

TrendMiner Success Series - Admin Track

September 9th, 2026

Connector Modernization TrendMiner Providers & ODBC Improvements

Welcome to the webinar. We will start momentarily.





Listen only mode



**Use the question area
in the side panel**

Q&A at the end



**A recording and slides will be
made available after the webinar**

In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



Why connectivity requires Windows

- Historian vendors ship Windows based SDKs (COM / .NET)
- ODBC drivers are only released and supported for Windows
 - JDBC drivers are not always available cross-platform
- Firewall security requirements
 - AWS security
 - Appliance is on different access level

Historians that require the Windows

- AspenTech IP21 – Windows ODBC interface
- Exaquantum – Windows .NET SDK
- Honeywell PHD – Windows .NET SDK
- Yokogawa Wonderware – Windows SDK/ODBC
- Aveva Wonderware – Windows .NET SDK/ODBC
- Aveva PI – Windows .NET SDK
- GE Proficy – Windows .NET SDK

Note: Refer documentation for full up-to-date list

In today's session



Plant Integrations

The foundation



WebHost → AppHost

Plant Integration management improvements



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up

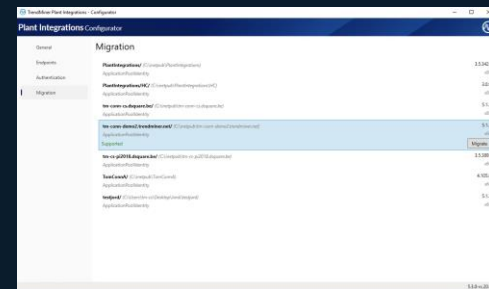
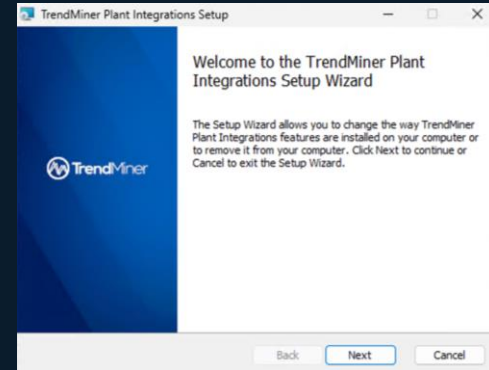


Installation and upgrades - easier and friendlier than IIS

Port 8001 should be available during installation

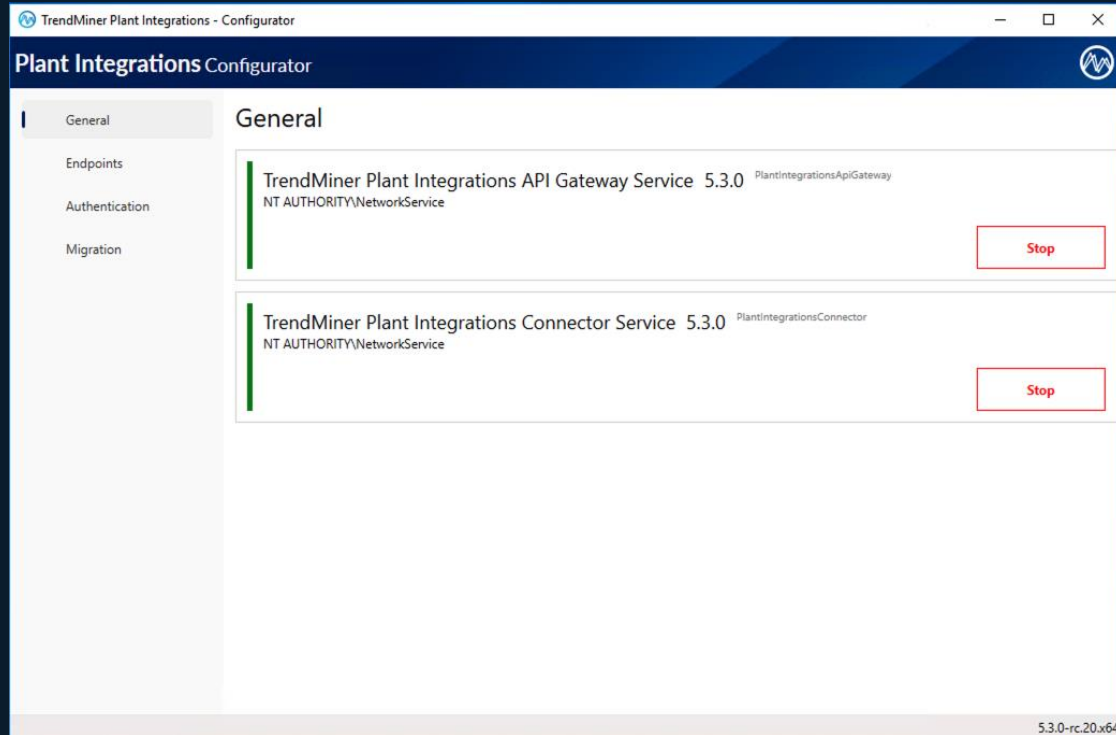
Local firewall rules are automatically updated

