



# NIAGARA SUMMIT 2026

SEAMLESS CONNECTIVITY,  
POWERFUL INTELLIGENCE

TRIDIUM 

*This document is a non-binding, confidential document that contains valuable proprietary and confidential information of Tridium and must not be disclosed to any third party without our written agreement. It does not create any binding obligation for us to develop or sell any product, service, or offering. Content provided here may not be altered or modified and must remain in the format originally presented by Tridium. Any descriptions of future product direction, intended updates, or new or improved features or functions are for informational purposes only and are not binding commitments by Tridium. The sale, development, release, and timing of any such products, updates, features, or functions remain in Tridium's sole discretion.*



**Build your first  
NCS cloud app  
in minutes**

**NIAGARA  
SUMMIT  
2026  
DEVELOPER**

Vidya Shivamurthy &  
Radu Siminiceanu

# Outline

- Niagara Cloud Suite public APIs
- Autogenerated client-code
- Integrating NDS with BI tools

# Typical workflow

- Create / scaffold the app
- Integrate the Niagara Cloud APIs
- Write your client code
  - Retrieve data from the cloud
  - Do something with the data

# Typical workflow

- Create / scaffold the app → application framework ✓
- Integrate the Niagara Cloud APIs
- Write your client code
  - Retrieve data from the cloud
  - Do something with the data

# Typical workflow

- Create / scaffold the app → application framework ✓
- Integrate the Niagara Cloud APIs → dependency / library ✓
- Write your client code
  - Retrieve data from the cloud
  - Do something with the data

# Typical workflow

- Create / scaffold the app → application framework ✓
- Integrate the Niagara Cloud APIs → dependency / library ✓
- Write your client code
  - Retrieve data from the cloud → REST HTTP calls ✓
  - Do something with the data

# Typical workflow

- Create / scaffold the app → application framework ✓
- Integrate the Niagara Cloud APIs → dependency / library ✓
- Write your client code
  - Retrieve data from the cloud → REST HTTP calls ✓
  - Do something with the data
    - UI : charts, diagrams, widgets, etc.
    - "headless": analytics, reports, etc.

# Typical workflow

- Create / scaffold the app → application framework ✓
- Integrate the Niagara Cloud APIs → dependency / library ✓
- Write your client code
  - Retrieve data from the cloud → REST HTTP calls ✓
  - Do something with the data
    - UI : charts, diagrams, widgets, etc.
    - "headless": analytics, reports, etc.

# Niagara Cloud Suite APIs

<https://www.niagara-cloud.com>

- [Alarms API](#)
  - /api/v1/alarms
- [History/Telemetry API](#)
  - /api/v1/egress
- [Control API](#)
  - /api/v1/control
- [Entity Model API](#)
  - /api/v1/entitymodel

# API Specs

<https://www.niagara-cloud.com/api/v1/alarms/swagger-ui/index.html>

The screenshot shows the Swagger UI for the Alarms Service API. At the top, the title is "Alarms Service API" with version "1.0" and "OAS 3.1" tags. Below the title is the URL "/api/v1/alarms/v3/api-docs". A "Servers" section contains a dropdown menu with the selected server "https://www.niagara-cloud.com/api/v1/alarms - Generated server url" and an "Authorize" button. The main content area lists two controllers: "alarm-history-controller" and "alarm-state-controller". Under "alarm-history-controller", there is a POST endpoint: "/customers/{customerId}/alarms:search" with the description "Query alarms by customerId." Under "alarm-state-controller", there are two GET endpoints: "/customers/{customerId}/devices/{deviceUuid}/alarms/state" (description: "Query alarm states by customerId, deviceUuid and, optionally, sourceId") and "/customers/{customerId}/devices/{deviceUuid}/alarms/count" (description: "Query alarm counts by customerId, deviceUuid and, optionally, sourceId"). At the bottom, a "Schemas" section is partially visible, showing "AlarmsRequest" with an "Expand all" link and the type "object".

