Platform Developer Roadmap: Leveraging new capabilities within AVEVA's Industrial Platform

Ecosystem Track

Joshua Kidd, Scott Dunham
Agenda

• What is new or in development on AVEVA Data Hub for developers

• Why the capabilities were built

• How will you interact with these capabilities
New data types and query patterns in AVEVA Data Hub

Event Data Store
Ability to store events with surrounding context and provide a rich contextual search API for retrieving the information.

Data Views supports Parquet
Ability to query data sets in a format familiar and common to data scientists.

Change Broker
Ability to sign up for and query changes to stream data.
Solid foundation to expanding the industrial data footprint

Process Data: Streams

- Timestamp
- Depth
- Pressure
- Speed
- Latitude
- Longitude
- Quality information

<table>
<thead>
<tr>
<th>Process Data: Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
</tr>
<tr>
<td>Depth</td>
</tr>
<tr>
<td>Pressure</td>
</tr>
<tr>
<td>Speed</td>
</tr>
<tr>
<td>Latitude</td>
</tr>
<tr>
<td>Longitude</td>
</tr>
<tr>
<td>Quality information</td>
</tr>
</tbody>
</table>

Events & Production Context: Events & Reference Data

Engineering & Operations Data: 1D, 2D, 3D

AVEVA Data Hub

Asset Context: Assets

© 2023 AVEVA Group plc and its subsidiaries. All rights reserved.
GraphQL

- A query language that provides a simple declarative way to retrieve data by accepting a nested object structure as a query rather than a text string
- A GraphQL schema is composed of types and operations {queries, mutations}

```graphql
type BatchEvent {
  id: ID!
  name: String!
  description: String
  startTime: DateTime!
  endTime: DateTime
  duration: TimeSpan
  state: EventState
  asset: Asset
  unitOperationEvents: [UnitOperationEvent]!
}

query {
  queryBatchEvents(
    where: { startTime: { gt: "2023-09-01 11:00:00" } },
    options: { size: 10, sort: { startTime: "DESC" } }
  ){
    id
    startTime
    duration
    state
    asset {
      id
      typeId
      streamReferences {
        name
      }
    }
  }
}
```
Tools: Python Notebook and Data Hub’s GraphQL Console
Demo Scenario: Batch Manufacturing Process

GraphQL Schema

- **Reference Data Types (Context)**
  - **Material**: represents a material type definition. An instance may be created to define a raw material, process intermediate, or a finished product.
  - **Recipe**: defines the equipment, materials and process steps required to manufacturer a product.
  - **UnitOperation**: an individual process within a recipe.
  - **Component**: defines the sequence and target quantity of a raw material ingredient.

- **Event Types**
  - **Batch**: The complete execution record of a recipe instance that manufactured a material lot. Includes specifics about the equipment used, raw materials consumed, and process steps executed.
  - **UnitOperationEvent**: An individual process step executed as part of a batch. Includes the equipment used and the raw materials consumed.
  - **ConsumptionEvent**: Event representing the consumption of a raw material component within a unit operation event
  - **Lot**: An event the models a group of material containers that share a common lot number.
  - **Alarm**: An event representing a process deviation or anomaly.
Demo:
Create a Graph Schema, Upsert Data, and Perform Queries
Event Graph

Events model meaningful observations at a specific point in time or over a span of time. Events can serve as bookmarks for your process data, placing it in context and allowing you to explore a variety of scenarios:

- What was my process doing during the event?
- What happened leading up to the event? After?
- Which events occur most frequently?

**Step 1: Checkout Data Hub’s GraphQL Console**

Browse to Data Hub’s new GraphQL Console
Step 3: Creating Types and Visualizing the Graph Schema

Reference Data (Context)

- **Material**: represents a material type definition. An instance may be created to define a raw material, process intermediate, or a finished product.
- **Recipe**: defines the equipment, materials, and process steps required to manufacture a product.
- **UnitOperation**: an individual process within a recipe.
- **Component**: defines the sequence and target quantity of a raw material ingredient.

Events

- **Batch**: The complete execution record of a recipe instance that manufactured a material lot. Includes specifics about the equipment used, raw materials consumed, and process steps executed.
- **UnitOperationEvent**: An individual process step executed as part of a batch. Includes the equipment used and the raw materials consumed.
- **ConsumptionEvent**: Event representing the consumption of a raw material component within a unit operation event.
- **Lot**: An event the models a group of material containers that share a common lot number.
- **Alarm**: An event representing a process deviation or anomaly.

