

Event Governance At Scale

A practical guide to effective Event Message Design

Author: Timothy S Hilgenberg <timhilco@gmail.com> Copyright 2021. All rights Reserved Working Draft V0.9.02 February 2022

Preface

NOTE: This book is an early draft of this manuscript

This document defines a standard for asynchronous messaging for any company's Event Architecture. It is a programming language-agnostic interface description for asynchronous message processing. It is also intended to be message platform-agnostic. All messages published by any producer must conform to these standards. The specification also provides consumer of the messages with the detailed information they need to properly understand, consume and process these messages

What's in the Book?

Chapter 1 - Why is Event Driven Architecture critical to tomorrow's application?

Chapter 2 - What are the types of message in Event Driven Architecture?

Chapter 3 - What are the Event Message Specification?

Chapter 4 - Domain Event Example - Consumer Business Process Events

Chapter 5 - System Event Example - Runtime Operations Events

Chapter 1

Why is Event Driven Architecture critical to tomorrow's applications?

Today's web based architectures have been predominantly synchronous. The user presses a button or link and then waits for a response. Since HTTP is the primary interaction protocol used in web/mobile application and it is a Request-Reply paradigm, it's driving this user interaction model. However, there are many application models in practice today, that do not fit this model. People may not think of it in this way, but email and text are really disconnected (asynchronous) interaction models. They make a request or submission and then can perform other action, but they are not truly waiting for a response. So there is precedence for user interface design using an asynchronous model. The key to designing these types of applications is using asynchronous messaging technologies (IBM MQ, Kafka, RabbitMQ) and the design and governance of an organization's domain related events.

Asynchronous technologies have been around for a very long time. The IBM mainframe environment had MQ middleware that allowed inter-machine communication and intra-platform (primarily distributed platform). There are even more modern technologies, like Kafka and RabbitMQ. In prior application designs and even today, these technologies were used for inter-platform communication and more balanced resource allocation and management. They were mostly used for underlying application operations, but not used in an interface design.

So, why are interface designers, using a disconnected paradigm like email or text? These types of interactions are very hard to design for. There are some situations, where the user expects a synchronous interaction. It is hard in an event driven model to

simulate a synchronous model. It requires a very low latency interactions, which was hard to do with older technologies. It was just easier to design a synchronous model when all the interactions were within the company computing ecosystem and there wasn't available technologies other than HTTP to provide the user response, a user expects from the application

What is different in today's environment?

- More and more of the services are being provided by external parties who are outside of the control of the main application. This leads to more interactions and increase wait time as the services are executed in a sequential manner
- More modern messaging technologies are available to application architect to lower any time latency. There is also more practical experience in Event Driven Architecture for architect and designers to leverage
- User are becoming more comfortable with a disconnected interaction model, where they submit a request, work on a different task and then act on the response they get after the fact. (Deep linking into the response)

As with any other scaled integration strategy, the key to success is standards and event design needs to follow this strategy.

Why are event design standards critical for organizational success?

Chapter 2 - Message Types

Overview

This chapter provide the key definitions related to the messages supported within the ecosystem. It will define the types of message and some of the guiding principles about them. It will also describe the message formats available for both internal and secure external usage. The following chapters contains the detail specifications for each of the message types:

Messages

A general-purpose data structure with no special intent. In the integration world, these are typically just streamed between the systems for logging, searching, and for other operational and regulatory reasons.

Message Types

As a component of the event architecture, there are 3 types of asynchronous messages:

Events

An Event is a message which informs various listeners about something which has happened in the past. It is sent by a producer which doesn't know and doesn't care about the consumers of the event. This type of Messages promotes highly decoupled systems using pub/sub architectures.

An Event is an immutable record of a single event at a moment in time.

Commands

Commands trigger something which should happen in the future. It's typically a one-to-one connection between a producer (who sends the command) and a consumer (who takes and executes the command) and, few times, the order of commands is also of utmost importance. Commands are usually performed by actors outside the current system. However, commands can also be rejected, requiring new error handling patterns.