# API Specs

<https://www.niagara-cloud.com/api/v1/alarms/swagger-ui/index.html>

The screenshot displays the Swagger UI for the Alarms Service API. At the top, it shows the API title "Alarms Service API" with version "1.0" and "OAS 3.1" specification. Below this, there is a "Servers" section with a dropdown menu showing the URL "https://www.niagara-cloud.com/api/v1/alarms - Generated server url" and an "Authorize" button. The main content area lists several API endpoints under two controllers: "alarm-history-controller" and "alarm-state-controller". The "alarm-state-controller" section includes a "GET" endpoint for "/customers/{customerId}/devices/{deviceUuid}/alarms/state" and another "GET" endpoint for "/customers/{customerId}/devices/{deviceUuid}/alarms/count". At the bottom, the "Schemas" section is highlighted with a red box, showing the "AlarmsRequest" schema with a red arrow pointing to it from the text "Data model".

Alarms Service API <sup>1.0</sup> <sup>OAS 3.1</sup>  
</api/v1/alarms/v3/api-docs>

Servers  
https://www.niagara-cloud.com/api/v1/alarms - Generated server url Authorize

**alarm-history-controller** ^

**POST** /customers/{customerId}/alarms:search Query alarms by customerId. lock dropdown

**alarm-state-controller** ^

**GET** /customers/{customerId}/devices/{deviceUuid}/alarms/state Query alarm states by customerId, deviceUuid and, optionally, sourceId lock dropdown

**GET** /customers/{customerId}/devices/{deviceUuid}/alarms/count Query alarm counts by customerId, deviceUuid and, optionally, sourceId lock dropdown

**Schemas** ^

**AlarmsRequest** dropdown Expand all **object**

**Data model** →

# API Specs

<https://www.niagara-cloud.com/api/v1/alarms/swagger-ui/index.html>

The screenshot displays the Swagger UI for the Alarms Service API. At the top, it shows the API title "Alarms Service API" with version "1.0" and specification "OAS 3.1". Below this, there is a "Servers" section with a dropdown menu showing the URL "https://www.niagara-cloud.com/api/v1/alarms - Generated server url" and an "Authorize" button. The main content area lists API methods under two controllers: "alarm-history-controller" and "alarm-state-controller". The "alarm-state-controller" section is highlighted with a red box and a red arrow pointing to it from the text "API methods". The methods listed are:

- POST** `/customers/{customerId}/alarms/search` Query alarms by customerId.
- GET** `/customers/{customerId}/devices/{deviceUuid}/alarms/state` Query alarm states by customerId, deviceUuid and, optionally, sourceId.
- GET** `/customers/{customerId}/devices/{deviceUuid}/alarms/count` Query alarm counts by customerId, deviceUuid and, optionally, sourceId.

Below the methods, there is a "Schemas" section with a dropdown menu showing "AlarmsRequest" and options to "Expand all" and "object".

# API Specs

<https://www.niagara-cloud.com/api/v1/alarms/swagger-ui/index.html>

**Alarms Service API** 1.0 OAS 3.1

</api/v1/alarms/v3/api-docs> ← specification

Servers

<https://www.niagara-cloud.com/api/v1/alarms> - Generated server url Authorize

**alarm-history-controller** ^

**POST** `/customers/{customerId}/alarms:search` Query alarms by customerId. lock v

**alarm-state-controller** ^

**GET** `/customers/{customerId}/devices/{deviceUuid}/alarms/state` Query alarm states by customerId, deviceUuid and, optionally, sourceId lock v

**GET** `/customers/{customerId}/devices/{deviceUuid}/alarms/count` Query alarm counts by customerId, deviceUuid and, optionally, sourceId lock v