Configuration: Authentication, endpoints, certificates, services management



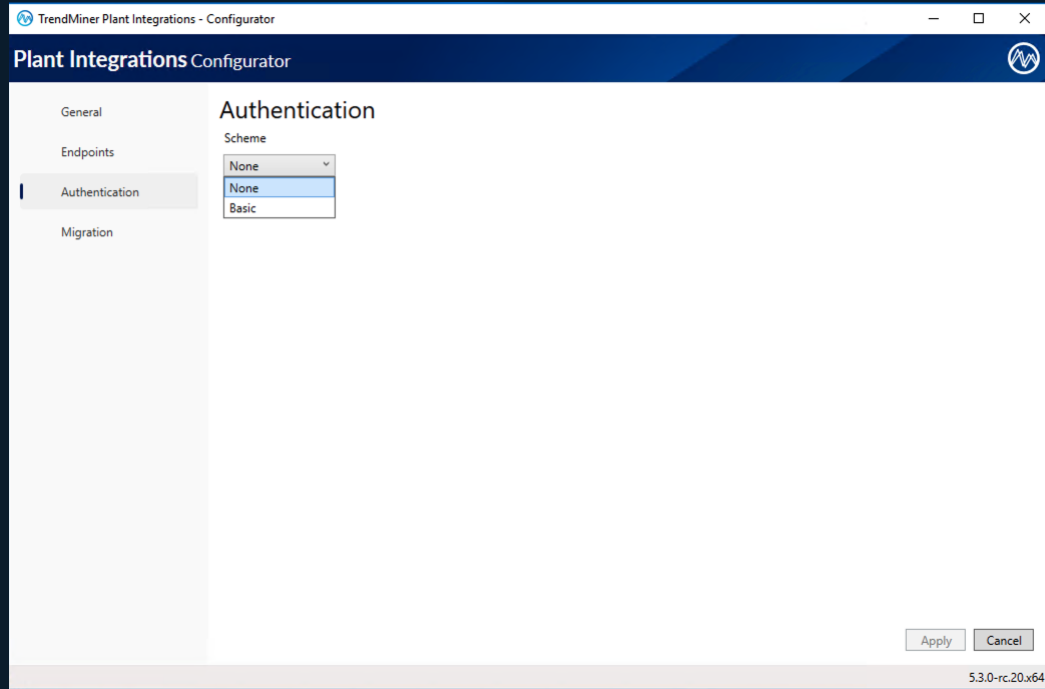
Management - easier and friendlier than IIS

General services overview



Management - easier and friendlier than IIS

Authentication Configuration



Management - easier and friendlier than IIS

Endpoints

TrendMiner Plant Integrations - Configurator

Plant Integrations Configurator

General

Endpoints

Authentication

Migration

Endpoints

Bindings

+

http://localhost:8080/

http://localhost:8001/

Type Host Port Path Certificate

https localhost 8001 / IIS Express Development Certificate

Update

Apply Cancel

5.3.0-rc.20.x64

Important Notes

Unsupported

- Currently one Plant Integration service
- Legacy ODBC, SQLite and OleDb providers do not migrate to AppHost
- Migration is supported from connector v5 only (R2025.R2 release)

Keep in mind

- After migration: update the port in ConfigHub to point to the new connector
- Fallback is possible back to IIS
- The modern replacement is Generic ODBC / JDBC

In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



Why Legacy File Based configuration is being retired

- Built long ago, its design predates today's connectivity standards
- Config and maintenance model has not kept pace with the platform
- No parity with Generic ODBC/JDBC tooling
- Legacy ODBC has become difficult to configure and maintain

Investigating Legacy Technologies Alternatives

- OleDb
- Sqlite

Control over Configuration

Config-Hub

DASHBOARD

Services

SETTINGS

Email configuration

Home page configuration

DOCUMENTS

Work organizer

SECURITY

Users

Groups

Clients

Access management

Certificates

Identity providers

Add a data source

Step 1 of 6

Close

Data source details

Name

TEST genODBC

Provider

Generic ODBC

Capabilities

Time series

Asset

Context

Connect via

connector-pipeline-con01.trendminer...

Cancel

Next

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

Config-Hub

DASHBOARD

Services

SETTINGS

Email configuration

Home page configuration

DOCUMENTS

Work organizer

SECURITY

Users

Groups

Clients

Access management

Certificates

Identity providers

Add TEST genODBC data source

Step 2 of 6

Close

Connection details

Parallel connections (global default)

2

Driver properties

key:value

key:value

Password

Connection string

Driver={SQL Server};Server=ServerName;D...

Username

Command Timeout

00:00:30

Test connection

Back

Next

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

SUCCESS

Access to Configuration

Config-Lib

DASHBOARD

SERVICES

SETTINGS

Email configuration

Home page config

DOCUMENTS

Work organizer

SECURITY

Users

Groups

Clients

Access management

Certificates

Identity providers

ADD TEST genODBC data source

Step 4 of 6

Time series details

Tag list query

select id, name, type, description, units from source

Index query

select ts, value from source where id=ID and ts>=[STARTDATE] and ts<=[ENDDATE] order by ts asc

Internal plotting

Plot query

select ts, value from source where id=ID and ts>=[STARTDATE] and ts<=[ENDDATE] order by ts asc

Digital states query

select stringValue, intValue from source where id=ID

Type mapping

discrete:DISCRETE
string:STRING

Back

Next

[illegible]

Generic ODBC/JDBC parameter syntax

Parameters now use meaningful names instead of positional `{/}{0}` markers

Note: there're small differences between ODBC and JDBC, refer documentation

Legacy ODBC

Legacy ODBC

- Using placeholders `{0}`, `{1}`, `{2}`, `{3}`
- Using unnamed parameters `?`, `?`, `?`, `?`