The difference in thinking between an event and a command is an event-X has occurred, rather than command-Y which should be executed.

Audit (Proposed)

An Audit message is an adhoc publishing of a domain business objects state. There is no true triggering action, either from a

business process or straight data change. It would typically be triggered in batch fashion with a query predicate. There will be situations where there are failures in the pipeline, which might lead to data inconsistencies with systems of record. Since some of the business objects are very stable and don't change often, the audit message is used to get to eventual data consistency. A full business object for a bounded context can be published on a periodic basis and then any consumer caches can be updated. Thus, can also be used to seed new consumers with domain data.

Event - Something of interest that has happened in the past

- Represents something that has happened
 - Has a defined timepoint
- Events are both a historical fact and a notification;
 - Notification - a call for action, this is considered a stateless event
 - A state transfer - pushing data wherever it is needed, known as an Event-Carried State Transfer
- Events are immutable
- Name should indicate a past tense action - past-participle verb
- Should be action oriented
- Definition:
 - The result of some outcome
 - Collins: a happening or occurrence, esp. when important
 - Can't be rejected, but can be ignored; no expectation of any future action
- Never includes a response
- Sometimes could be the result of a business process (i.e. completed enrollment) or command (i.e. Change medical plan election)

Command - Represents a request to perform an important action task

- Webster Definition: *to direct authoritatively*
- Other Definitions
 - Represents an intent to perform some sort of action
- The action has not yet happened
 - Example: Change medical plan election
- Named with an imperative Verb
 - The action has not yet happened
 - Example: Change medical plan election
- Has an explicit expectation that something (a state change or side effect) will happen in the future
- Can be rejected
- Typically, a one-way request or fire message with no response
 - However, this can be used in situations where an async request/reply is desired
 - Not to be confused with a synchronous Request/reply like REST API Message provides a call back
- Can generate one or more events as part of its processing

Audit - Represents the current state of a business object - Published on specific schedule
- Proposed

- To be defined
-

Chapter 3 Event Message Specifications

Overview

Key features of Event Message Specification are:

- Provides Unique Message Identifier
- Provides auditing data, date, time, as to when the message was created
- Is Independent and self-contained
- Provides simple headers and metadata to facilitate routing and filtering within the event processing network
- Provides provenance for the message
 - *When did it happen*
 - *Who the source is, both system of record and publishing platform*
 - *What was the cause of the event*
- Supports schema version control and message validation
- Provide the ability to find a subject in a system of record
 - Relates business object can be attached to the message
- Provides correlation for cross event processing.
- Provides the ability to submit test or synthetic event for testing

Event Types

The follow sections provide the specification for the types of event support by the architecture. (Note: Some event types are in the prototype stage) The event types are:

Types

Consumer

Business Process State Change Event

Business Object Data Change Event

User Experience Action Event

Generic Goal Event

Runtime

Platform Processing Event

Event Message Overview

The event message is a JSON document containing one JSON object named **message**. The message object contains 3 JSON objects: a header, a common header for specific event types and a free form body for each event.

Message Header

Notes:

undefined

Header provides the following key features:

- Message Metadata
 - Id
 - Message typing
- Event MetaData
 - Name
 - Schema Definition
- Event Context
 - Event Context Metadata
 - Subject Type and Id
 - Related Resources
- Audit History/Chain of Custody
 - Publishers
 - System of Record/Source Publisher
- Every message type, events or commands, will have a common standard header
- There will only be one format or schema for the header and the object is required
- The name of the JSON object is **header**
- It contains fields that describe the message at the highest levels and it identifies the source and type of the message
 - These fields determine the format and names of the fields that follow in the message object.
- Since this is JSON, routing or filtering consumers can use only the header to determine routing of message or if the consumer is interested in processing the message

Event Type Header

- It is a secondary header that contains the common elements for all messages of a type
- Each event type will have its own header name and structure. Examples:
 - uxEventHeader - for Ux events
 - bpEventHeader - for business process events
- The messageNamespace field in the header will indicate which event type header is in the message
- There will be a format/schema for each event type
- There is a small bounded list of event types

