



Quote to Order Event

Salesforce Industries & Revenue Cloud Expert Services

Lindemberg Venancio

- SF Industries and Revenue Cloud Senior Developer
- 25+ years of experience in the Telecom industry
- 5+ years in Salesforce Industries
- 4 years creating Salesforce Industries content
- Certified OmniStudio Developer and Consultant Revenue Cloud Consultant
Certified
- Rio de Janeiro, Brazil



Agenda

- Business context & real-world use case
- QuoteToOrderCompletedEvent: official documentation
- QuoteToOrderErrDtlEvent: the nested error event
- Identifying success vs. error from the schema
- How to consume the event
- Our testing findings and example implementation



Business Context & Real-World Use Case

Splitting a Quote into Multiple Orders

`createOrdersFromQuote` (API v65.0+) replaces the deprecated single-order action, and exists specifically to split one Quote into multiple Orders.

- **CreateSingleOrder**: one Order from the Quote (legacy behavior, synchronous)
- **CreateOrderByGroup**: multiple Orders, split by quote line group
- **CreateOrderByField**: multiple Orders, split by a quote line item field (e.g. `Product2Id`)

Order headers are created synchronously; order line items are created asynchronously. This is exactly why one Quote can produce several `QuoteToOrderCompletedEvent` messages.

Example Walkthrough: Quote Split by Address

A Quote with 3 Addresses with Create Orders by Recipients into 3 Orders.

Arrival order	Order	HasErrors	Detail
1st event	Address 1	false	OrderId / OrderNumber populated Success
2nd event	Address 2	false	
3rd event	Address 3	false	

In our use case, the quote will be split based on Quote Recipients (Address).

Upon completion of each conversion, a custom record will be created to track the status of the Quote to Order conversion so we can visualize how this process works.

When all conversions are finished, the quote will be locked and the user will receive a notification.