Positional binding – SQL injection risk when substituted as raw text

Generic ODBC / JDBC

Generic ODBC / JDBC

- Using named parameters
 - `{ID}` – tag name in ODBC, `id` in JDBC
 - `{NAME}` – tag name
 - `{INTERPOLATION}` – interpolation type name
 - `{STARTDATE}` – formatted datetime string
 - `{ENDDATE}` – formatted datetime string
 - `{NUMBEROFINTERVALS}` – number of
 - `{INDEXRESOLUTION}` – index resolution

Named binding – no SQL injection; query must be written to rely on parameters

Generic ODBC/JDBC as an Alternative

- Ignition stores tag history in underlying database
 - MySql
 - MSSQL
 - Postgres TimescaleDB
- Wonderware
- Siemens PCS7
- AspenTech

Not just time series

- Legacy ODBC: time series data only
- Generic ODBC/JDBC and custom connectivity support all three data types:
 - Time Series – the classic case, tag values over time
 - Context – events, batches, process anomalies
 - Asset – structural and hierarchical asset data

Generic JDBC removes the middle tier

Advantages

- Middle Tier
 - Plant Integration: Historian → Plant Integrations → TrendMiner
 - JDBC: Historian → TrendMiner via JDBC driver
- No Windows machine in the middle — where a JDBC driver is available

Except

- Indirect connection still applies when no native driver exists — Windows connector is required
- AWS SaaS uses a reverse-proxy agent for private VPN tunneling and requires Plant Integration.
- Security reasons

Generic JDBC Drivers

Supplied

- MSSQL
- Postgres
- Snowflake
- Databricks
- MariaDB
- Sqlite

Third-Party

- Copy Jar to appliance

In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



Connector API: build your own integration

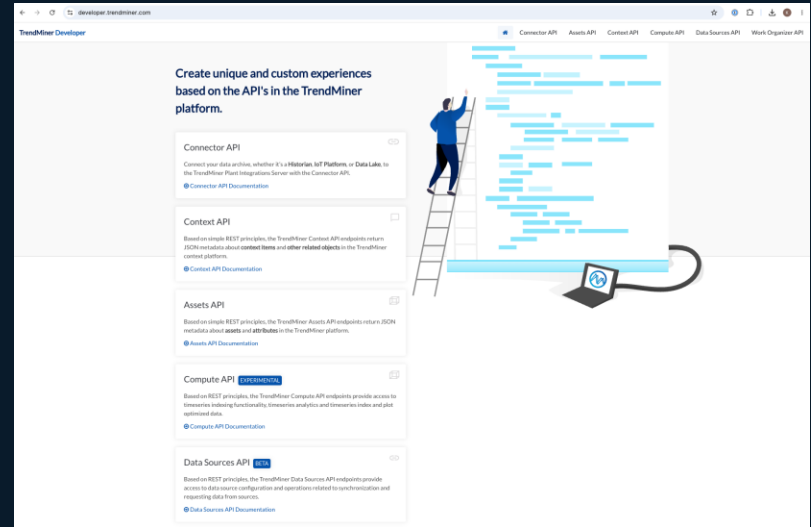
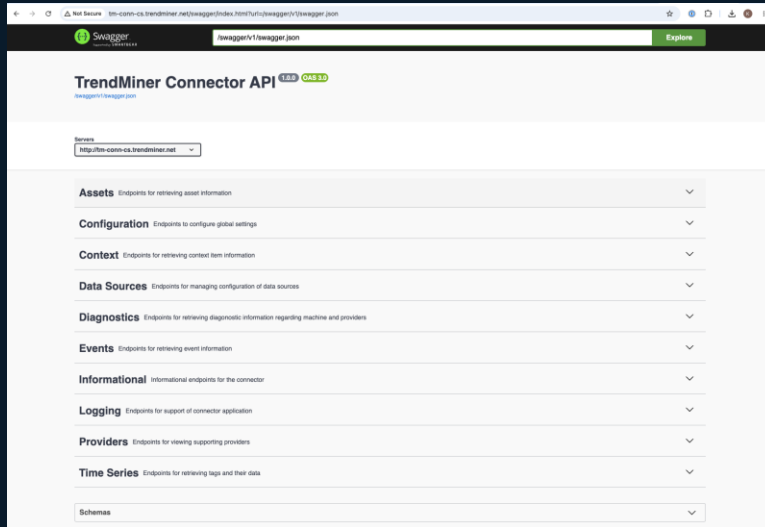
Starting Point:

- Swagger
- [Developer Portal](#)
- [Example Documentation](#)
- Community Posts

Examples:

- Ignition Connectivity
 - Connector API: WebDev module usage by means of Ignition Designer (Jython)
 - Connector API: Custom Java Module option
- Litmus
- Any REST API Service, e.g. Node.js

Demo: Node.js simulated Plant Integrations API



Demo: Node.js simulated Plant Integrations API

The image displays two screenshots of a code editor, likely Visual Studio Code, showing the implementation of a Node.js API for plant integrations.