Message Body

- Contains the fields that are specific to a given instance of an event type
 - The system will have a large unbounded set of events. The body represents the specific fields for a given event
- The eventBodyNamespace in the Event Type Header will describe the schema for the fields in the body
- The name of the JSON object is **body**

Internal Event JSON Structure

To keep it simple and easy to produce and consume, the event message has a very flexible structure and is basically an unstructured document. The goal is to have a schema for the header, each event type header and every event data (i.e body) itself. We would like to have a schema dictionary which has a JSON or AVRO schema as it values and it’s keyed by some name. The hierarchy is as follows:

- There is only one header schema (key name: header)
- To determine the <eventTypeheader> name, the header.messageNamespace field contains the name of the event type
- To determine the body schema name, the header.eventBodyNamespace field determine the name for the body schema

The internal event structure looks as follows:

```
{"message" : "header" : { ... }, "<eventTypeHeader>" : { ... }, "body" : { ... } }
```

Samples

```
{"message" : "header" : { "messageNamespace": "com.hilco.messages/uxEvent", "eventName" : "PageABC:clicked" ... },  
"uxEventHeader" : { ... }, "body" : { ... } }
```

```
{"message" : "header" : { "messageNamespace": "com.hilco.messages/bpEvent", "eventName" :  
"ContributionRateChange:Completed" ... }, "bpEventHeader" : { ... }, "body" : { ... } }
```

Table 1. Event Header Schema Fields Table (Choose between this and the next section)

Field Name	Type
messageId	String
messageType	String
messageNamespace	String
messageVersion	String
messageTopic	String
eventName	String
eventBodyNamespace	String
contextTag	String
action	String

Event Header Schema Fields

- messageId
- messageType
- messageNamespace
- messageVersion
- messageTopic
- eventName
- eventBodyNamespace

- contextTag
- action
- messageTimestamp
- businessDomain
- correlationId
- correlationIdType
- normalizedClientId
- globalBusinessObjectIdentifier
- publisherId
- publisherApplicationName
- publisherApplicationInstanceId
- publishingPlatformsHistory
 - publisherId
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationName
 - publisherApplicationInstanceId
 - messageId
 - messageTopic
 - eventName
 - messageTimestamp
 - sequenceNumber
- personIdentificationSystemOfRecord
 - systemOfRecordSystemId
 - systemOfRecordApplicationName
 - systemOfRecordApplicationInstanceId
 - systemOfRecordDatabaseSchema
 - platformInternalId
 - platformExternalId
 - platformClientId
- correlatedResources
 - correlatedResourceType

- correlatedResourceIdentifier
- correlatedResourceState
- correlatedResourceDescription
- isSyntheticEvent

Event Header Field Specification

Note: Mandatory fields are denoted with an asterick.

messageId	Global and Unique (UUID) Identifier of message.
messageType	Describes the type of message. Valid Values: <ul style="list-style-type: none"> • Event • Command • Audit
messageNamespace	Namespace is used to distinguish between different types of messages (events vs commands), source (internal vs external), and schema versions to avoid collision and help in processing the messages. The namespace can be used as an external endpoint to provide the schema and other machine-readable information for the event type and the latest major version. Used to provide message definition and validation. Valid Values: <ul style="list-style-type: none"> • com.hilco.messages/events/person/uxEvent • com.hilco.messages/events/person/businessProcessEvent • com.hilco.messages/events/person/dataChangeEvent • com.hilco.messages/events/person/goalEvent • com.hilco.messages/events/operations/platformProcessingEvent
messageVersion	Conveys the version number (major.minor) of the message, and describes the structure of the overall message at hand. Valid values managed by governance Example: 1.1
messageTopic	Logical name to describe the type of event. Note: this is not the physical topic name (i.e kafka topic) of the messaging system. Valid Values: BusinessProcess DomainDataChange UserExperience Goal PlatformProcess
eventName	Provides a standard name of the actual event that happened based on a user's behavior action. It will be treated as a label/code and used for filtering, routing, general analytics and simple processing of events in the ecosystem. It should be a combination of the business process name and action taken on that process. There are specific naming conventions used to determine the value of the field. It is a field that will require governance approval.

