

Part 1- Using Postman to call APIs

Getting your environment variables setup and doing the authentication

Using Postman to Call Fenergo APIs

- Download and Install Postman <https://www.postman.com/downloads/>
- Create a new project.
- Create an Authentication API Call
- Configure Variables so the Access Token and Tenant Id is available to other API calls
- Make a Fenergo API Call to Query a Legal Entity.

Some Required Details

- Client Credential
 - Client Id
 - Client Secret
 - Scope
- Tenant Id
- The URL endpoint for the Fenergo Identity Provider
- The URL endpoint for the Fenergo API Gateway

- Created 2 API Methods in Postman
- Authorization Request
- Parsed the response and set some Postman Variables
- Made a QUERY Api call to retrieve Legal Entity Data

Part 2 - Creating an Entity with APIs

On the Fenergo SaaS Platform

George McGrane

Using Postman to Create an LE

- Before this – Review the video to Make API calls with Postman.
- Look at the Create Legal Entity Swagger Document Specification
- Take the boilerplate request and edit it
- Make the API call
- View the LE record created on the Fenergo UI

Look at the Redoc Spec

- The API document we want to look at is available on the Fenergo Document Portal at: <https://docs.fenergox.com/api-docs/entitydata-command-v2>

The screenshot displays the Redoc API documentation for the 'Create entity' endpoint. On the left, a sidebar shows a search bar and a list of API endpoints under the 'Entity' category, including 'Create entity', 'Create entity with full record in response', 'Create url to upload file', and 'EntityDraft'. The main content area is titled 'Create entity' and lists two bullet points: 'Creates a new entity.' and 'Returns the id of the created entity.' Below this, it specifies 'Required permissions: EntityDataEdit, EntityGroupManagementEdit'. The 'AUTHORIZATIONS' section indicates 'Bearer' authentication. The 'HEADER PARAMETERS' section lists two required parameters: 'X-TENANT-ID' (string, required, example: 'b11f8be3-f29b-4959-8964-956d4af7c468', description: 'The UiD of the tenant representing organization') and 'X-CORRELATION-ID' (string, example: 'fee5b5f5-b220-4fef-ba38-bf802b98d74d', description: 'The UiD of the correlation id'). On the right, the 'Request samples' section shows a 'POST' request to '/api/v2/entity' with a 'Payload' button. Below the button, the 'Content type' is set to 'application/json'. A 'Copy' button is present, and the payload is displayed in a dark-themed code editor with syntax highlighting.

```
{
  "data": {
    "type": "Individual",
    "targetEntity": "Related Pe
+ "properties": { ... },
+ "dataSources": { ... },
    "policyJurisdiction": "Glot
+ "policyJurisdictions": [ ...
    "alternateId": "1124efd2-34
+ "category": [ ... ],
+ "accessLayers": { ... }
  }
}
```

- Took the default sample request from the API Specification.
- Broke it down and simplify the request.
- Made the API Call to Create the new Legal Entity
- Looked at the Entity in the Fenergo User Interface
- Retrieved back the Entity with the API.