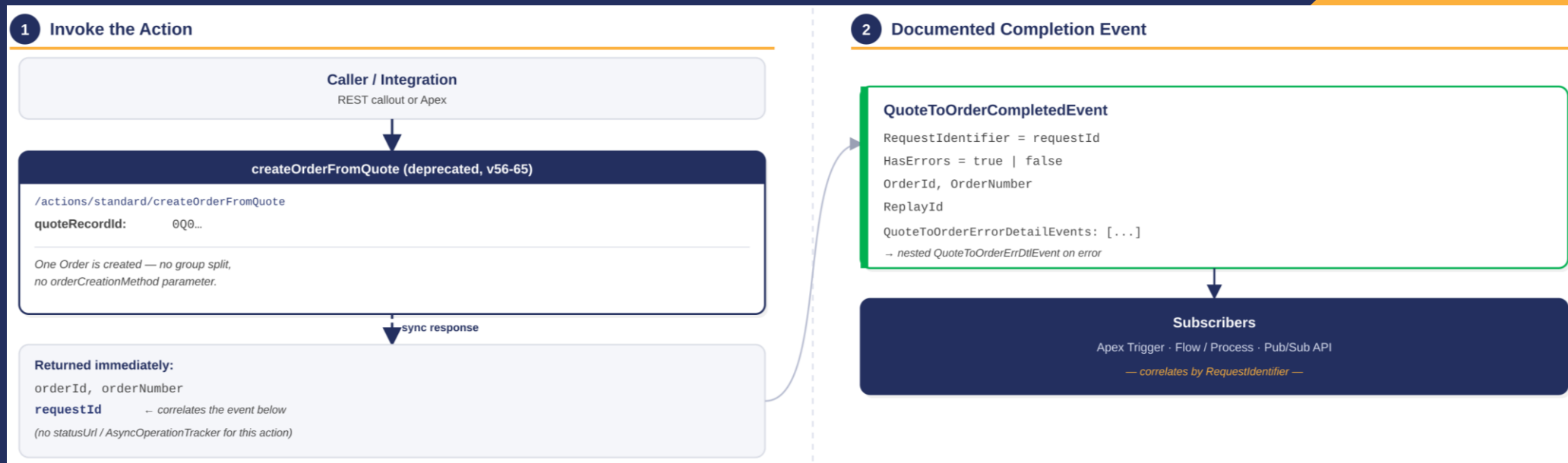


QuoteToOrderCompletedEvent: Official Documentation

Definition & Licensing Requirements

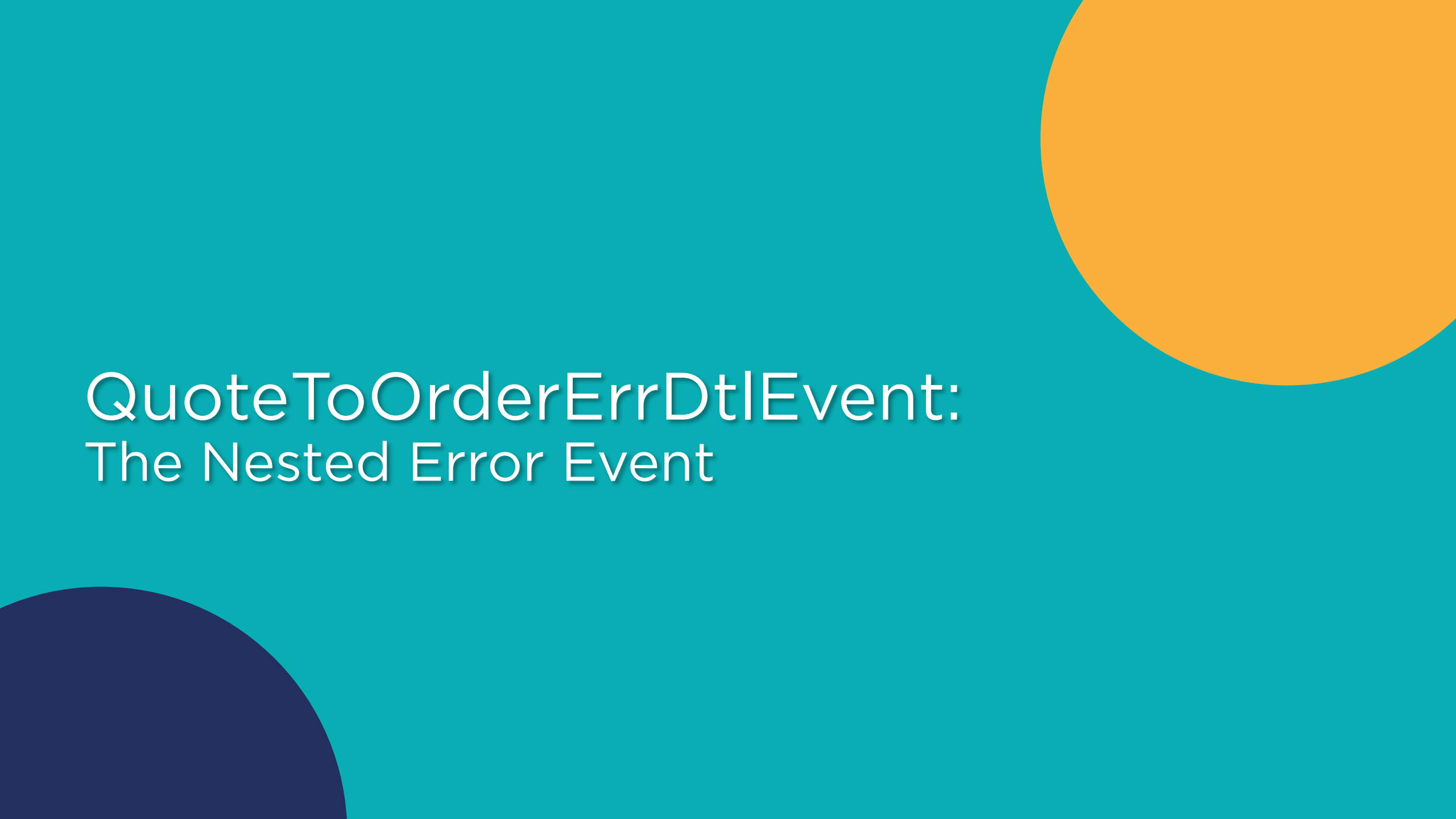
Notifies subscribers when the `/actions/standard/createOrderFromQuote` REST request completes — with success or error. Available from API v56.0.