eventBodyNamespace	Describes the specific schema and version of the body field structure of the event. The body structure and metadata details are understood based on this combination. This field is optional and only be set if there is a structure or schema for the body. If there is not body, then this field should not be sent.
contextTag	Machine readable generic label for the event type. Its purpose is to provide a label that encoded some additional context for the event. It is highly structured, follows a specific format and provides valid values to allow program and applications, like analytics, to easily consume the values. See event type for more details on the values. To reduce the complexity in trying to capture all the level and types of components, we are going to encode all contextual or hierarchical information into a single label or tag. This tag along with the user action on this tag should reduce the complexity of the event structure and make it easier for the consuming tools to do their work without having to get into the details of the body structure To make it more human readable, there will be an encoding standard to make it more human readable and make it easier to parse the tag if necessary.
action	Represents the action or happening based on the event type. See event type for more details on the valid values. For events, the action should be described in the past tense and the name should be initial caps.
messageTimestamp	Describes the date and time at which the actual event was generated by publishing systems. To be provided by producer component and should not be derived by message publishing framework(s) or component(s) The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.
businessDomain	Describes the business domain under which the event/command was generated. Valid Values: <ul style="list-style-type: none"> • Person • Worker • PersonWorker • Health • DefinedContribution • DefinedBenefit • Operations • N/A (for domains that do not match up to our organization service domains, This would include areas like Productivity, CustomerService/ServiceNow and Advocacy)
correlationId	Describes the globally unique identifier (UUID) typically generated within the publishing application. This is used to correlate multiple messages across a logical process. The messageId is unique for the individual message, but the correleationId can be repeated across multiple messages

correlationIdType	Describes the type of correlation identifier. Valid Values: SessionId - for participant actions and sessions BatchId - for batch processing jobs. This is the actual instance id of a job type. PublisherCorrelationId - for publisher specific correction type (Typically used if the above two does not apply)
normalizedClientId	Identifies the Normalized Client Identifier across a Companies Platforms
globalBusinessObjectIdentifier	Describes the global identity of the participant within hilco, in particular the CustomerMaster platform. Required if source platform of record Ids are not present and the event is related to a participant: Note: sometimes this is referred to as the universalId.
publisherId	Identifies the publishing company entity of the message.
publisherApplicationName	Describes the name of the publisher application platform or service.
publisherApplicationInstanceId	Describes the specific instance of the publisher application or service.

publishingPlatformsHistory

Publishing Applications history and details. This is the history and providence of the message, It is the array, describing the platforms that have been processing a given message from the edge platforms to any internal consumer applications.

publisherId

Identifies the publishing company entity of the message.

publisherApplicationName

Describes the name of the publisher application platform or service. See Appendix for list of publishing applications.

publisherApplicationInstanceId

Describes the specific instance of the publisher application or service.

messageId

Describes the messageId for the given prior message instance. See above for field details

messageTopic

Describes the messageTopic for the given prior message instance. See above for field details

eventName

Describes the eventName for the given prior message instance. See above for field details

messageTimestamp

Describes the messageTimestamp for the given prior message instance. See above for field details

sequenceNumber

The sequence should be from earliest to latest in chronological order. The publisher should only append to the array If the array is provided as input from a message, then the new publisher should increase the sequence number and append the consumed/input header data to the array. If this is the originating or edge processor, then the sequence number should be set to one (1), not zero

personIdentificationSystemOfRecord System of Record containing details related to finding a person. Required if globalPersonIdentifier is not present and the event is participant related.
systemOfRecordSystemId:::Identifies the system of record company entity of the message. Sometimes referred to as the partner ID.

systemOfRecordApplicationName

Describes the name of the publisher application platform or service. This section should contain the best system for person related data. (For hilco, this is CustomerMaster). If that system is not available, then the publishing application should provide the best platform available.