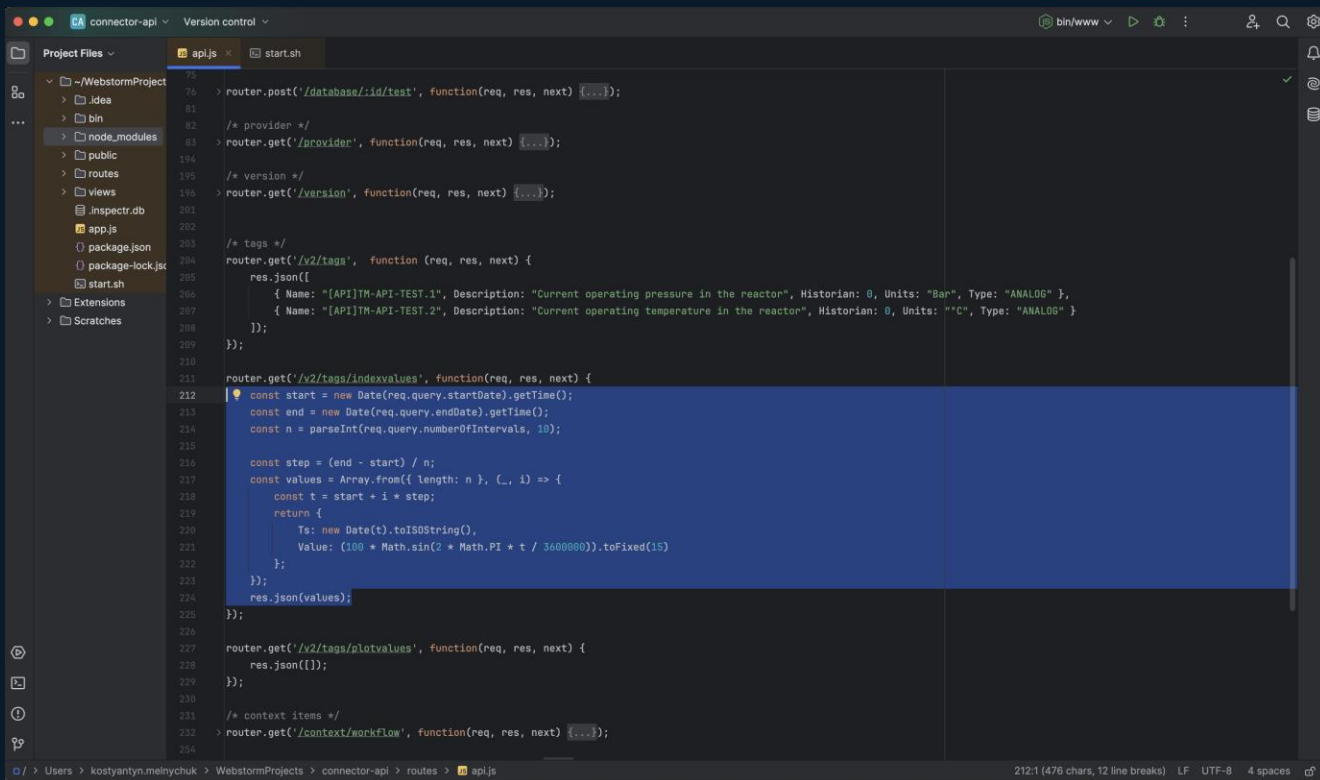
Left Screenshot: Shows the `api.js` file in the `connector-api` project. The code defines a REST API with endpoints for database operations and context management.

```
17 // database +/  
18 router.get('/database', function(req, res, next) {});  
19  
20 router.get('/database/:id', (req, res) => {});  
21  
22 router.put('/database/:id', (req, res) => {});  
23  
24 router.get('/database/:id/test', function(req, res, next) {});  
25  
26 router.post('/database/:id/test', function(req, res, next) {});  
27  
28 // provider +/  
29 router.get('/provider', function(req, res, next) {});  
30  
31 // version +/  
32 router.get('/version', function(req, res, next) {});  
33  
34 // tags +/  
35 router.get('/x2/tags', function(req, res, next) {  
36   res.json({});  
37 });  
38  
39 router.get('/x2/tags/indexvalues', function(req, res, next) {});  
40  
41 router.get('/x2/tags/platvalues', function(req, res, next) {});  
42  
43 // context items +/  
44 router.get('/context/workflow', function(req, res, next) {});  
45  
46 router.get('/context/type', function(req, res, next) {});  
47  
48 router.get('/context/field', function(req, res, next) {});  
49  
50 router.get('/context/changes', function(req, res, next) {});  
51  
52 router.get('/context', async function(req, res, next) {});  
53  
54 router.post('/context', function(req, res, next) {});  
55
```

Right Screenshot: Shows the `database.js` file in the `connector-api` project. The code defines a REST API for database operations, including a `providers` endpoint.

```
41 const databases = { 0: 1, 2: 3 } = { 1: db1, 2: db2 };  
42  
43 const providers = { name: 'TEST_PROVIDER',  
44   {  
45     "name": "TEST_PROVIDER",  
46     "label": "Example API Provider",  
47     "capabilities": ["TIME_SERIES", "ASSET", "CONTEXT"],  
48     "properties": [  
49       {  
50         "name": "HOST",  
51         "type": "SINGLE_LINE",  
52         "label": "Host",  
53         "placeholder": null,  
54         "capabilities": ["TIME_SERIES", "ASSET", "CONTEXT"],  
55         "required": true,  
56         "encrypted": false,  
57         "defaultValue": null  
58       },  
59       {  
60         "name": "USERNAME",  
61         "type": "SINGLE_LINE",  
62         "label": "Username",  
63         "placeholder": null,  
64         "capabilities": [],  
65         "required": false,  
66         "encrypted": false,  
67         "defaultValue": null  
68       },  
69       {  
70         "name": "PASSWORD",  
71         "type": "SINGLE_LINE",  
72         "label": "Password",  
73         "placeholder": null,  
74         "capabilities": [],  
75         "required": false,  
76       }  
77     ]  
78   }  
79 }  
80
```