Schemas ^

**AlarmsRequest** > Expand all `object`

# API client code

# API client code

Auto-generated

# API client code

AI generated

# API client code

OpenAI

# API client code

OpenAPI

# OpenAPI

- Swagger spinoff (Wordnik / SmartBear Software)
- V3 → Specification language for HTTP APIs (open standard)
  - Consistent structure and syntax ✓
- Toolset for complete API lifecycle
  - Requirements, design, configuration, deployment, testing ✓

# Why OpenAPI?

- ✓ Correct
- ✓ Complete
- ✓ Consistent

# The OpenAPI generator

<https://github.com/OpenAPITools/openapi-generator>

<b>API clients</b>	ActionScript, Ada, Apex, Bash, C, C# (.net 2.0, 3.5 or later, .NET Standard 1.3 - 2.1, .NET Core 3.1, .NET 5.0. Libraries: RestSharp, GenericHost, HttpClient), C++ (Arduino, cpp-restsdk, Qt5, Tizen, Unreal Engine 4), Clojure, Crystal, Dart, Elixir, Elm, Eiffel, Erlang, Go, Groovy, Haskell (http-client, Servant), Java (Apache HttpClient 4.x, Apache HttpClient 5.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, REStEasy, Vertx, Google API Client Library for Java, Rest-assured, Spring 5 Web Client, Spring 6 RestClient, MicroProfile Rest Client, Helidon), JetBrains HTTP Client, Julia, k6, Kotlin, Lua, N4JS, Nim, Node.js/JavaScript (ES5, ES6, AngularJS with Google Closure Compiler annotations, Flow types, Apollo GraphQL DataStore), Objective-C, OCaml, Perl, PHP, PowerShell, Python, R, Ruby, Rust (hyper, reqwest, rust-server), Scala (akka, http4s, scalaz, sttp, swagger-async-httpclient, pekko), Swift (2.x, 3.x, 4.x, 5.x, 6.x), Typescript (AngularJS, Angular (9.x - 19.x), Aurelia, Axios, Fetch, Inversify, jQuery, Nestjs, Node, redux-query, Rxjs), Xojo, Zapier
Server stubs	Ada, C# (ASP.NET Core, Azure Functions), C++ (Httpplib, Oat++, Pistache, Restbed, Qt5 QHTTPEngine), Erlang, F# (Giraffe), Go (net/http, Gin, Echo), Haskell (Servant, Yesod), Java (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, Jersey, RestEasy, Play Framework, PKMST, Vert.x, Apache Camel, Helidon), Julia, Kotlin (Spring Boot, Ktor, Vert.x), PHP (Flight, Laravel, Lumen, Mezzio (fka Zend Expressive), Slim, Silex, Symfony), Python (FastAPI, Flask), NodeJS, Ruby (Sinatra, Rails5), Rust (rust-server), Scala (Akka, Finch, Lagom, Play, Cask, Scalatra)

# The OpenAPI generator

<https://github.com/OpenAPITools/openapi-generator>

API clients	40 languages
Server stubs	16 languages

# Any language...

AI

# C#

# Standalone library

<https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/7.20.0/openapi-generator-cli-7.20.0.jar>

```
java -jar openapi-generator-cli.jar generate \  
-g java \  
-i alarms.json \  
-o ncs-alarms-client \  
--group-id com.tridium \  
--artifact-id ncs-alarms-client-java \  
--artifact-version 0.1
```

# Standalone library

<https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/7.20.0/openapi-generator-cli-7.20.0.jar>

```
java -jar openapi-generator-cli.jar generate \  
-g java \  
-i alarms.json \  
-o ncs-alarms-client \  
--group-id com.tridium \  
--artifact-id ncs-alarms-client-java \  
--artifact-version 0.1
```

# Standalone library

<https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/7.20.0/openapi-generator-cli-7.20.0.jar>

```
java -jar openapi-generator-cli.jar generate \  
-g java \  
-i alarms.json \  
-o ncs-alarms-client \  
--group-id com.tridium \  
--artifact-id ncs-alarms-client-java \  
--artifact-version 0.1
```