systemOfRecordApplicationInstanceId

Describes the specific instance of the system of record containing the person

systemOfRecordDatabaseSchema

Describes the database schema instance of the system of record containing the person

platformInternalId

Describes the internal identity of the participant within the platform. Only provided if the publishing platform is a source system of record and not a pure publisher application

platformExternalId

Describes the external identity of the participant within the platform. Only provided if the publishing platform is a source system of record and not a pure publisher application

platformClientId

Describes the client Id in the publishing platform. This is a platform specific ClientID. The normalized ClientId is above

correlatedResources

Describes a list of the related resources. These are key *bounded contexts* associated with the primary business entity. This can be 'campaign' or 'business process' or some other resource related to the action performed by the end user.

correlatedResourceType

Describes the type of the related resource. Valid Values:

- PersonActivity
- Document
- Plan
- Transaction
- Fund
- Account
- Address
- PersonDefinedBenefitCalculation
- Campaign & PersonCampaign

correlatedResourceIdentifier

Identifies the primary key of related resource. This can be the external or internal unique identifier of the resource.

correlatedResourceState

Identifies the state or status of related resource at the time the event occurred.

correlatedResourceDescription

Description of related resource at the time the event occurred.

isSyntheticEvent

Is this a synthetic or fake event? If true, assumes this is an event that should be processed under special circumstance, meaning don't change state or issue commands. Used for testing/monitoring in production by sending in fake events

Potential Extensions: *dataContentType* - This will be helpful if the body is not JSON. Our current best practice is that all body payloads, should be JSON. The values would follow HTTP mime types

Chapter 4 - Domain Event Example - Consumer Business Process Events

This chapter discusses the following consumer related events:

- Business Process State Change events
- Business Domains Object data changes
- Consumer UX interactions events

- Consumer Activities event

All of these event would use the Event Header described in the prior chapter.

Business Process State Change Event

The purpose of this event type is capture events related to the processing of business events. These events are sometimes known as transaction and are used to manage the changes in business domain objects. Upon the completion of a business process, it will typically also create a Business Object Data Change event.

It supports two types of business processes. The first is a long running transaction that has many states and has time gaps in between the actions of the business process. The second is a straight through or single task, where the action is completed in a single unit of work

Business Process examples:

- Contribution Rate Change
- Annual Enrollment
- Defined Benefit Retirement
- State changes includes:
 - Start
 - Complete
 - Valid
 - In Error

Business Process State Change Header Specification

JSON Name: bpEventHeader

businessProcessReferenceId	Describes the primary key or Business Process Reference Identifier of the business process person instance as described in platform(s). businessProcessId::Describes the internal identifier of the business process. It is the template used to create the person specific instances of the business process.
rawBusinessProcessName	Describes the pre-normalized Business Process Name as described in platform(s). This could be the client specific business process name or a normalized name across clients. It would be used primarily for reporting, filtering and aggregating businessProcessDescription::Describes the long more formal description of the business process. businessProcessStatus::Business Process status.Valid Values: <ul style="list-style-type: none"> • Created • Valid • Invalid • Completed • Canceled
businessProcessEffectiveDate	Effective date of the business process

businessProcessChangeTimestamp Timestamp of when the business process was changed. The timestamp must be in the ISO 8601 date format standard.

EventName Standards

For the eventName, the standard will be the following fields separated by a colon (":") in camel case.

- *tag* which represents the business process name, hopefully normalized and
- *action* which represents the business process action

Tag Definition

The tag represents the business process name. It should be the normalized version of the business process name.

- Format
 - Free format single alpha numeric value
 - No formal specification is defined

Action Definition

The action represents the types of actions that result from the change in status of the business process during the processing of the consumer's business process.

- Action Component Valid Values
 - Started
 - Updated
 - Completed
 - Canceled

Body Definition Considerations

The body section is named **body**. The **body** can be any valid JSON schema

Business Object Data Change Event

The purpose of this event type is to capture the changes to key domain business objects.