Demo: Node.js simulated Plant Integrations API

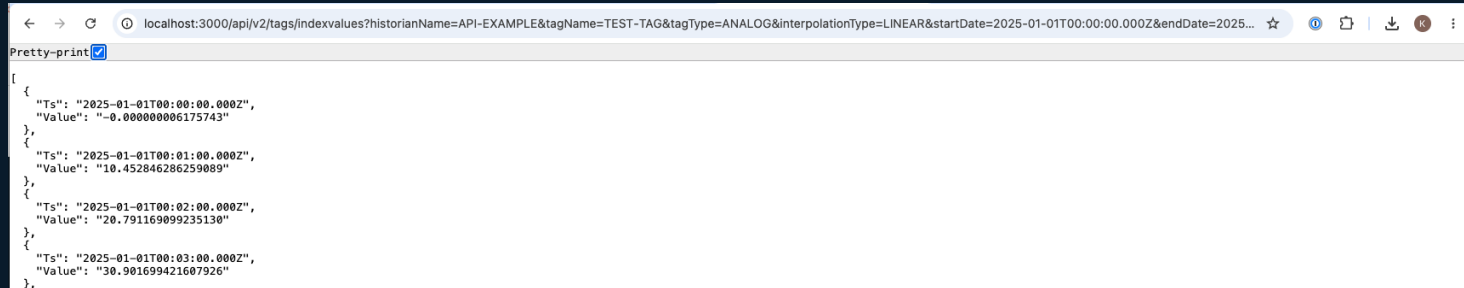
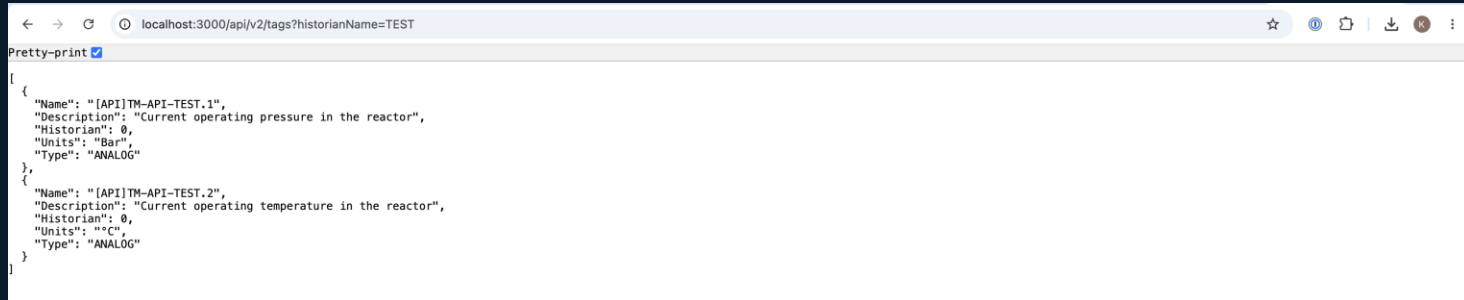
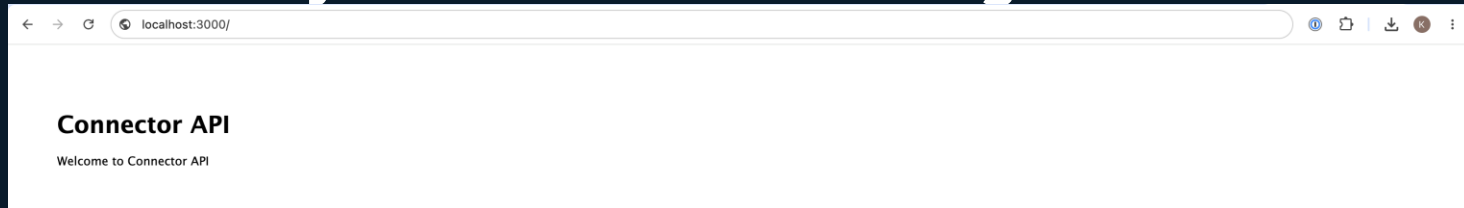


The screenshot shows a VS Code editor window with a project named 'connector-api'. The file explorer on the left shows the project structure, including 'node_modules', 'public', 'routes', 'views', and 'package.json'. The main editor displays the 'routes' file, which contains the following code:

```
73 > router.post('/database/:id/test', function(req, res, next) {...});
74
75 /* provider */
76 > router.get('/provider', function(req, res, next) {...});
77
78 /* version */
79 > router.get('/version', function(req, res, next) {...});
80
81 /* tags */
82 > router.get('/v2/tags', function(req, res, next) {
83   res.json([
84     { Name: "[API]TM-API-TEST.1", Description: "Current operating pressure in the reactor", Historian: 0, Units: "Bar", Type: "ANALOG" },
85     { Name: "[API]TM-API-TEST.2", Description: "Current operating temperature in the reactor", Historian: 0, Units: "C", Type: "ANALOG" }
86   ]);
87 });
88
89 > router.get('/v2/tags/indexvalues', function(req, res, next) {
90   const start = new Date(req.query.startDate).getTime();
91   const end = new Date(req.query.endDate).getTime();
92   const n = parseInt(req.query.numberOfIntervals, 10);
93
94   const step = (end - start) / n;
95   const values = Array.from({ length: n }, (_, i) => {
96     const t = start + i * step;
97     return {
98       Ts: new Date(t).toISOString(),
99       Value: (100 * Math.sin(2 * Math.PI * t / 3600000)).toFixed(15)
100     };
101   });
102   res.json(values);
103 });
104
105 > router.get('/v2/tags/plotvalues', function(req, res, next) {
106   res.json([]);
107 });
108
109 /* context items */
110 > router.get('/context/workflow', function(req, res, next) {...});
111
```

The status bar at the bottom indicates the file is 'api.js' and shows the current cursor position: 212:1 (476 chars, 12 line breaks) LF UTF-8 4 spaces.