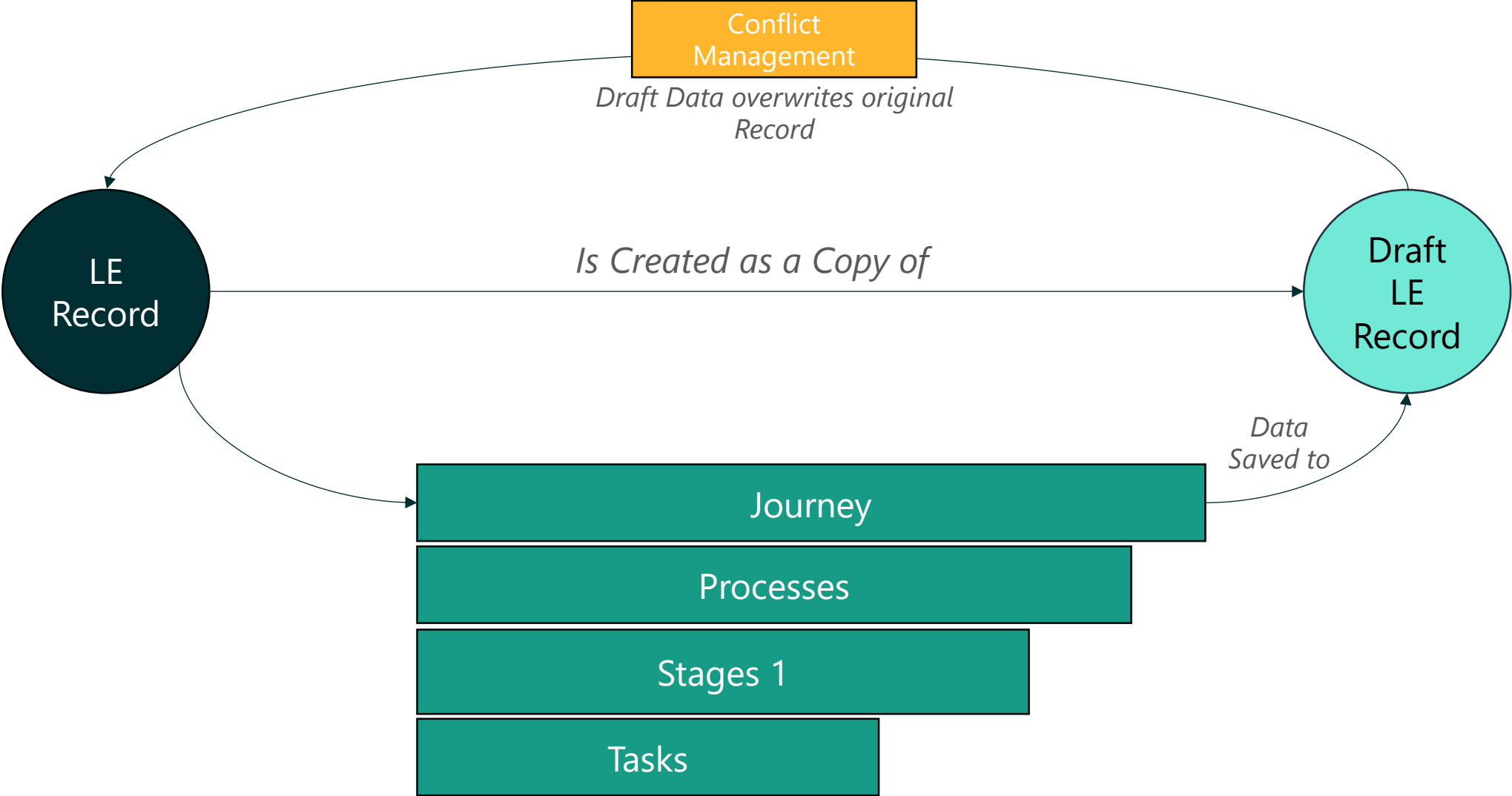
Part 3 - Creating a Journey and Draft LE Record with APIs (explaining that pattern)