- Business Objects include:
 - Person
 - Person Defined Contribution
 - Person Health Management
 - Person Defined Benefit
 - Person Document[TH1]
- Data actions Include:
 - Creation
 - Updates
 - Deletion
 - Master Data Management Document Merge/Split

Business Objects Data Change Event Specifications

Business Objects Data Change Header .JSON Name: boEventHeader

businessObjectResourceType


Describes the primary domain data object type that was changed. Valid Values:

- person
- personDefinedContribution
- personHealthManagement
- personDefinedBenefit
- personDefinedBenefitCalculation
- personDocument Editor: Think about moving this to 'tag'. Need to determine in the Identifier is included in the tag

businessObjectIdentifier

Provides the primary domain data object key of the business object that was changed

additionalBusinessObjectResource

Provides any additional resource type and key to help further identify the component that changed. This is similar to the pathing ( /resource/{id}) in a REST URL

additionalBusinessObjectResourceType

Additional resource type

additionalBusinessObjectResourceIdentifier

Additional resource identifier or primary key

dataChangeTimestamp

Timestamp of the data change in the source platform. The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.

EventName Standards

For the eventName, the standard will be the following fields separated by a colon (":") in camel case.

- *tag* which represents the business object name and
- *action* which represents the CRUD operation taken against the business object

Tag Definition

The tag represents the business object name. Editor Note: Should tag replace 'businessObjectResourceType' .

- Format
 - Free format single alpha numeric value
 - No formal specification is defined

Action Definition

The action defines the type of data maintenance (CRUD) action taken on the business object. Editor Note: action is replacing the dataAction field in prior versions.

- Action Component Valid Values

dataAction

Describes the data change or CRUD action performed on business object.- Create, Update, Delete. Also includes an primary key changes and MDM document merging.

- Create
- Update
- Delete
- MdmDocumentMerge
- MdmDocumentSplit

Body Definition Considerations

- The body section is named 'body'
 - body can be any valid JSON schema
 - Contains one predefined element 'extension'
 - Extension is a private area that can contain its own schema
 - The field is an map/array with:
 - Namespace as a key and,
 - Any valid JSON schema as its value

Data Fields Best Practices by Data Action

- Update

The recommendation for data fields to report is to provide only the fields that changed providing both old and new Best practice recommendations:

- PII
 - Fields: Bank/Credit Account Numbers,
 - Provide old/new unchanged from CustomerMaster; no masking required
- Arrays
 - Provides Lowest Level Detail field, include all cascading keys
 - Example: Contact → streetAddress → { AddrID → OldZipcode, newZipcode }
 - Include all the fields at the same level as the changed field in entire array data object
 - For fields in a high level/hierarchy, include all keys and simple primitive types (strings, numbers,etc) at the same hierarchy
- Do not include objects or arrays in the higher levels Do not include non-changing arrays at the same level
- Create
 - Provide the entire New document
 - Alternate: Keys Only
- Delete

- Only provide a delete event if the entire document is being deleted, not if one of the source systems deleted a person
- In the body, provide the primary document key (UniversalId or Mongo _id) and any IdMapping table
- If the object/person is being delete in a given platform, but the person still exists in another platform, treat as an Update
- Only delete when no more IdMappings exist in the document

Master Data Management Platforms/CustomerMaster

- Merge
- Treat as an MDM Merge Update event with two sections of data, one for survivor and one for deleted
- Both sections
- Survivor _id & Deleted _id
- Id Mapping for both survivor and deleted
- Survivor document section contains the update record for the survivor document (see Update section)
- Deleted document section
- Reason for merge
- The Platform that caused the change to occur
 - System Instance
 - Merge Field Change (old, new)
- Split - No new events, just two new event being generated
 - Web service call to deletePersonId service, which cleans up IdMapping and domain sections
 - Generates a Normal Update event
 - Web Service call refreshPersonForInternalId service, which causes a refresh through Ingest
 - Generates a Normal Update event

User Experience Action Event

Events related to the behavioral actions taken by the participant in our user experience channels. Channel include web/Upoint, mobile, IVA/chat and other future user devices like Voice Assistants.

