Platform Developers: Putting it all together with a demo + Q&A with experts!

Derek Endres & Yuki Kimura
Derek Endres
Sr. Manager Product Readiness Guild
• AVEVA
• Derek.Endres@aveva.com

Yuki Kimura
Principal Software Developer
• AVEVA
• Yuki.Kimura@aveva.com
AGENDA

• Data Ingress
  • SDS Type Definition
  • SDS API vs OMF
  • Best Practices
  • DEMO!!!!

• Data Egress
  • Streams vs Assets vs Data views
  • DEMO!!!!
The Power of AVEVA™ Connect

AVEVA Connect enables a hybrid data architecture through cloud offerings

- Remote operations monitoring
- Data science & advanced analysis
- Partner Applications
- Data sharing with trusted ecosystem partners

AVEVA Connect Industrial Cloud Platform

AVEVA Data Hub

- Open RESTful API’s

- AVEVA PI Server
- Edge Data Store
- Remote assets
- AVEVA Historian
- AVEVA Edge
- AVEVA System Platform

© 2022 AVEVA Group plc and its subsidiaries. All rights reserved.
A cloud-native industrial platform designed for aggregating, storing, enriching, accessing, analyzing, and securely sharing real-time operations data from historians, edge devices, and more.

- Managed, secure, multi-tenant platform
- Operated & maintained by AVEVA
- High speed, scalable, elastic, & resilient
- Modern, secure REST APIs
- Built & deployed on Microsoft Azure
AVEVA Data Hub

Remote monitoring
Data science & AI/ML platforms
3rd party analytic tools
Data sharing with business partners
Custom & partner applications
Reporting & Dashboards

A cloud-native industrial platform designed for aggregating, storing, enriching, accessing, analyzing, and securely sharing real-time operations data from historians, edge devices, and more

- Managed, secure, multi-tenant platform
- Operated & maintained by AVEVA
- High speed, scalable, elastic, & resilient
- Modern, secure REST APIs
- Built & deployed on Microsoft Azure

Supported Regions
West US (California)
North Europe (Ireland)
Australia East (New South Wales)
Sequential Data Store
Flexible **Sequential Data Store** that keeps related data together.

Stream types & streams

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

**IR Camera**

[24.56, 24.58, ...]

Limited functionality / custom consumer application required for this functionality.
Complex vs Simple Types

What is a simple type?

- Key-Value pair
- Often Timestamp Value
- Adapters
- PI Server
- AVEVA Historian

```json
{
"Timestamp": "2023-07-17T21:12:08.1915788Z",
"Temperature": 0
}
```

```json
{
"Timestamp": "2023-02-08T02:51:58.707077Z",
"Value": 0,
"IsQuestionable": false,
"IsSubstituted": false,
"IsAnnotated": false,
"SystemStateCode": null,
"DigitalStateName": null
}
```
Complex vs Simple Types

What is a complex type?

- Key with multiple properties
- Compound indices
- Nested types

{ "Timestamp": "2022-02-03T16:09:27763Z", "ParticleCount0.3": 408, "ParticleCount0.5": 134, "ParticleCount1.0": 17, "ParticleCount2.5": 4, "ParticleCount5.0": 2, "ParticleCount10.0": 2, "PM1.0S": 2, "PM1.0E": 2, "PM2.5S": 2, "PM2.5E": 2, "PM10S": 5, "PM10E": 5 }
Complex vs Simple Types

Which should I choose?

• All types should have a value for each property on each event
  • If an event can and will occur without every property it should be broken apart or interpolation will not be valid

• Stream views can be used to view a type in a different way or to permanently retype a stream
# Ingress and REST APIs

## Diagram

![AVEVA Data Hub Diagram](image)

<table>
<thead>
<tr>
<th>Web Portal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected Community</td>
</tr>
<tr>
<td>Trending</td>
</tr>
<tr>
<td>Context</td>
</tr>
<tr>
<td>Sequential Data Store</td>
</tr>
<tr>
<td>Ingress</td>
</tr>
</tbody>
</table>

**Security**

- Power BI Connector
- REST APIs
SDS API vs OMF

What is SDS API?