George McGrane

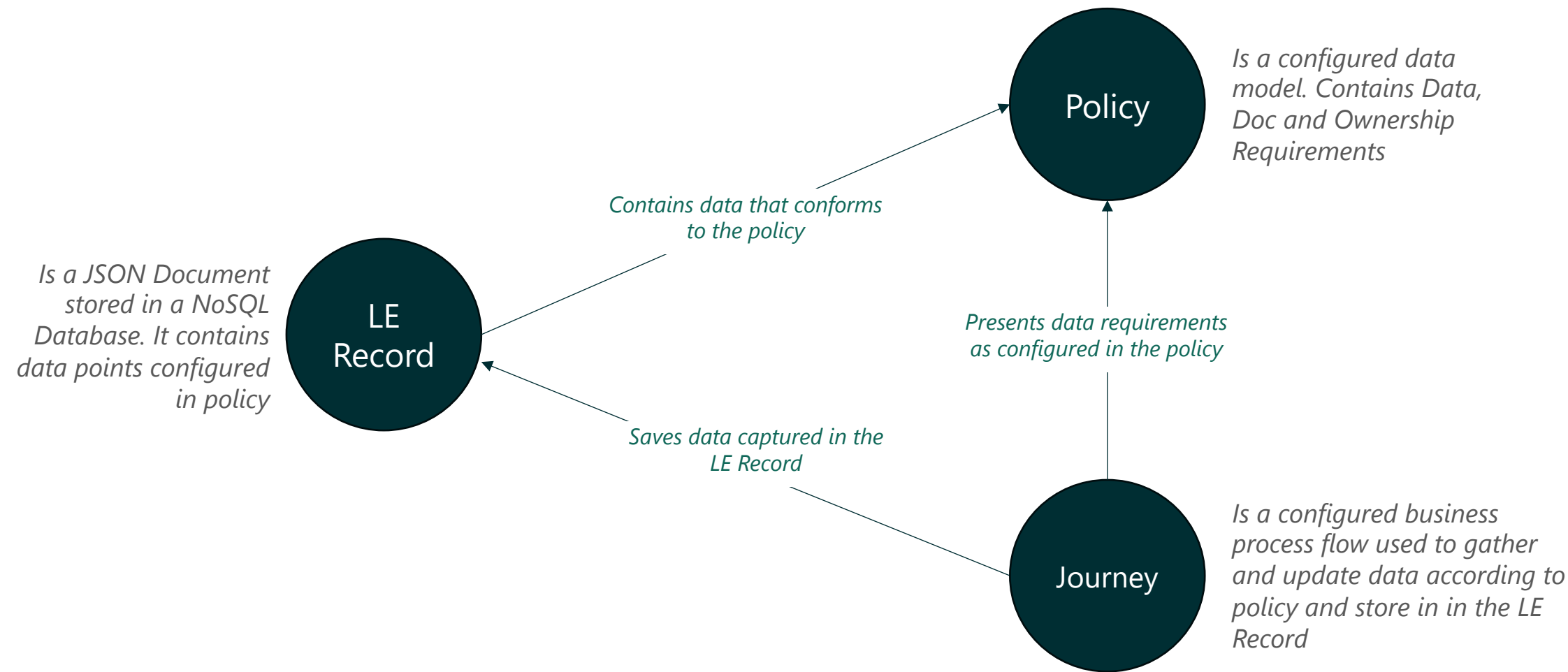
Using Postman to Create a Journey and Draft LE Record

- Before this – Review the video to Make API calls with Postman.
- Understand what the pattern means in the context of managing data and process on the Fenergo SaaS Platform.
- Understand the relationship between LE Data, Policy and Journey
- Look at the Create Journey API Request
- Look at the Create Draft LE Data API Request
- Update the draft record and compare it against the original.

Understanding the Pattern



Relationship between LE Data, Policy & Journey





Policy

Journey

- When we use the term “Scope” it relates to “Which” Policy or Journey we want to work with.
- Clients can have multiple Policy's and Journey Types.
- Configuring Scoping Rules dictates which one gets used and when.
- We want to make an API call that creates a Journey for a given Legal Entity. We must know which one to use so we can do that via APIs.

Let's Look at the Redoc Specs

- The API Specifications we want to look at are available on the Fenergo Document Portal.
- *Journey Logic Engine* : <{baseUrl}/api-docs/journey-engine-v2>
- *Journey Command* : <{baseUrl}/api-docs/journey-command>
- *Entity Data Command* : <{baseUrl}/api-docs/entitydata-command-v2>
- *Entity Data Query* : <{baseUrl}/api-docs/entitydata-query-v2>

- Created a new Verified LE
- Looked at the available Journeys in Scope
- Created a Journey Instance for the Legal Entity
- Created a Draft Record and associated to the Journey Instance
- Updated the Draft Record
- Retrieved back the Draft Record to see the changes

Part 4 - The Eventing Pattern

Events and Working with Webhooks on Fenergo

George McGrane

'Event'

- *a thing that happens or takes place.*
- *occurrence of something important or noteworthy*

'Webhook'

- *web based call-back mechanism.*
- *a means to notify a 3rd party system of an event via an api*

- Before this – Review the video on working with Entity Data Journeys & Draft LEs.
- Understand what is meant by the eventing pattern including what is an event and what is a webhook.
- Look at creating test webhook endpoints in the UI and API.
- Generate some events and look at the content.
- The role of a Correlation Identifier.
- Understand the HMAC signature and validating event messages.