The purpose of this event type is to capture the pure behavioral events related to the interactions of the users on our various channels - displaying pages, clicking button or links. These events are not the result of any business process or data change events. They are used for:

- Behavior actions for data reporting and analytics
- Provide notifications to non-domain processes (document management, campaigns) to drive their underlying processes
- Actions may include, but not limited to:
 - Button clicks
 - Link or action selections
 - Page or screen displays
 - Hover
 - IVA or chat intents

User Experience Action Event Specifications

JSON Name: uxEventHeader

channel

Describes the channel (or UI application) where the event generated.

userDevice

Identifies the device used by end-user.

deviceTimestamp

Represents the timestamp on the device (May be different from the publisher timestamp). The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.

sessionId

Represents the unique session of end user on our channels.

sessionCreateTimestamp

Session created time. The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.

applicationName

User Experience application name

applicationVersion

Version of the application

EventName Standards

For the eventName, the standard will be the following fields separated by a colon (":") in camel case.

- UxControlName
- UserAction

Tag Definition

In the Ux channels, there are an unbounded set of device actions a user can take: pressing buttons, displaying pages, starting process flows. In addition, they are an unbounded set of specific controls (buttons, etc) throughout the interface. For reporting and other activities, there is a need to capture that a specific control has been acted upon: pressing a specific button within a specific group of controls within a page within a business process flow.

To reduce the complexity in trying to capture all the level and types of components, we are going to encode all hierarchical information into a single label or tag. This tag along with the user action on this tag should reduce the complexity of the event structure and make it easier for the consuming tools to do their work.

To make it more human readable, there will be an encoding standard to make it more human readable and make it easier to parse the tag if necessary. The tag values need to take into account all types of user interfaces and devices. We need to support new and emerging interfaces beyond web and mobile channels. The following sections discuss the naming approach.

Tag Component Valid Values

***Channel**

- Web
 - Flow - A user's perceived outcome process or unit of work

undefined

- Denotes flow of interaction (pages) or conversation between user and system
- Page
- Widget or Multiple Control Component
- Elemental Ux Control
 - Button, includes clickable icons - Clickable
 - Link - Clickable
 - CheckBox - Selectable
 - Text - Display, Hover, Table Element
 - TextBox - Keyboard Actions → Tabbing ,Enter pressed
 - Bounded Lists → Radio Buttons or checkboxes or DropDown Lists or Dials - Selectable
- Mobile - TBD
- Smart Assistant/Alexa
- IVA/Chat
- Other on Non-Channel - Treatment or Theme Example xxxA/xxxB

Format

- Ordered sets of tuples separated by underscore '_'
- The tuple is the following fields separated by dash '-'
 - LogicalName determined by Ux Designer and Data Analyst
 - UxControl Valid Value in all caps
- The order is from highest level (aFlow) to specific UX Control, (Button)

Example: <Flow_Name>-FLOW_<Page_Name>-PAGE or Retirement-FLOW_HubPage-PAGE

Action Definition

The action defines the type of user actions taken by the user when interacting with the channel/device. Valid Values for userAction:

- Displayed
- Clicked
- Entered

Body Definition Considerations

- The body section is named **body**
 - **body** can be any valid JSON schema
 - Contains one predefined element **extension**
 - Extension is a private area that can contain its own schema
 - The field is an map/array with:
 - Namespace as a key and,
 - Any valid JSON schema as its value

- This can be any significant data or data of interest for reporting at the time of the UX Event

Consumer Goal Event

Events related to the action taken by the consumer in the context of reaching a personal goal.

A goal is non-transactional outcome the consumer is trying to attain. For example, the person wants to lose 20lbs as a health goal

- Actions may include:
 - Started
 - Completed

Personal Goal Action Header

Note: The Personal goal only requires the main header

JSON Name: pgEventHeader

Tag Definition

The tag represents the name of the personal goal in a machine readable format.

- Format
 - Free format single alpha numeric value
 - No formal specification is defined

Action Definition

The action defines the type of task actions taken against a personal goal.