- **When it fires:** at the end of the Quote → Order conversion, automatically by the platform
- **Licensing requirement:** available only with Revenue Cloud (Special Access Rules)
- **Subscription channel:** `/event/QuoteToOrderCompletedEvent`



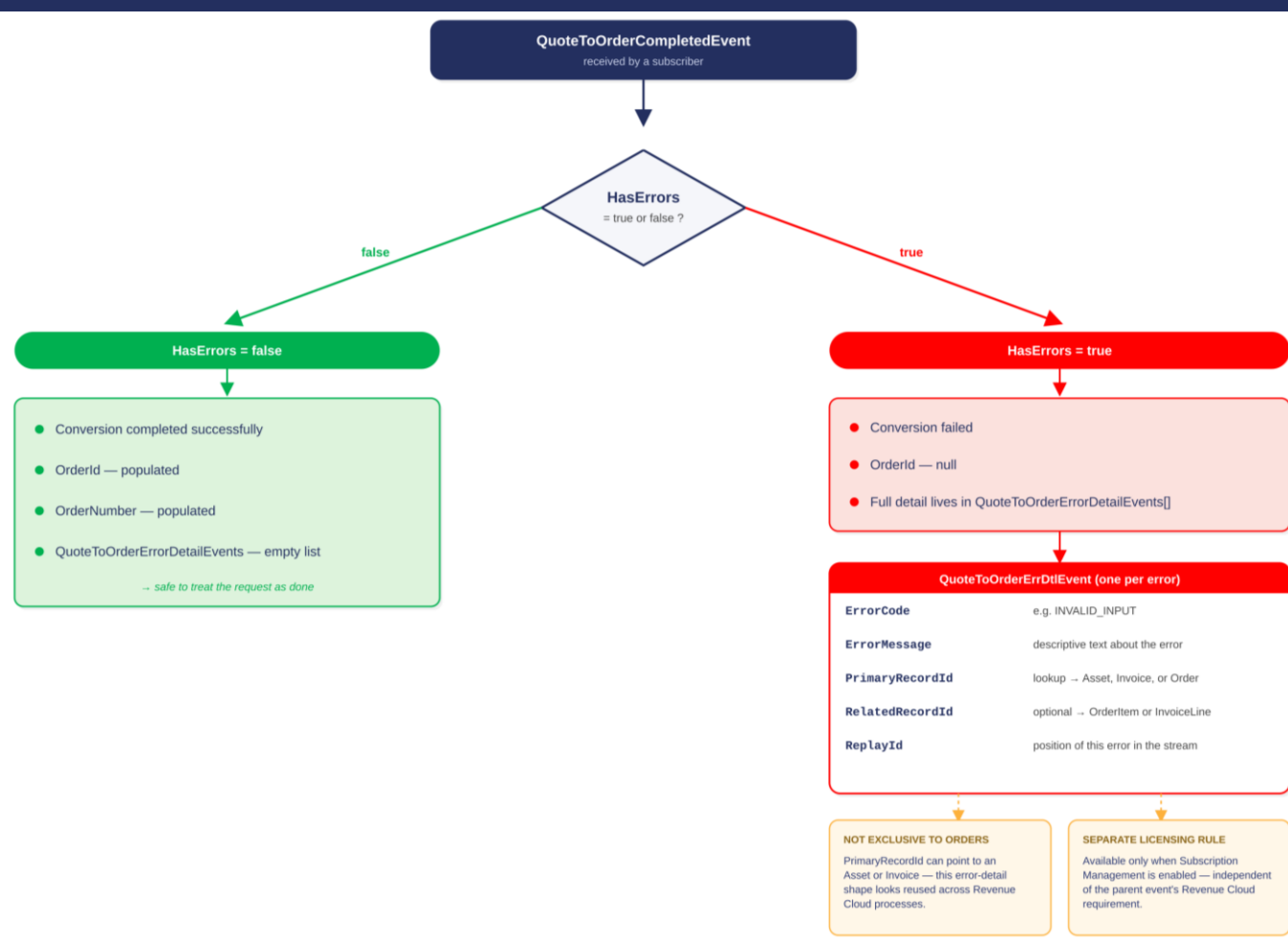
Field Reference : QuoteToOrderCompletedEvent