Demo: Node.js simulated Plant Integrations API



Demo: Node.js simulated Plant Integrations API

test-api

CONNECTOR DETAILS

Name

test-api

Host

https://tm-test-api.in-spectr.dev

Username

admin

Optional

Password

Optional

Cancel

Save connector

 Edit API-EXAMPLE data source

Step 1 of 3

X Close

Data source details

Name

API-EXAMPLE

Provider

example api trendminer provider

Capabilities

Time series read

Asset read

Context read

☒

☐

☐

Connect via

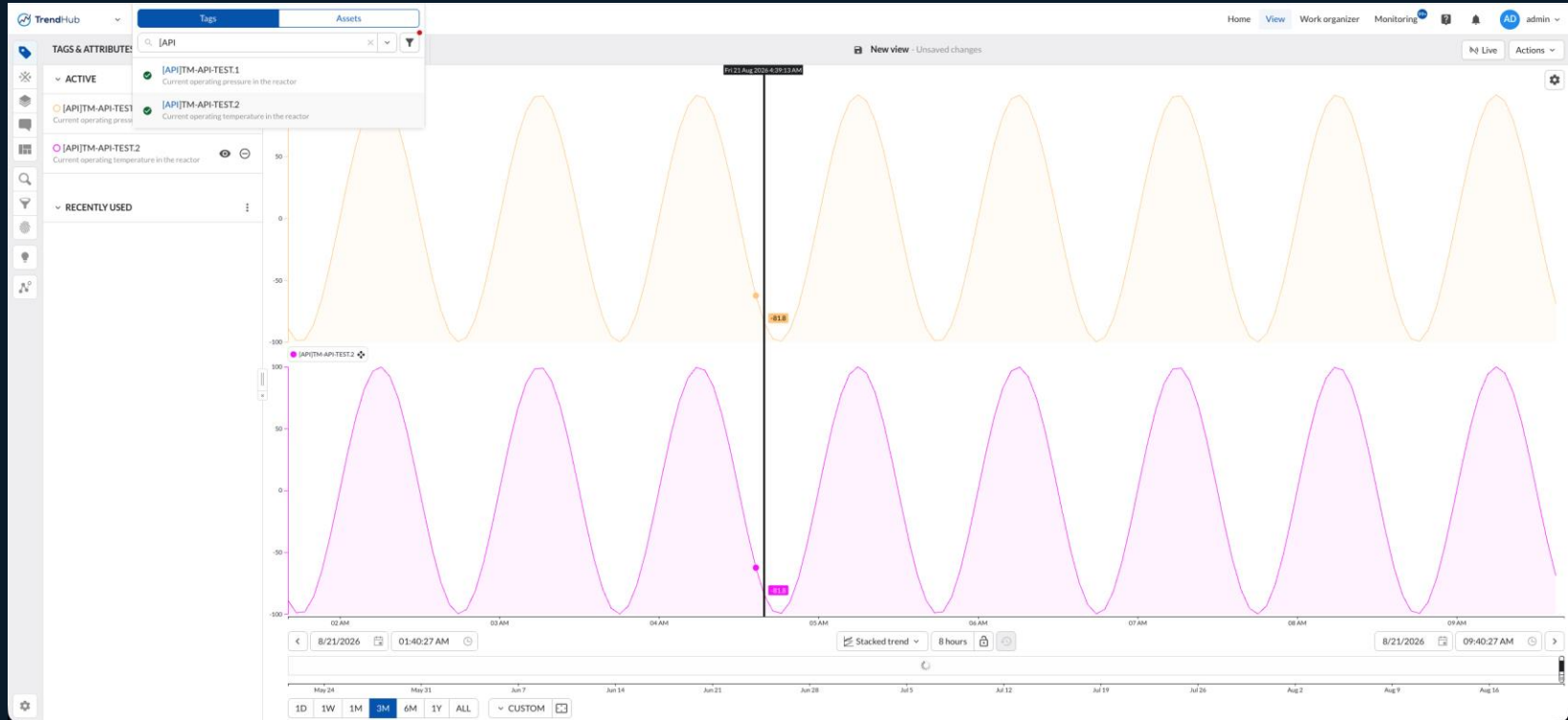
test-api

Cancel

Next

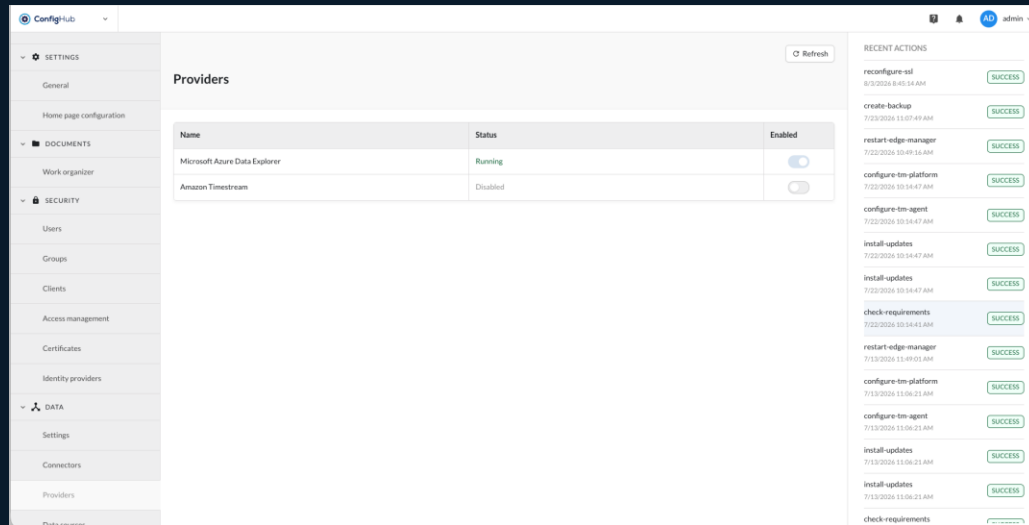
Save data source

Demo: Node.js simulated Plant Integrations API



Additional built-in providers

- Canary (Built-In)
- Amazon Timestream (Runtime)
- Azure Data Explorer (Runtime)