- Action Component Valid Values
 - Started
 - Completed

Body Definition Considerations

- The body section is named **body**
- body can be any valid JSON schema
 - Contains one predefined element **extension**
 - Extension is a private area that can contain its own schema
 - The field is an map/array with:
 - Namespace as a key and,
 - Any valid JSON schema as its value
 - This can be any significant data or data of interest for reporting at the time of the UX Event

Chapter 5 - System Event Example - Runtime Operations Events

Events occurring during the running of the application on a specific platform or system resource (server, data base, etc).

Focused mostly on system resource issues. Debug events would also fall into this category

undefined

Platform Processing Event - Events related to the action of completing a discreet process or unit of work and providing the resource computation of that unit of work.

These events reflect the fact that (1) the process occurred, which is used for counting instances of the process (2) the time stamps of then it occurred, which is used for elapse time and windows of time, (3) additional resource usage, which is used for more detailed resource consumption analytics and (4) any additional status and metadata related to the process of the unit of work. They event can be used for both operational and application event where counting, and resource utilization reporting is of interest to the business

- Platform processing units of work being metered include:
 - Docker Containers → Service API calls
 - Enterprise CI/CD Build pipeline → API Service builds

System Resource Manager Event (Under consideration) - Events related to the actions of managing a system resource manager. This includes the overall operations of the resource manager and the system administrator actions to administer the resource manager.

A resource manager has the following characteristics. Example of a resource manager is a server, JVM, Web Server, Docker container, data base manager and queue manager. * The platform contains important application data * The resource is shared by multiple applications * The resource is managed centrally by an system operations staff, who is responsible for the installation, upgrade and overall operations of the resource * Privileged access authority is required to perform system administration functions

Resource manager operational event actions include: * Start * Stop * Abort * Restart

System Administrator event actions include: * Logon * Password change * Commands (?) * Group Functions (?)

Potential Future Operational Events * Runtime System Error Events - Runtime Events because of hardware or software issues * Code Deployment Events - DevOps Events because of program/business logic development. Include both code and configuration * Client Deployment Events - Events related to the deployment of client assets, in particular provision migrations. ** Might also include client level processing runtime errors

Operations Platform Process Event

Event Header See the above header section for definition of the event message header

Platform Process Event Header

JSON Name: oppEventHeader

Note: Mandatory fields are highlighted in bold.

platformProcessStartTimestamp

Timestamp of when the resource consumption started. The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.

platformProcessEndTimestamp

Timestamp of when the resource consumption ended. The timestamp must be in the RFC 3339/ISO 8601 date format standard. See Appendix for details.

platformProcessElapsedTime

Elapse time in milliseconds

platformProcessEffectiveDate

Effective Date of process

platformProcessStatus

Process status. Valid Values:

- Aborted
- Completed
- Canceled

Potential Platform Process Header Fields

- API/Program call Request and Response size
- Memory size at event publication
 - Heap sizes , etc
- CPU utilization for process
- Thread count at event publication

EventName Standards

For the eventName, the standard will be the following fields separated by a colon (':') in camel case.

- PlatformProcess
- Action

Tag Definition

- Format
 - Ordered sets of tuples separated by underscore '_'

Action Definition

The action defines the type of action or state changes of the process.

Action Component Valid Values

Action Valid Values processAction * Started * Completed * StateChanged

Body Definition Considerations

- The body section is named 'body'
 - 'body' can be any valid JSON schema
 - Contains one predefined element 'extension'
 - Extension is a private area that can contain its own schema
 - The field is an map/array with:
 - Namespace as a key and,
 - Any valid JSON schema as its value
- This can be any significant data or data of interest for reporting at the time of the process state change.

Appendix

CloudEvents Comparision

id → messageId source → eventName and Namespace specVersion → messageVersion dataContentType → JSON only
dataSchema → messageNamespace, eventNamespace (body) subject → Subject/Business Object time → messageTimestamp

Last updated 2022-02-23 14:21:26 -0500