<i>Field</i>	<i>Type</i>	<i>Role in the use case</i>
RequestIdentifier	string	Correlates the event to the specific createOrderFromQuote call
HasErrors	boolean	Directly indicates whether the process encountered an error
OrderId / OrderNumber	string	Populated only on success; null when the conversion failed
QuoteToOrderErrorDetailEvents	list	One QuoteToOrderErrDtlEvent per error that occurred
ReplayId	string	Position in the stream: allows replaying missed events
EventUuid	string	Unique identifier for the event message
CorrelationIdentifier	string	Reserved for future use



QuoteToOrderErrDtlEvent: The Nested Error Event

Identifying Success vs Error from the Schema



HasErrors = false

- Conversion completed successfully
- OrderId populated
- OrderNumber populated
- QuoteToOrderErrorDetailEvents empty

HasErrors = true

- Conversion failed
- OrderId is null
- Details in QuoteToOrderErrorDetailEvents
- Each error carries an ErrorCode and ErrorMessage

Error Fields & What the Lookups Reveal

Field	Detail
ErrorCode	The error code (e.g. INVALID_INPUT)
ErrorMessage	Descriptive text about the error
PrimaryRecordId	Lookup to Asset, Invoice, or Order
RelatedRecordId	Optional lookup to InvoiceLine or OrderItem
ReplayId	Position of the error event in the stream

NOT EXCLUSIVE TO ORDERS

- PrimaryRecordId can point to Asset or Invoice, not just Order.
- This error-detail structure looks shared across multiple Revenue Cloud processes.
- Available only when Subscription Management is enabled: a separate rule from the parent event.