The screenshot displays the ConfigHub web interface. On the left is a sidebar with navigation menus for SETTINGS, DOCUMENTS, SECURITY, and DATA. The main content area is titled 'Providers' and contains a table with the following data:

Name	Status	Enabled
Microsoft Azure Data Explorer	Running	☒
Amazon Timestream	Disabled	☐

To the right of the table is a 'Refresh' button. Further right is a 'RECENT ACTIONS' log listing various system events such as 'reconfigure-od', 'create-backup', 'restart-edge-manager', 'configure-tm-platform', 'configure-tm-agent', 'install-updates', 'check-requirements', and 'restart-edge-manager', each with a timestamp and a 'SUCCESS' status.

Need help?

TrendMiner can make an offer to investigate your connectivity options.

- 2 phases:
 - Investigation and performance report
 - Actual implementation and productization
- One-year updates included
- No cure no pay*
- Contact your CSM

*In case the connection cannot be set up or performance of the data source is insufficient, only part of the investigation effort is charged, not the full price.

In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



Future improvements:

Query Designer

DASHBOARD

Services

SETTINGS

General

Email configuration

MANAGEMENT

License

Backup and restore

Diagnostics

TrendMiner upgrade

SECURITY

Users

Groups

Clients

Access management

SSL

Certificates

Identity providers

Admin access

Data sources / Plant 4 – Production DB / Query Designer

Time series - Query Designer

2 out of 3 required queries filled in

Tag list

Plot data

Raw data

Statistical aggregations

Tag metadata

Tag interpolation (Optional)

Tag list

Query used to create tags. Must return: **id, name, type, description, units**

```
SELECT
tag_id,
tag_name,
description,
unit,
data_type
FROM Historian.tags
WHERE active = 1
ORDER BY tag_name
LIMIT [LIMIT];
```

Copy

View template

Run query

Template variables

Enter test values to run this query. These values are used for testing only and will be substituted at runtime.

[LIMIT]

Optional

5

[STARTDATE]

2026-04-27 00:00:00

[ENDDATE]

2026-04-28 00:00:00

[ID]

[CURSOR]

Future improvements

Future improvement:

Test queries

Config+hub

Work organizer

SECURITY

Users

Groups

Clients

Access management

Certificates

Identity providers

DATA

Settings

Agents

Connectors

Data sources

Webhooks

Diagnostics

AI CONFIGURATION

MCP connection

Data sources > JDBC SQLite1 > Time series read - Query Designer

Time series read - Query Designer

2 out of 2 required queries are filled in

Tag list

Index

Plot (Optional)

Digital states (Optional)

Index

Ingests time series data for a specific tag

Query

Copy View template Run query

1 SELECT ts, value FROM Data WHERE id={ID} AND datetime(substr(ts, 0, 11), 'auto')>= datetime({STARTDATE}) AND datetime(substr(ts, 0, 11), 'auto')<= datetime({ENDDATE}) AND {INDEXRESOLUTION} = 60 order by ts asc

Last run succeeded

10 rows - 21ms

ts	value
1709449200000	33
1709449260000	53
1709449320000	76
1709449380000	94
1709449440000	57
1709449500000	59

Template variables

Enter test values to run this query. These values are used for testing only and will be substituted at runtime.

{ID}

WTS-uniter_1

{STARTDATE}

3/3/2024 08:00:00 AM

{ENDDATE}

3/28/2024 08:00:00 AM

{INTERPOLATION}

Optional

Enter a test value

{INDEXRESOLUTION}

Optional

60

Future improvements

Future improvement:

Diagnose problems

ConfigHub

Groups

Clients

Access management

Certificates

Identity providers

DATA

Settings

Agents

Connectors

Data sources

Webhooks

Diagnostics

AI CONFIGURATION

MCP connection

Agent LLM configuration

DASHHUB

External content

URL embedding

Data sources > JDBC SQLite1 > Time series read - Query Designer

Time series read - Query Designer

2 out of 2 required queries are filled in

• Tag list • Index • **Plot (Optional)** • Digital states (Optional)

Plot

Retrieves time series data for plotting a specific tag

Save query

Query

Copy View template Run query

```
1 SELECT ts, value
2 FROM Data
3 WHERE id={ID} AND datetime(substr(ts, 0, 11), 'auto')>= datetime({STARTDATE}) AND datetime(substr
4 ts, 0, 11), 'auto')<= datetime({ENDDATE})
5 ORDER BY ts asc
```

✓ Last run succeeded

0 rows - 0ms

```
{
  "queryText": [
    {
      "errorCode": "VARIABLE_REFERENCE_MISSING",
      "message": "Variable {INTERVALS} is not used in the query text. This might
produce more data points than supported by the system.",
      "details": {}
    }
  ]
}
```

ts	value
----	-------

Template variables

Enter test values to run this query. These values are used for testing only and will be substituted at runtime.