• Sequential Data Store
  • Purpose-built sequential data storage for AVEVA Data Hub and Edge Data Store

• Terminology
  • Types
  • Streams
  • Events

• Send multiple events to a stream

```json
[
  {
    "Timestamp": "2023-07-17T21:12:08.1915788Z",
    "Value": 0
  },
  {
    "Timestamp": "2023-07-17T21:15:00.6197394Z",
    "Value": 10
  },
  {
    "Timestamp": "2023-07-17T21:20:00.7899553Z",
    "Value": 30
  }
]
```
SDS API vs OMF

What is OMF?

- Open Message Format
  - Generic format that can be interpreted by PI, ADH and EDS
- Terminology
  - Types
  - Containers
  - Data values
- Send multiple data values to multiple containers

```
[{
    "containerid": "Tank1_PressureMeasurements",
    "values": [
        {
            "Pressure": 12.0
        },
        {
            "Pressure": 11.5
        }
    ]
}, {
    "containerid": "Tank2_PressureMeasurements",
    "values": [
        {
            "Pressure": 14.0
        },
        {
            "Pressure": 15.1
        }
    ]
}]
```
SDS API vs OMF

Similarities and differences

• SDS
  • Single Stream
  • Immediate response/validation
  • EDS and ADH only

• OMF
  • Multiple containers
  • Minimal validation
  • Can send to PI, EDS and ADH

• Simple and Complex stream definitions *
• Non-time series and compound indices*
SDS API vs OMF

What data ingress do I choose?

• Adapter, PI Server, AVEVA Historian, AVEVA System Platform, Edge Data Store
• OMF for ingress applications
• SDS for situations where you are reading SDS data and writing it back into a few streams
Ingress and REST APIs
Data Ingestion Best Practices

- In-order data (increasing, for the stream)
- SDS Type reuse
- UOMs can be set on type and/or streams
- Naming patterns
Create Metadata from Stream name patterns with Metadata Rules

STREAM NAME
BAC.101313.Phase.PV
BAC.101413.Phase.PV

Metadata RULES

BAC identifies the adapter name
101313 identifies the equipment number
Phase identifies the measurement
PV identifies the value
Create Assets from Stream name patterns with Asset Rules

STREAM NAME
DNP3-1.WindTurbine03.RotorSpd.0
DNP3-1.WindTurbine03.RotorDeflection.0
DNP3-1.WindTurbine04.RotorSpd.0

Asset RULES
WindTurbine03 identifies the asset name
RotorSpd identifies the Rotor Speed property
Use Case – define problem

Ingressing data from particle sensors

- Small IoT devices running on Linux
- Multiple devices in multiple locations
- Need custom code to read the values, send to AVEVA Data Hub (Sequential Data Store) as streams
- Preference to develop in .NET Core
DEMO – Ingress

Steps

• Start with Github sample: https://github.com/AVEVA/sample-adh.omf_ingress-dotnet
  • Simplify by deleting OMF connection creation logic
    • Assume OMF connections and clients have been created
  • Delete logic to clean up OMF connections, SDS types and streams

• Replace Main() method as follows:
  • Read constants from appsettings.json to configure app
  • Send OMF type and container messages
  • Setup timer loops for collecting and sending data
  • Add sensor-specific code to collect the data and add to a queue
  • Send data from queue
This organization is under construction

AVEVA Samples

AVEVA is a proven leader in enabling operation intelligence. In this GitHub repo, we provide navigation to sample repos which will help you get started with using AVEVA technology. The samples are intended to help you get started and are not production applications and libraries.