How to Consume the Event

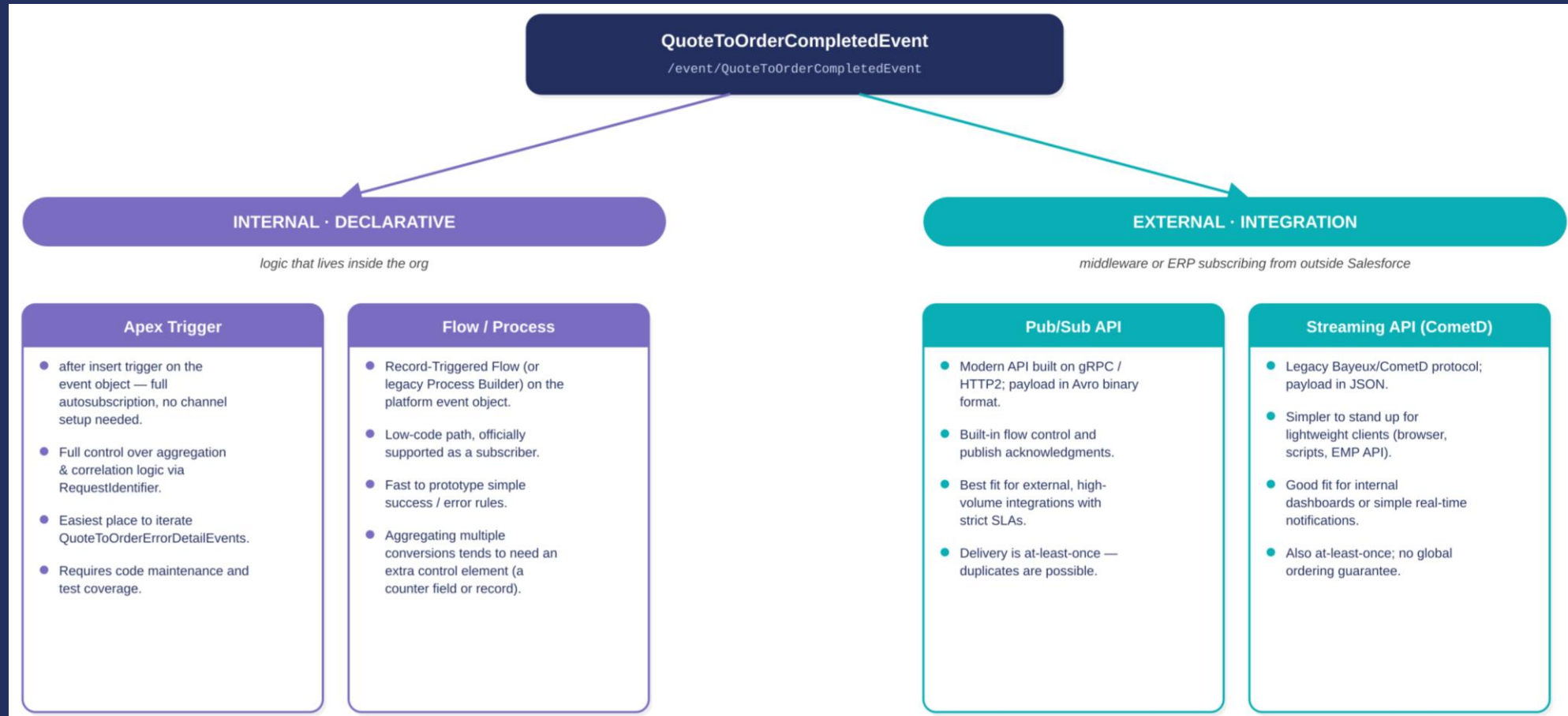
Ways to Consume the Event

- Apex Triggers
- Flows
- Processes
- Pub/Sub API
- Streaming API (CometD)

DESIGN IMPLICATION

- Apex and Flow/Process are internal, declarative paths. Best for logic that lives inside the org.
- Pub/Sub API and CometD are for external integrations with a middleware or ERP subscribing from outside Salesforce.
- None of the five is “the” correct choice by default; the right pick depends on where the aggregation logic should live based on your use case and systems.

Ways to Consume the Event



Apex Triggers & Flows / Processes

Apex Trigger

- after insert trigger on the event object — full autosubscription, no channel setup needed.
- Full control over aggregation and correlation logic (RequestIdIdentifier).
- Easiest place to iterate QuoteToOrderErrorDetailEvents.
- Requires code maintenance and test coverage.

Flow / Process

- Record-Triggered Flow (or legacy Process Builder) on the platform event object.
- Low-code path, officially supported as a subscriber.
- Fast to prototype simple success/error rules.
- Aggregating multiple conversions tends to need an extra control element (a counter field or record).