- What Does an Event Notification Look Like?

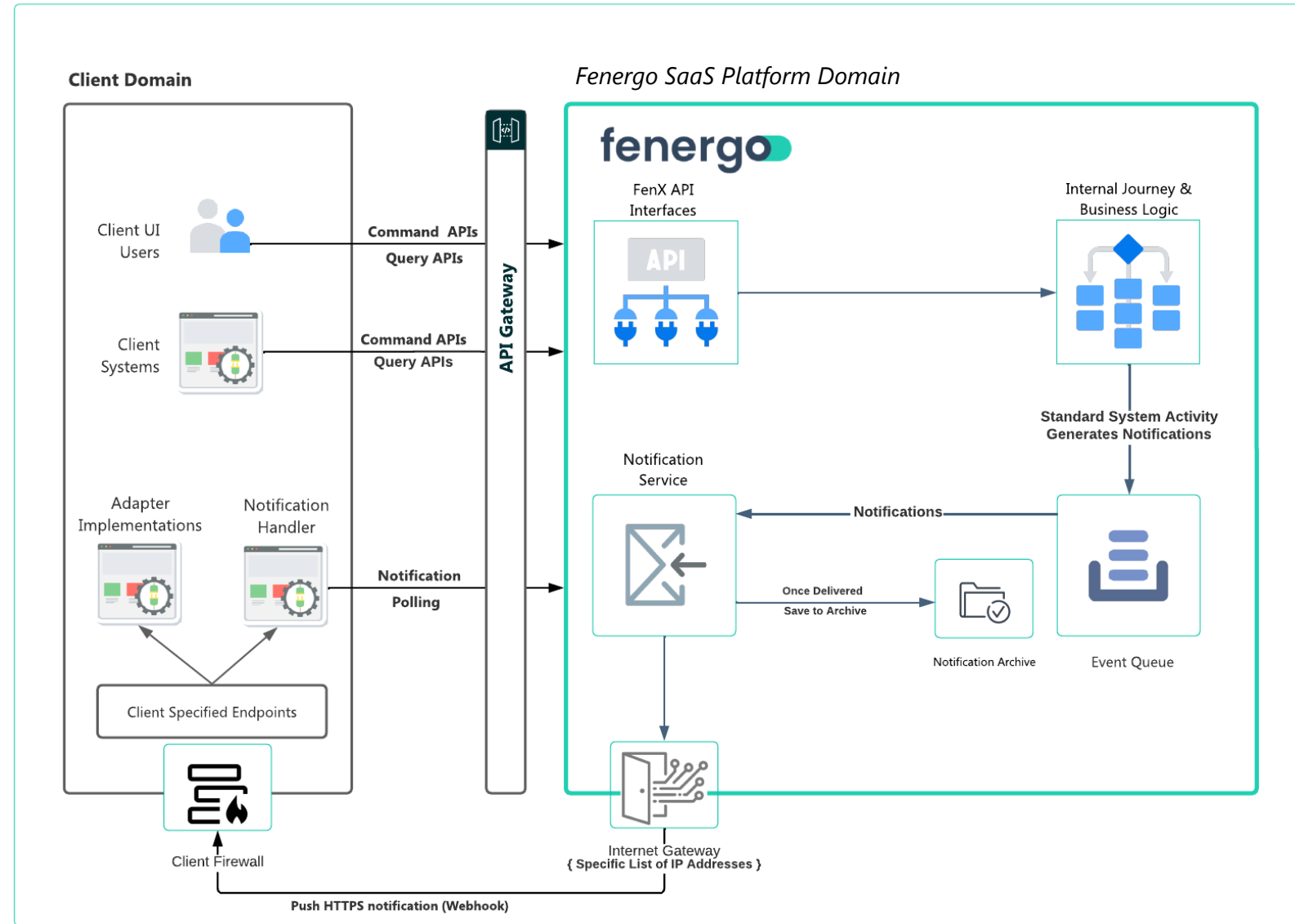
- It is a simple JSON message

- Contains all the details required to trace the source of the ' **Event** '

```
{  
  "Id": "c9843b58-4bf5-54d5-8ecd-487b8b495d08",  
  "TenantId": "XXXXX-xxxx-xxxx-xxxx-XXXX",  
  "EventType": "entitydata:draftupdated",  
  "RelativeUrl": "entitydataquery/api/entity/ebd6f93d-db65-4352-9774-841deb6e185e/draft/a7fb46df-d474-49d8-83bf-1b97fa063f06",  
  "Payload": {},  
  "When": "2024-04-08T10:55:31+00:00",  
  "CorrelationId": "54a11189-1b6c-4449-9b6e-bdd7a450ecc7",  
  "CausationId": "93f6a1f2-6934-4a7a-9d1a-258cfd8d0ecb"  
}
```

The 'eventing' pattern and what it does.

- Events are generated as client UI or system users interact with the platform.
- When an event happens, a "Notification" of that event is generated.
- Clients have two options for working with these notifications.
- Clients can Poll a **polling endpoint** or have notifications delivered to a **webhook destination**.



Events and Working with Webhooks

- Full Details are covered on the Developer hub. The APIs will be looking at:
- *webhooks API: `{baseUrl}/api-docs/eventnotifications-webhooks`*

The screenshot displays the API documentation for the 'Webhooks' endpoint. On the left, a sidebar lists various actions: 'List all webhook configurations' (GET), 'Add a webhook configuration' (POST), 'Delete webhook configuration' (DEL), 'Update a webhook configuration' (PUT), 'Get webhook configuration' (GET), 'List all available event notification types' (GET), and 'Test webhook availability' (GET). The main panel shows details for the 'List all webhook configurations' endpoint. It includes a description: 'This method will list all webhook configurations for