# Standalone library

<https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/7.20.0/openapi-generator-cli-7.20.0.jar>

```
java -jar openapi-generator-cli.jar generate \  
-g java \  
-i alarms.json \  
-o ncs-alarms-client \  
--group-id com.tridium \  
--artifact-id ncs-alarms-client-java \  
--artifact-version 0.1
```

# Standalone library

<https://repo1.maven.org/maven2/org/openapitools/openapi-generator-cli/7.20.0/openapi-generator-cli-7.20.0.jar>

```
java -jar openapi-generator-cli.jar generate \  
-g javascript \  
-i alarms.json \  
-o ncs-alarms-client
```

# As local dependency

Example (Maven project)

```
<dependency>
```

```
<groupId>com.tridium</groupId>
```

```
<artifactId>ncs-alarms-client-java</artifactId>
```

```
<version>0.1</version>
```

```
<scope>system</scope>
```

```
<systemPath>
```

```
  ${project.basedir}/lib/ncs-alarms-client-java-0.1.jar
```

```
</systemPath>
```

```
</dependency>
```

# Maven plugin, with local spec file

```
<plugin>
```

```
<groupId>org.openapitools</groupId>
```

```
<artifactId>openapi-generator-maven-plugin</artifactId>
```

```
<version>7.20.0</version>
```

```
<executions><execution>
```

```
<goals><goal>generate</goal></goals>
```

```
<configuration>
```

```
<generatorName>java</generatorName>
```

```
<inputSpec>${project.basedir}/resources/alarms.json</inputSpec>
```

```
<configOptions>
```

```
<sourceFolder>src/gen/java/main</sourceFolder>
```

```
</configOptions>
```

```
</configuration>
```

```
</execution></executions>
```

```
</plugin>
```

# Maven plugin, with URL spec

```
<plugin>
```

```
<groupId>org.openapitools</groupId>
```

```
<artifactId>openapi-generator-maven-plugin</artifactId>
```

```
<version>7.20.0</version>
```

```
<executions><execution>
```

```
<goals><goal>generate</goal></goals>
```

```
<configuration>
```

```
<generatorName>java</generatorName>
```

```
<inputSpec>https://www.niagara-cloud.com/api/v1/alarms/v3/api-docs</inputSpec>
```

```
<configOptions>
```

```
<sourceFolder>src/gen/java/main</sourceFolder>
```

```
</configOptions>
```

```
</configuration>
```

```
</execution></executions>
```

```
</plugin>
```

# Example #1

- Language: Java
- Framework: Spring
- Project: Maven
- IDE: IntelliJ

# Example #1

- Time: 15 minutes
- Lines of code:
  - manually written: ~100 loc
  - reused (auth templates): ~100 loc
  - generated: ~7,000 loc
  - spec file: ~1,000 loc
- Savings: ~98.78% generated code

# Auth

## Bearer token (JWT) example

```
AlarmStateControllerApi api = new AlarmStateControllerApi();
ApiClient client = api.getApiClient();
client.setBasePath(alarmsUrl);
HttpBearerAuth auth =
    (HttpBearerAuth) client.getAuthentication("bearerAuth");
auth.setBearerToken(tokenSupplier);
...
List<AlarmCount> alarmCounts =
    api.getAlarmCounts(customer, deviceUuid, null);
```

# Notes

- Additional dependencies may be needed
- WebApp auth vs User auth vs Device auth
- Paginated API responses

# To sum up



Manually written code



OpenAPI autogenerated client.

# ... alternatively



Manually written code



AI generated code.