Pub/Sub API & Streaming API (CometD)

Pub/Sub API

- Modern API built on gRPC / HTTP2; payload in Avro binary format
- Built-in flow control and publish acknowledgments
- Best fit for external, high-volume integrations with strict SLAs
- Delivery is at-least-once with duplicates possible

Streaming API (CometD)

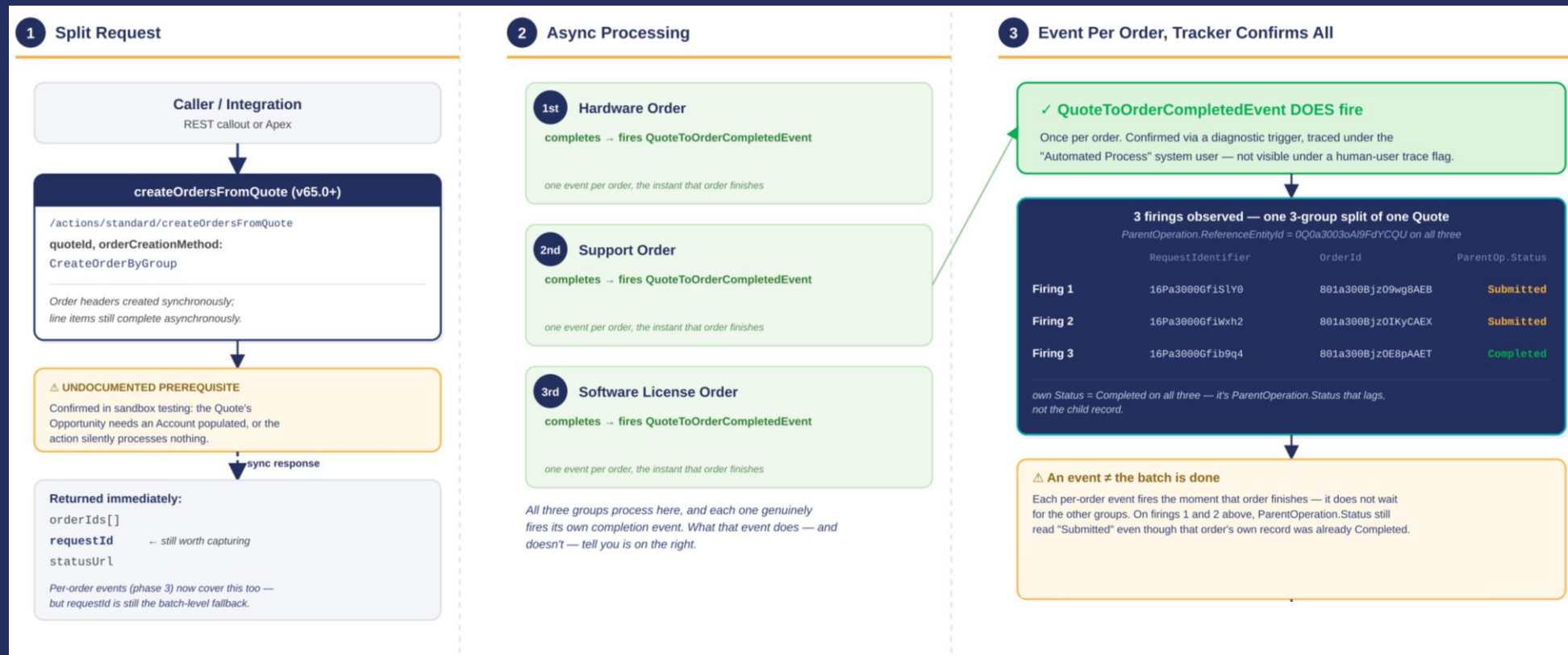
- Legacy Bayeux/CometD protocol; payload in JSON
- Simpler to stand up for lightweight clients (browser, scripts, EMP API)
- Good fit for internal dashboards or simple real-time notifications
- Also at-least-once; no global ordering guarantee

Testing Findings

TESTING FINDINGS

CreateOrdersFromQuote (v65+)

Confirmed in Sandbox testing: **QuoteToOrderCompletedEvent** *fires for each Order*, however querying AsyncOperationTracker is the only way to confirm all orders have completed successfully