Create a Function that returns a TypeProperty from Json

```python
# create type properties from json
def type_properties_from_json(data):
    return TypeProperty(
        PropertyTypeCode=prop("PropertyTypeCode"), upper(),
        prop("id"),
        flags=PropertyTypeFlags(prop("flags")) if "flags" in prop else None,
        property_type_id=prop("property_type_id") if "property_type_id" in prop else None,
        remote_reference_name=prop("RemoteReferenceName"),
    ) for prop in data
```

Load and Create Reference Data Types

```python
# create reference data types
reference_data_types = load_json( '../config/reference-data-types.json')
for item in reference_data_types:
    # create property
    reference_data_type_properties = type_properties_from_json(item("Properties"))
```
Step 5: Simulate and Visualize Event Batch Data

Simulate Raw Material and Product Lots, Batches, UnitOperationEvents, and Material Consumption Events

```python
[116] lots = []
raw_materials = {}
def get_raw_material_lot(material_id, current_date):
    if material_id in raw_materials:
        lot = raw_materials[material_id]
        expiration = datetime.fromisoformat(lot['eventEndTime'][1:3])
        if expiration <= current_date:
            return lot
    # otherwise create a new lot
    lot_number = 'LOT-{:02d}'.format(material_id + 1)
    expiration = current_date + timedelta(days=random.uniform(1, 20))
    lot = {
        'id': lot_number,
        'number': lot_number,
        'eventStartTime': current_date.isoformat()[:19],
        'eventEndTime': expiration.isoformat()[19:],
        'material': (material_id)
    }
    # capture current lot in catalog of raw materials
```
Step 4: Create and Visualize Reference Data

```python
# bulk create reference data
reference_data = load_json('./config/reference-data.json')
for item in reference_data:
    client.ReferenceData.getOrCreateReferenceData(namespace_id,
    item["type"], json.dumps([item["items"]]))

query = '''
query {} {
  queryRecipe {
    id
    name
    product {
      id
      name
    }
    unitOperations(options: {
      sort: {
        sequence: ASC
      }
    }) {
      id
      name
      assetTypeId
      sequence
      duration
      billedMaterials(options: {
        sort: {
          sequence: ASC
        }
      }) {
        id
        name
        quantity
        sequence
      }
    }
  }
}
...'''
```
Visualize Raw Material Lots Consumed by Batches

```python
[1]:
  ```
  ... 
  [add query]
  ...
  consumedLots = client.GraphQL.execute(query=namespace_id, query=query)
  file_name = "consumedLots.html"
  network = visualize_graphql(consumedLots['data']['events'], query='batch', file_name)
  network.show(file_name)
```

Step 6: Advanced Queries

Lot Genealogy: Retrieve Recipes from Raw Material Lot Consumed

Lot -> ConsumptionEvent -> UnitOperation -> Batch -> Lot

```python
[128]:
  ```
  {
    events { 
      queryLot(where: { 
        number: { 
          eq: "LOT-Glucose-2023-05-07T06:00:00Z"
        } 
      }) { 
      id 
      number 
      consumptionEvents { 
        id 
        name 
        unitOperationEvent { 
          id 
          name 
          batch { 
            id 
            number 
  ```
Step 6: Advanced Queries

Lot Genealogy: Retrieve Recipes from Raw Material Lot Consumed

Lot -> ConsumptionEvent -> UnitOperation -> Batch -> Lot

```python
query="...
[add query]
...

consumed_lots = client.GraphQL.executeQuery(namespace_id, query=query)
file_name = "genealogy.html"
network = visualize_graphql(consumed_lots['data']['events'], 'query1', file_name)
network.show(file_name)
```
A GraphQL query or mutation has not yet been run. Use the editor or GraphQL Explorer to author one and press Send Request to retrieve GraphQL data.
New data types and query patterns in AVEVA Data Hub

Event Data Store

Ability to store event data and provide a rich contextual search API for retrieving the information

Data Views supports Parquet

Ability to query data sets in a format familiar and common to data scientists

Change Broker

Ability to sign up for and query changes to stream data
Data Views supports a variety of formats

- Object-style JSON
- Table-style JSON
- CSV
- Apache Parquet (new!)
Demo: DataViews Apache Parquet Format
Data Views

Import Dependencies

```python
%pip install pandas
%pip install pyarrow
```

```python
import pandas as pd
import requests
import json
from datetime import datetime, timedelta
import io
```

Verify Pandas Parquet Engine

```python
pd.io.parquet.get_engine('auto')
```

Load Application Settings
New data types and query patterns in AVEVA Data Hub

Event Data Store
Ability to store event data and provide a rich contextual search API for retrieving the information

Data Views supports Parquet
Ability to query data sets in a format familiar and common to data scientists

Change Broker
Ability to sign up for and query changes to stream data
Industrial operations data is always changing...