# The Art of Seamless Integration: A Journey with NDS API

**NIAGARA  
SUMMIT  
2026**

# Buildings : Data-Mine

- 1 Alarms
- 2 History
- 3 Live Data
- 4 Schedule
- 5 Model

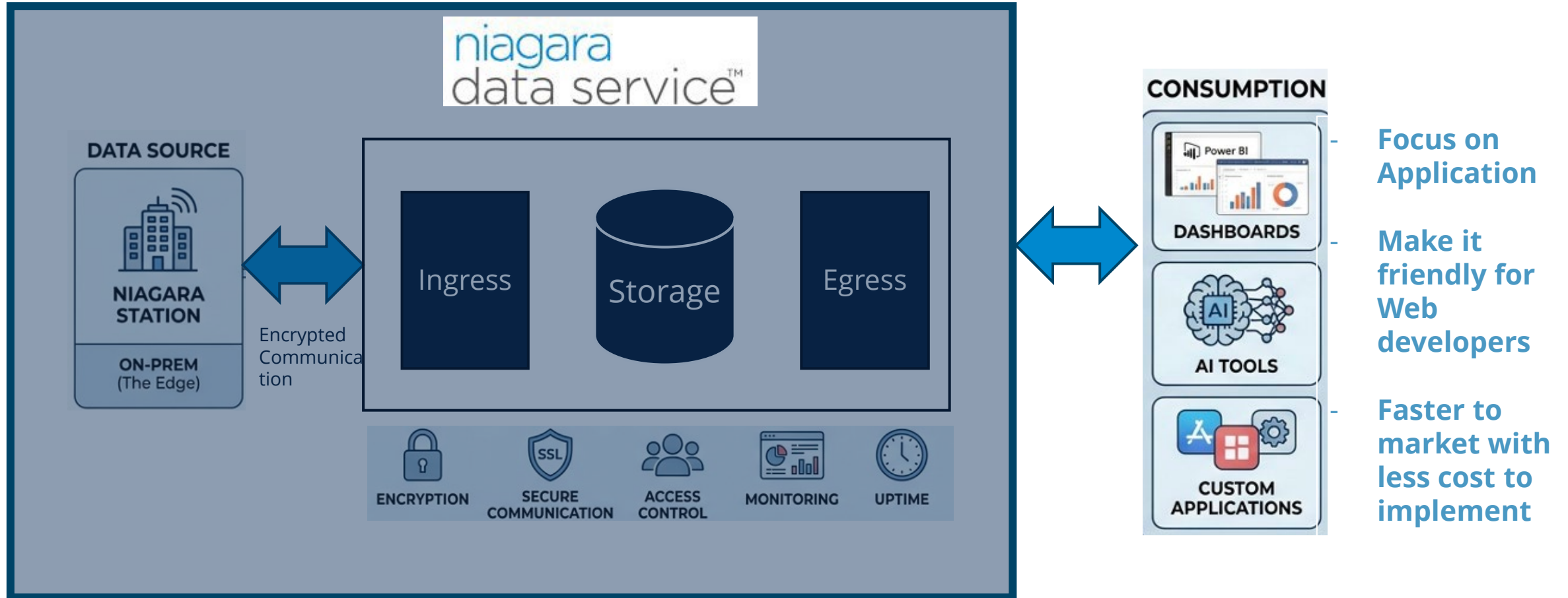


# Unlocking Building Data with NDS



- 1.Data collection
- 2.Connectivity to other controllers

# Unlocking Building Data with NDS



# Analytics & AI tools



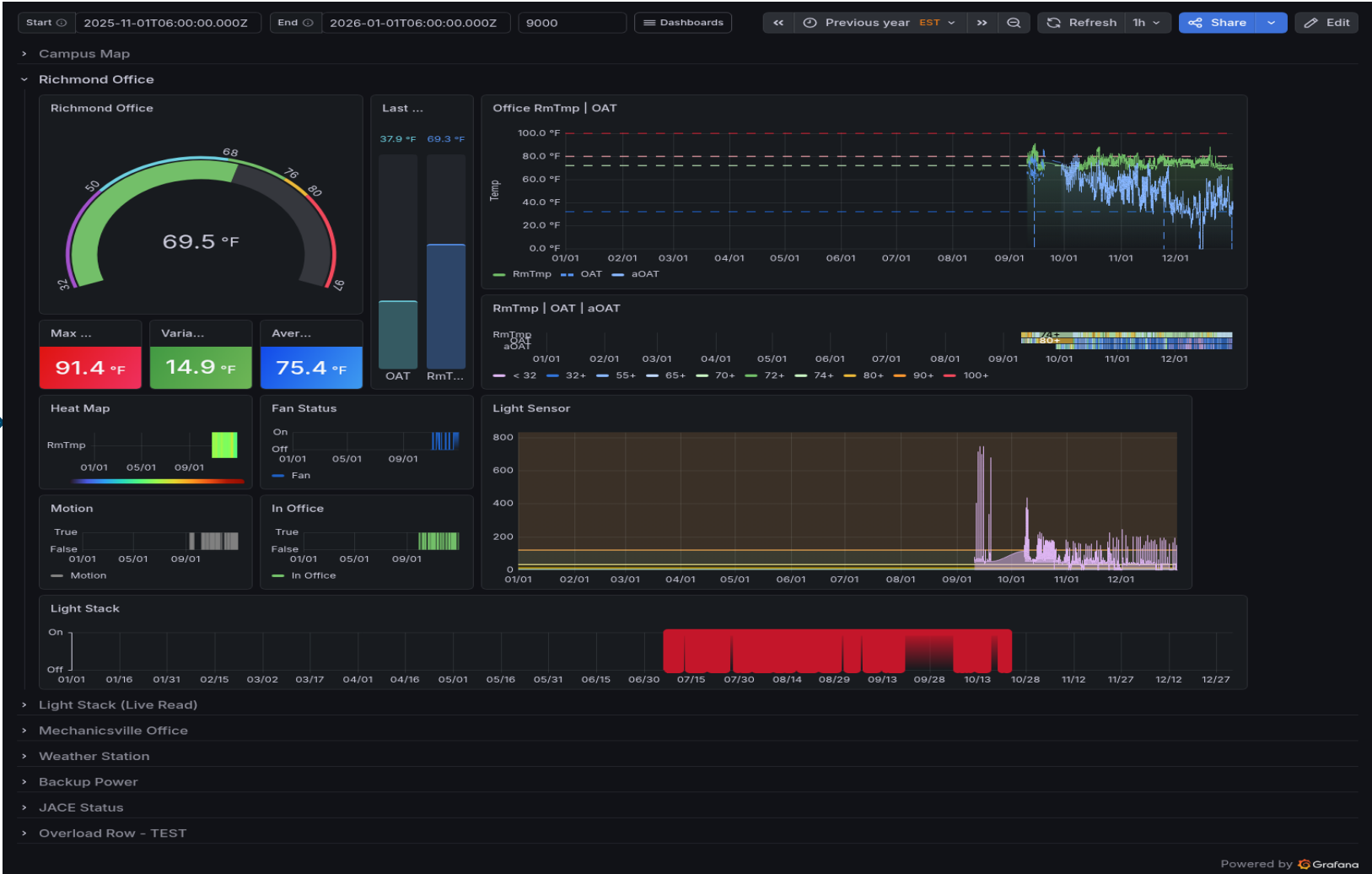
- Partner tools (Forge, Conexao, Autodesk Tandem, Elipsa...)

# Prerequisites

- Niagara Stations with NDS subscription registered to NCS and is sending data.
  - Copy the [System GUID](#) and [Cloud ID](#) of the points
- Service account created in NCS Portal

# Pattern: Data Visualization

niagara  
data service™



# Pattern: Business Intelligence

niagara  
data service™



# Open Integration


- Integration with your preferred applications
- Extends Niagara's "universal adapter"



Thank you

**NIAGARA  
SUMMIT  
2026  
DEVELOPER**

Vidya Shivamurthy &  
Radu Siminiceanu



# UP NEXT: Q&A



# NIAGARA SUMMIT 2026 DEVELOPER

Q&A