specific tenant'. Under 'Required permissions', it states 'Following permissions are required: WebhookAccess'. The 'AUTHORIZATIONS' section shows 'Bearer'. The 'HEADER PARAMETERS' section lists 'X-TENANT-ID' as a required string parameter, with an example value: 'b11f8be3-f29b-4959-8964-956d4af7c468'. On the right, a 'Response samples' section shows a dropdown for 'Content type' set to 'text/plain' and a 'No sample' message.

- **note – not a command / query pattern.*

Webhook Signature

- Signature allows the recipient to VERIFY the sender.
- Signature can ONLY be calculated using "HMACSHA256" with the Shared Secret and the full webhook HTTP Body.

Headers

connection	close
content-length	412
content-type	application/json
x-amzn-trace-id	Root=1-6613cd23-bd45943565249399a99a5f4b; Sampled=0
x-fenx-signature	sha256=E3F4416F1640323B6EC48341EDC0E87FAFF9BD07DF74CAB5..
host	webhook.site

**example for another day 😊*

Using the ' *CorrelationId* '

- A **Correlation Identifier** is something that can be used to trace the **source** of an '*event*' to the notification generated and sent, after that '*event*' occurs.
- Add a correlation id to your API (such as Update Draft Entity) and that same Id will be sent back as the Correlation Id in the body of the Event Notification
- Let's add the `x-correlation-id` header to our API call and see it being returned in the webhook.

- Looked at what is meant by the *eventing pattern*, what an event is and what a webhook is.
- Created a test webhook endpoint using the API.
- Generated some events using other API calls and looked at the content of the event notifications generated.
- Explored the use of a Correlation Identifier.
- Looked at a HMAC signature and validating event messages.
(more to come on this).