- Live data
- Uploads of manual measurements
- Recalculations
- Edits of incorrect readings

... and there are consequences downstream

- Less accurate predictions and remote monitoring
- Inaccurate combined data sets for BI reporting
- Service provider recommendations based on incomplete data
Change data can impair and complicate solutions
Customers and Partners surfaced common challenges

- Reconciling destination with source using large queries
- Taking on risk by assuming no changes in data
- Managing unique data sharing solutions for each trusted partner
Changes in Streams’ data can be queried in AVEVA Data Hub
Use an API route to create Signups for change data on Streams

*(In Preview)*

Stream A:
**Update:**
2023-06-15 08:00:00, 86.1

Stream B:
**RemoveWindow:**
start: 2023-06-12 07:00:00
end: 2023-06-12 08:20:00

Stream C:
**Replace:**
2023-06-14 08:12:00, 25.9
2023-06-14 08:13:00, 23.7

2023-06-15 08:00:15
Create a *Signup* with Stream IDs from a *Namespace / Community*

2023-06-15 08:00:35
*Query* to obtain change data since *Signup* creation
Keep up with your data as it arrives

(In Preview)

1. Definition of Signup lasts for up to 24h
2. Change data retained for an hour before it expires
3. Signup’s expiration time renewed to another hour

Create a Signup with Stream IDs from a Namespace

Query to obtain change data since Signup creation
Demo:
Create and query a Signup
AVEVA™ Data Hub - Change Broker Demo

1. Query an Asset to get its Stream IDs
2. Use some of the Asset's Stream IDs to create a Signup
3. Query the Signup for change data
4. Edit the Signup to include all the Stream IDs for the Asset
5. Confirm the Signup's definition is updated
Demo:
Feed an External App, Algorithm, Notice
AVEVA™ Data Hub - Change Broker Demo 2

1. Configure your External App to create a Signup

2. Query the Signup for change data

3. Configure Algorithms or threshold Alerts or Notifications
New data types and query patterns in AVEVA Data Hub

Event Data Store
Ability to store event data and provide a rich contextual search API for retrieving the information

Data Views supports Parquet
Ability to query data sets in a format familiar and common to data scientists

Change Broker
Ability to sign up for and query changes to stream data

In Preview!

Released!

In Preview! Dec Release
Enabling developers on AVEVA Data Hub

• Update and enhance API Console to support new capabilities
• Release sample code to simplify getting started in various languages
• Support common data formats that integrate into a broader technology ecosystem
Join us in a Lighthouse Project to prove & shape the value of new technology for your business scenarios!

**AVEVA Lighthouse Program**

**AVEVA provides:**
- AVEVA pre-released software
- Installation & configuration support
- Technical R&D support
- Program management

**Participating customer provides:**
- Viable scenario
- Resources to use AVEVA pre-released software for your scenario
- Product feedback on use of software for scenario
- Willingness to document a success story and participate in a future AVEVA public presentation

*Email us at lighthouse@aveva.com to engage with our team!*
Where to find more information

Overview & Resources


Documentation

Joshua Kidd

Software Architect

• AVEVA Solutions Limited
• joshua.kidd@aveva.com

Scott Dunham

Staff Technical Product Manager

• AVEVA Solutions Limited
• scott.dunham@aveva.com
Questions?

Please wait for the microphone.
State your name and company.

Please remember to...

Navigate to this session in the mobile app to complete the survey.

Thank you!
This presentation may include predictions, estimates, intentions, beliefs and other statements that are or may be construed as being forward-looking. While these forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could result in actual outcomes differing materially from those projected in these statements. No statement contained herein constitutes a commitment by AVEVA to perform any particular action or to deliver any particular product or product features. Readers are cautioned not to place undue reliance on these forward-looking statements, which reflect our opinions only as of the date of this presentation.

The Company shall not be obliged to disclose any revision to these forward-looking statements to reflect events or circumstances occurring after the date on which they are made or to reflect the occurrence of future events.
ABOUT AVEVA

AVEVA is a world leader in industrial software, providing engineering and operational solutions across multiple industries, including oil and gas, chemical, pharmaceutical, power and utilities, marine, renewables, and food and beverage. Our agnostic and open architecture helps organizations design, build, operate, maintain and optimize the complete lifecycle of complex industrial assets, from production plants and offshore platforms to manufactured consumer goods.

Over 20,000 enterprises in over 100 countries rely on AVEVA to help them deliver life's essentials: safe and reliable energy, food, medicines, infrastructure and more. By connecting people with trusted information and AI-enriched insights, AVEVA enables teams to engineer efficiently and optimize operations, driving growth and sustainability.

Named as one of the world’s most innovative companies, AVEVA supports customers with open solutions and the expertise of more than 6,400 employees, 5,000 partners and 5,700 certified developers. The company is headquartered in Cambridge, UK.

Learn more at www.aveva.com