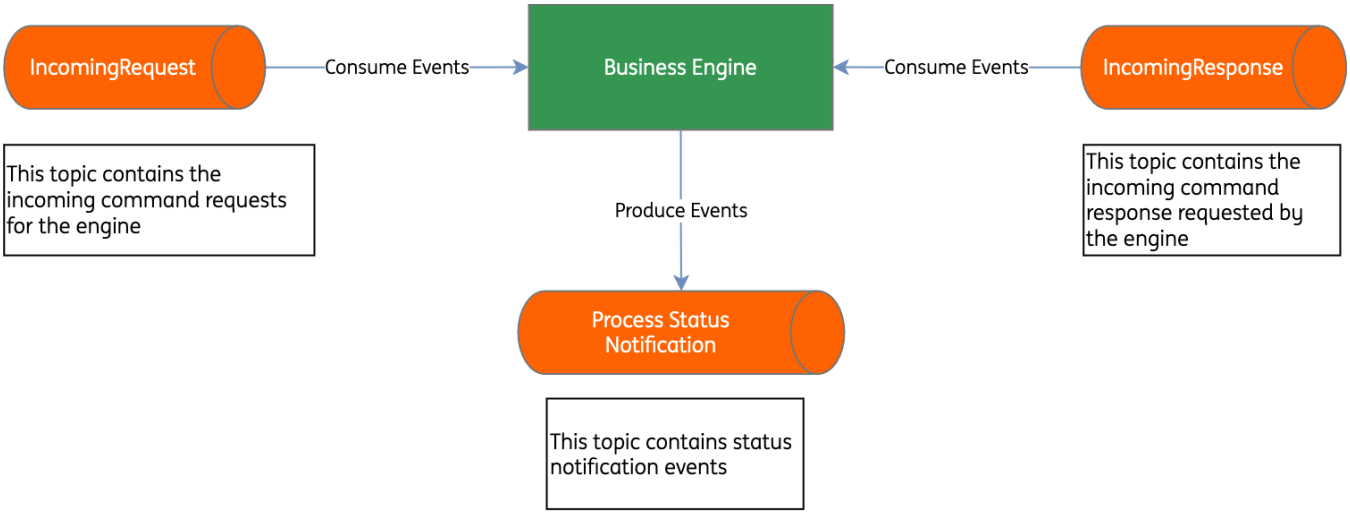
This object is event specific and contains the event data.

Topics Topology Proposal

This section depicts a proposed Topics Topology related to a specific business Engine.

We identified three topics types:

- **IncomingRequest**: this topic contains events which are incoming requests from other peers. These events are consumed by the engine
- **IncomingResponse**: this topic contains events which are incoming responses requested by the business engine. These events are consumed by the engine
- **Notification**: this topic contains events related to business object status notification. These events are produced by the engine



Naming Convention Proposal:

Type	Syntax	Note	Example
Request	"ITA-" + <<OneNameEngineCamelStyle>> + "-INRQST-" + <<version>> + "-TOPIC"		ITA-InsuranceAPI-INRQST-001-TOPIC
Response	"ITA-" + <<OneNameEngineCamelStyle>> + "-INRSPS-" + <<version>> + "-TOPIC"		ITA-InsuranceAPI-INRSPS-001-TOPIC
Notification	"ITA-" + <<OneNameEngineCamelStyle>> + "-PRCSNTFCN-" + <<version>> + "-TOPIC"		ITA-InsuranceAPI-PRCSNTFCN-001-TOPIC

Important Notices:

- This topology implies that Incoming Request and Response events have all the same priority
- The Engine is the owner of the avro schemas of these topics

Activity list

Outcomes

Assumptions

An assumption is something that we need to prove to transform it in a principle.

ID	Name	Explanation	Rationale
A.1			

Constraint

A constraint is something you can't do, so you have to adequate the solution.

ID	Name	Explanation	Rationale
C.1			

Decisions

ID	Name	Explanation	Link to decision registry
D.1			

Open points

ID	Assignee	Status	Explanation	Closing Notes
O.1	Architect Board	OPEN	Headers metadata replication. Some header field can be replicated, both in kafka payload and headers) for technical reason for instance filtering, dispatching by correlationid	

References

ID	Description	Attachment
R.1	ESA - Auditing Logging TPA	ESA - OWA - Audit Logging TPA
R.2	Business Event Categories Endorsed (Draft)	https://orangesharing.com/pages/viewpage.action?pageId=305110214
R.3	Maggie's Event Taxonomy	Event Taxonomy for the Global Data AB - v1.0.pdf