The official AVEVA samples are organized by technology and accessible through the following table, note you can always utilize GitHub search to find a repo too:
using System;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Extensions.Configuration;
using OSIsoft.Data.Http;
using OSIsoft.Identity;
using OSIsoft.OmfIngress;
using OSIsoft.OmfIngress.Models;

namespace OmfIngressClientLibraries {
  public static class Program {
    public static string Resource { get; set; }
    public static string TenantId { get; set; }
  }
  private static IConfiguration _config;
  private static IDevice _omfDevice;
  private static IOmIngressService _omfIngressService;
  private static int _timeout = 2 * 60 * 1000;
}
```csharp
public static void Main()
{
    SetupConfiguration();
    SendTypeAndContainerAsync().GetAwaiter().GetResult();
    StartCollectionAndSending();

    while (true)
    {
        Thread.Sleep(8000);
        Console.WriteLine(Device.LastCollectTime + "..." + Device.LastSendTime);
    }
}

private static void StartCollectionAndSending()
{
    _timerCollectData = new (ReadInterval);
    _timerCollectData.Elapsed += new (CollectData);
    _timerCollectData.Enabled = true;

    _timerSendData = new (SendInterval);
    _timerSendData.Elapsed += new (SendData);
    _timerSendData.Enabled = true;

    Console.CancelKeyPress += delegate
    {
        // Handle cancel key press
    }
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParticleSensorType</td>
<td>ParticleSensorType</td>
</tr>
</tbody>
</table>
```csharp
public static string StreamId { get; set; }

public static string DeviceClientId { get; set; }

public static string DeviceClientSecret { get; set; }

public static void Main()
{
    SetupConfiguration();
    SendTypeAndContainerAsync().GetAwaiter().GetResult();
    StartCollectionAndSending();

    while (true)
    {
        Thread.Sleep(8000);
        Console.WriteLine(Device.LastCollectTime + "..." + Device.LastSendTime);
    }

    private static void StartCollectionAndSending()
    {
        _timerCollectData = new (ReadInterval);
        _timerCollectData.Elapsed += new (CollectData);
        _timerCollectData.Enabled = true;
    }
```
public static void SetupConfiguration()
{
    IConfigurationBuilder builder = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("appsettings.json");
    _config = builder.Build();

    // **** Client-constants ****
    TenantId = _config["TenantId"]; 
    NamespaceId = _config["NamespaceId"]; 
    Resource = _config["Resource"]; 
    ClientId = _config["ClientId"]; 
    ClientSecret = _config["ClientSecret"]; 
    ConnectionName = _config["ConnectionName"]; 
    DeviceClientId = _config["DeviceClientId"]; 
    ReadInterval = Convert.ToDouble(_config["ReadInterval"]); 
    SendInterval = Convert.ToDouble(_config["SendInterval"]); 
    GetDeviceClientSecret();

    _omfDevice = new Device(Resource, TenantId, NamespaceId, DeviceClientId, DeviceClientSecret);
}

/// <summary>
/// Gets the device secret and saves it securely.
/// </summary>
public static string DeviceClientID { get; set; }

public static string DeviceClientSecret { get; set; }

public static void Main()
{
    SetupConfiguration();
    SendTypeAndContainerAsync().GetAwaiter().GetResult();
    StartCollectionAndSending();

    while (true)
    {
        Thread.Sleep(8000);
        Console.WriteLine(Device.LastCollectTime + "..." + Device.LastSendTime);
    }
}

private static void StartCollectionAndSending()
{
    _timerCollectData = new (ReadInterval);
    _timerCollectData.Elapsed += new (CollectData);
    _timerCollectData.Enabled = true;

    _timerSendData = new (SendInterval);
    _timerSendData.Elapsed += new (SendData);
}
```csharp
public static string StreamId { get; set; }

public static string DeviceClientId { get; set; }

public static string DeviceClientSecret { get; set; }

public static void Main()
{
    SetupConfiguration();
    SendTypeandContainerAsync().GetAwaiter().GetResult();
    StartCollectionAndSending();

    while (true)
    {
        Thread.Sleep(8000);
        Console.WriteLine(Device.LastCollectTime + " "+ Device.LastSendTime);
    }
}