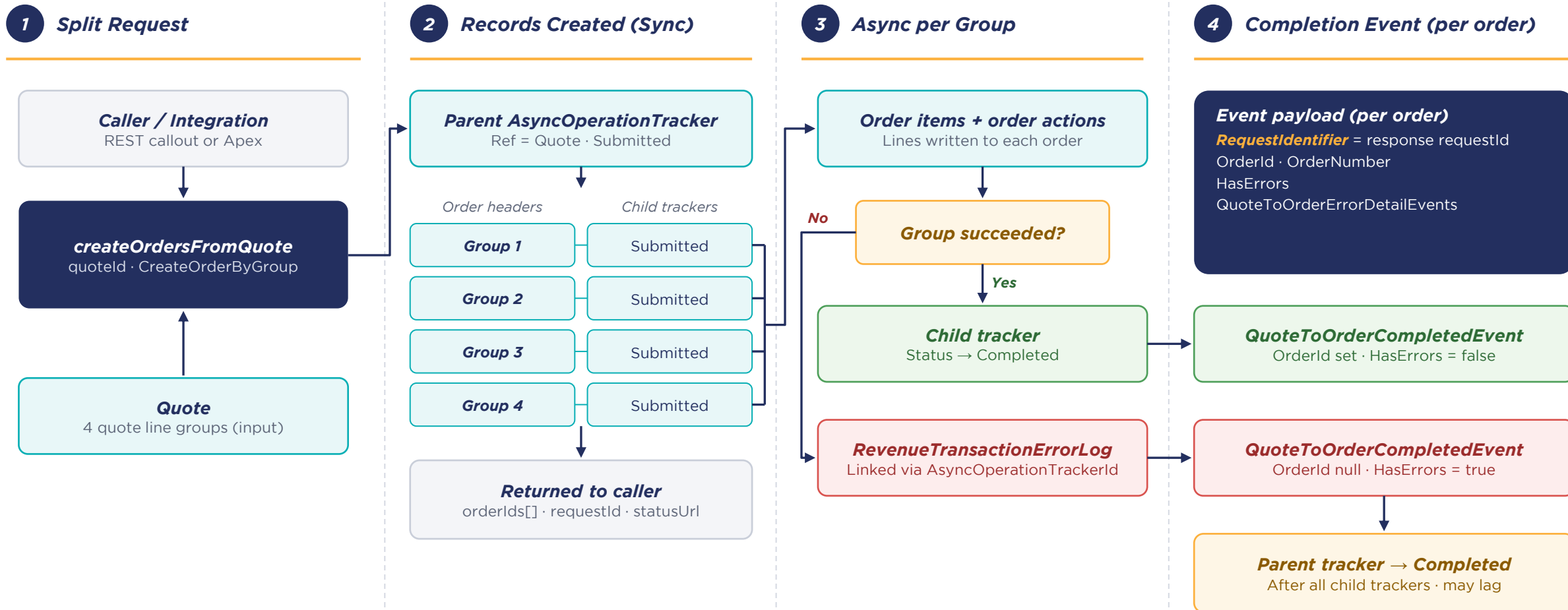


TESTING FINDINGS



CreateOrdersFromQuote: Record Lifecycle

From the split request to **QuoteToOrderCompletedEvent**: every record created along the way