[ID]

WTS-cables_1

[STARTDATE]

UTC 1/1/2024 01:38:18 PM

Local time: 1/1/2024 2:38:18 PM

[ENDDATE]

UTC 1/28/2024 01:38:18 PM

Local time: 1/28/2024 2:38:18 PM

[INTERPOLATION]

Optional

Enter a test value

[INTERVALS]

Optional

Enter a test value

Future improvement:

Draft concept: Active/Inactive

ConfigHub
work organizer

SECURITY

Users

Groups

Clients

Access management

Certificates

Identity providers

DATA

Data sources > View data source

Overview: AQA JDBC DRAFT

Activate

Details Metrics

Saved as draft - not syncing yet
Configure required queries for each capability below, then activate to start syncing data. Queries can be tested without affecting live data.

Details Edit

Name AQA JDBC DRAFT

Provider Generic JDBC

Identity providers	AQA ADX123	Microsoft Azure Data Expl...	Time series read	8/31/2026 12:03:49 AM	
	AQA AWS IoT SiteWise	AWS IoT SiteWise	Time series read	8/31/2026 12:03:49 AM	
			Asset read	Never	
			Context read	Never	
DATA	AQA AWS Timestream	Amazon Timestream	Time series read	8/31/2026 12:03:50 AM	
Settings	AQA JDBC DRAFT	Generic JDBC	Time series read	Never	
Agents			Asset read	Never	
			Context read	Never	
Connectors	AQA ODBC	Generic ODBC	Time series read	8/31/2026 12:03:01 AM	
		tm-connector-pipeline-con...	Asset read	7/9/2026 2:58:20 PM	
Data sources			Context read	Never	
Webhooks	AQA ODBC DRAFT	Generic ODBC	Time series read	Never	
		tm-connector-pipeline-con...	Asset read	Never	

Future improvement:

Draft concept: Active/Inactive

Time Series

Last synced

Never

Please activate the data source first to enable this operation.

Refresh tag cache

Future improvement:

Standalone runtimes for all providers: Stability and performance



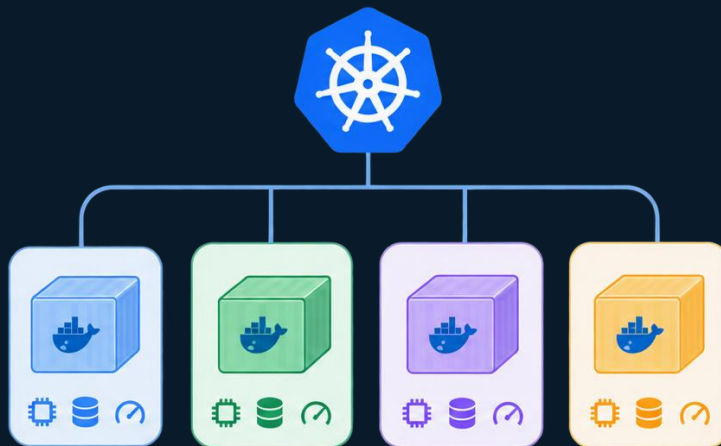
- Any failure does not impact other services



- Manages its own system resources



- Better management



In today's session



Plant Integrations

The foundation



WebHost → AppHost

Migration path



Legacy ODBC vs Generic ODBC

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



documentation.trendminer.com

The screenshot shows the TrendMiner documentation website. The header features the TrendMiner logo on the left and navigation links (User Guide, Admin Docs, Community, Developer Portal) on the right. A 'Version: LATEST' dropdown menu is open, showing a list of versions: 2026.R2.0 (latest), 2026.R1.0, 2025.R4.0, 2025.R3.0, 2025.R2.0, and 2025.R1.0. The main content area has a dark blue background with the text 'How can we help you today?' and a search bar. Below this, five white boxes with circular icons represent different documentation sections: Developer and API Documentation, Release Notes, Installation Guidelines, Upgrade Notes, and Technical Support Resources. At the bottom, a 'FEATURED CONTENT' section displays four items with numbered orange circles: 1. Developer and API Documentation, 2. Release Notes (with a sub-link '2026.R2.0 - Release Notes'), 4. Installation Guidelines (with a sub-link 'TrendMiner architecture'), and 6. Upgrade Notes (with a sub-link 'Prerequisites').

TrendMiner

User Guide Admin Docs Community Developer Portal Version: LATEST

2026.R2.0 (latest)
2026.R1.0
2025.R4.0
2025.R3.0
2025.R2.0
2025.R1.0

How can we help you today?

Search

Developer and API Documentation Release Notes Installation Guidelines

Upgrade Notes Technical Support Resources

FEATURED CONTENT

1 Developer and API Documentation

2 Release Notes
2026.R2.0 - Release Notes

4 Installation Guidelines
TrendMiner architecture

6 Upgrade Notes
Prerequisites

Pages referenced in this webinar

- [Generic ODBC/JDBC documentation](#)
- [TrendMiner connectivity overview](#)
- [Migration WebHost → AppHost](#)
- [Supported time series connectivity which requires query definition](#)
- [How to install third party JDBC driver](#)
- [Connector API documentation](#)

In today's session



Why connectivity requires Windows

The foundation



WebHost → AppHost

Migration path



Legacy ODBC: what changes

New patterns & parameters



Custom connectivity

Connector API



Future improvements

What's coming



Useful Resources

Documentation & links



Q&A

Wrap up



Questions?