private static void StartCollectionAndSending()
{
    _timerCollectData = new (ReadInterval);
    _timerCollectData.Elapsed += new (CollectData);
    _timerCollectData.Enabled = true;
}
```
public static void Main()
{
    SetupConfiguration();
    SendTypeAndContainerAsync().GetAwaiter().GetResult();
    StartCollectionAndSending();
    while (true)
    {
        Thread.Sleep(8000);
        Console.WriteLine(Device.LastCollectTime + "..." + Device.LastSendTime);
    }
}

private static void StartCollectionAndSending()
{
    _timerCollectData = new (ReadInterval);
    _timerCollectData.Elapsed += new (CollectData);
    _timerCollectData.Enabled = true;

    _timerSendData = new (SendInterval);
    _timerSendData.Elapsed += new (SendData);
    _timerSendData.Enabled = true;
Egress
What are streams?

• Fundamental block of SDS
• An instantiated object of an SDS type
• Can have UOMs, interpolation, extrapolation defined on each property at the stream or type
• Metadata and tags
• Can be shared to a community
• Summary data calls
• Stream views
• Remove or rename properties
• Change UOM
How to best read data out?

What are assets?

- Fundamental block in ADH
- Can be based on an asset type
- Collection of streams
- Can read just some properties (streams)
- Metadata
- Asset status
How to best read data out?

What are data views?

• Based on a query against assets or streams
  • Can query other namespaces and communities
• Pre-defined (time range is variable)
• Re-name columns
• Summary values
• Different output forms
  • JSON
  • CSV
  • Table
  • Parquet
How to best read data out?

Things to consider

• Repeated pre-defined query vs ad-hoc investigation
• Interpolated vs stored (note all support both...)
• Tabular data?
• Default value
• Separating query into smaller ranges on different threads
• Paging
• UOMs
• Summary Calculations
Wind farm operator

• Operating fleet of wind turbines
• Need one year’s worth of 5 second data for 10 assets with ~50 streams each
• Need to make the data available for machine learning project
  => would like the data in a centralized data lakehouse in Microsoft Fabric to perform ML analytics at scale

• Use Data Views in AVEVA Data Hub to transform data from assets to a more structured tabular form, and determine the API and query parameters to egress in Apache Parquet format
• Use Data Pipelines to ingest this data as efficiently as possible into Fabric using pagination and parallelization

• AVEVA collaborated with Microsoft in designing such a pipeline
Latest Service Updates

PI to Data Hub 2.2 is released

Jul 17, 2023, 2:33:09 PM

AVEVA is pleased to announce a new release for PI to AVEVA Data Hub. With this release, PI to Data Hub now enables reporting for AVEVA PI Data Infrastructure – aggregate tag licensing model. This licensing option allows customers to purchase PI Tags in aggregate (with a minimum committed number of tags) across any number of deployed PI Servers rather than having to specify a fixed number of PI Tags for a specific PI Data Archive server.

New! AVEVA PI Data Infrastructure – aggregate tag support

Quick Links

- View API documentation
- Explore working code samples provided in multiple programming languages
- View service blog
- Manage Users And User Access For Your Organization
- Manage clients and secrets for securely accessing your data
- Experiment with the REST API console

Yesterday's Resource Usage

Oct 9, 2023

<table>
<thead>
<tr>
<th>Streams Stored</th>
<th>Streams Accessed</th>
<th>Shared Streams Accessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>70,513</td>
<td>1,478</td>
<td>0</td>
</tr>
</tbody>
</table>

System Health

- Health Status: Ok

PI to Data Hub Agents

- Total Agents: 14
  - Good: 7
  - Warning: 2
  - Bad: 5
  - Stopped: 0
Link to Tutorial Blog

• Data Pipelines Tutorial: Ingest files into a Lakehouse from a REST API with pagination ft. AVEVA Data Hub | Microsoft Fabric Blog | Microsoft Fabric

https://bit.ly/3PUy2Vq
Additional Useful Links

• AVEVA and OSIsoft Github Samples
  • AWC Demo Code – https://github.com/AVEVA/demos-aveva_world_2023

• OMF Documentation

• AVEVA Data Hub Documentation

© 2023 AVEVA Group Limited and its subsidiaries. All rights reserved.
Derek Endres
Sr. Manager Product Readiness Guild
- AVEVA
- Derek.Endres@aveva.com

Yuki Kimura
Principal Software Developer
- AVEVA
- Yuki.Kimura@aveva.com
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