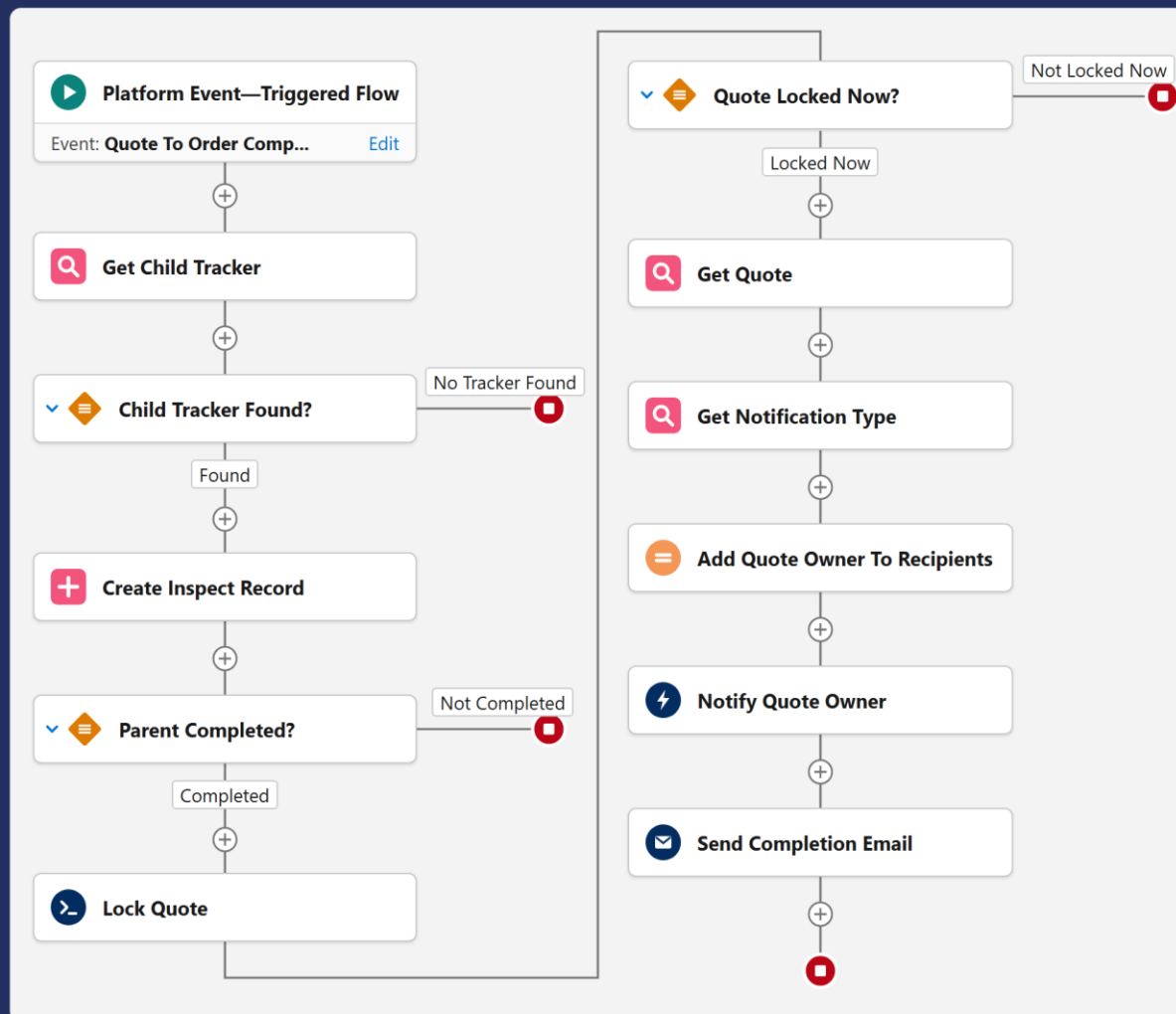


EXAMPLE IMPLEMENTATION



Use Flow to Trigger off each Platform Event

Trigger individual actions for each order, and or final action once all orders has been successfully activated i.e. locking the quote



Demo

Use Case

One Quote with 3 addresses or Quote Line Recipients is converted to Order via Create Orders by Recipients, resulting in 3 Orders. Once the final Order is generated, we lock the Quote to block the user from changing the Quote which may then make the Quote and Orders out of sync.

Business Case: Lock quote on initiating quote to order + In-App + Email notification to sales user that process has completed

[GitHub project with source code](#)

Demo:

- Walkthrough of configuration and code as a developer
- Demo of run-time behavior of creating a Quote, adding Recipients, converting to Order, and resulting tracker object records created, quote locked, in-app and email notifications sent to user

Thank you