OPC Day Finland 10.12.2025
AI Agents: MCP & OPC UA for Claude

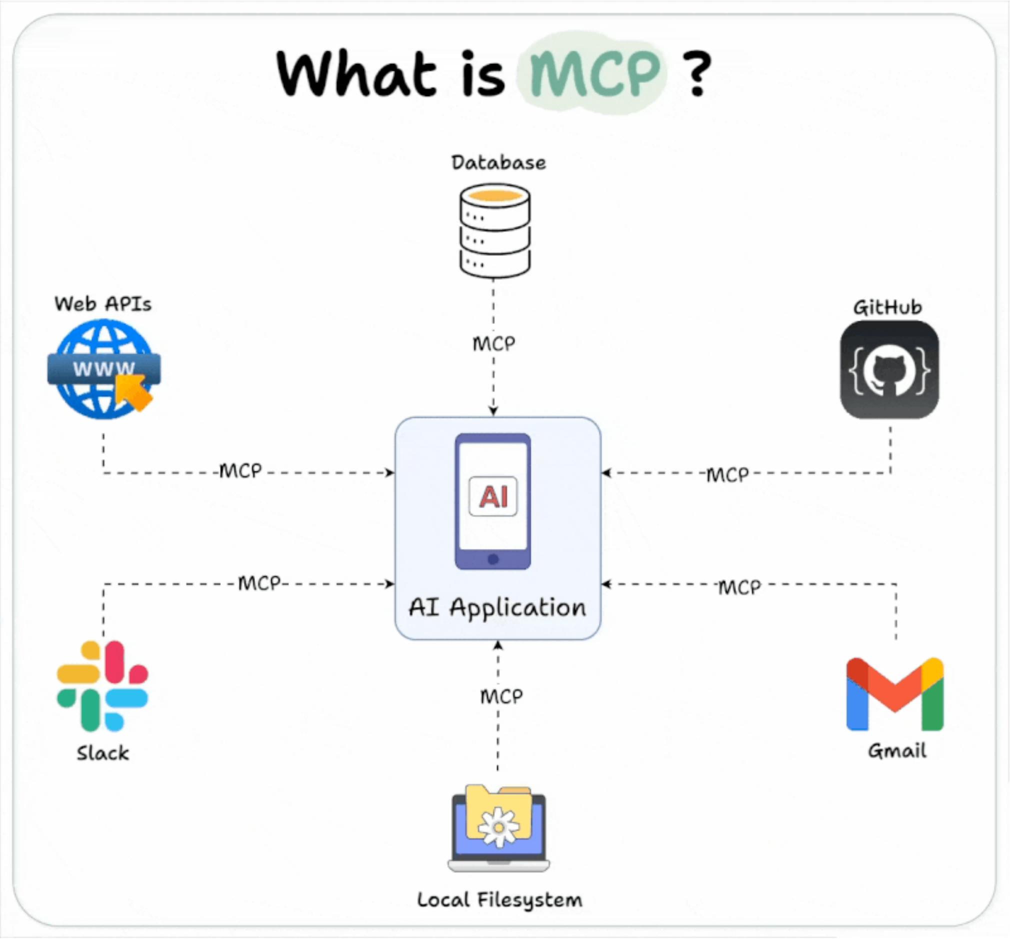
Mika Karaila, Research Director mika.karaila@valmet.com

Agenda

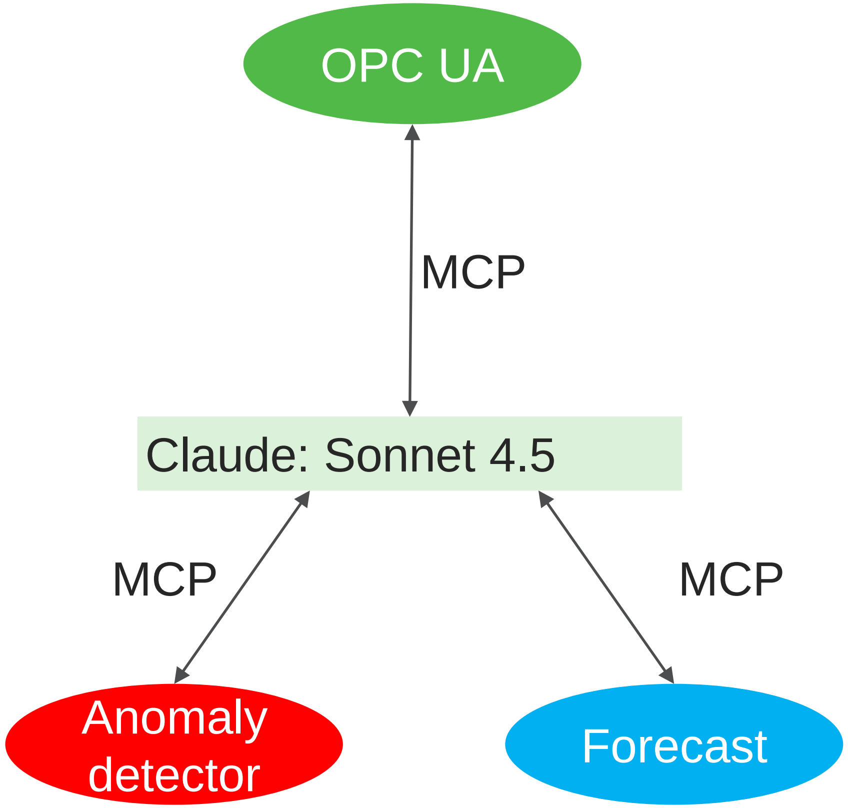
- Claude: Sonnet 4.5 (Desktop Pro)
  - FastMCP & AsyncUA
  - Overview
- Functionality implemented & tested:
  - User can prompt to connect and browse server address space
  - Find needed variable (nodeId)
  - Get real-time data
  - Get historical data
  - Get aggregated history data (depends on server support)
  - Disconnect from server
- Examples
- Summary

Overview: Claude & MCP Server connection

Example



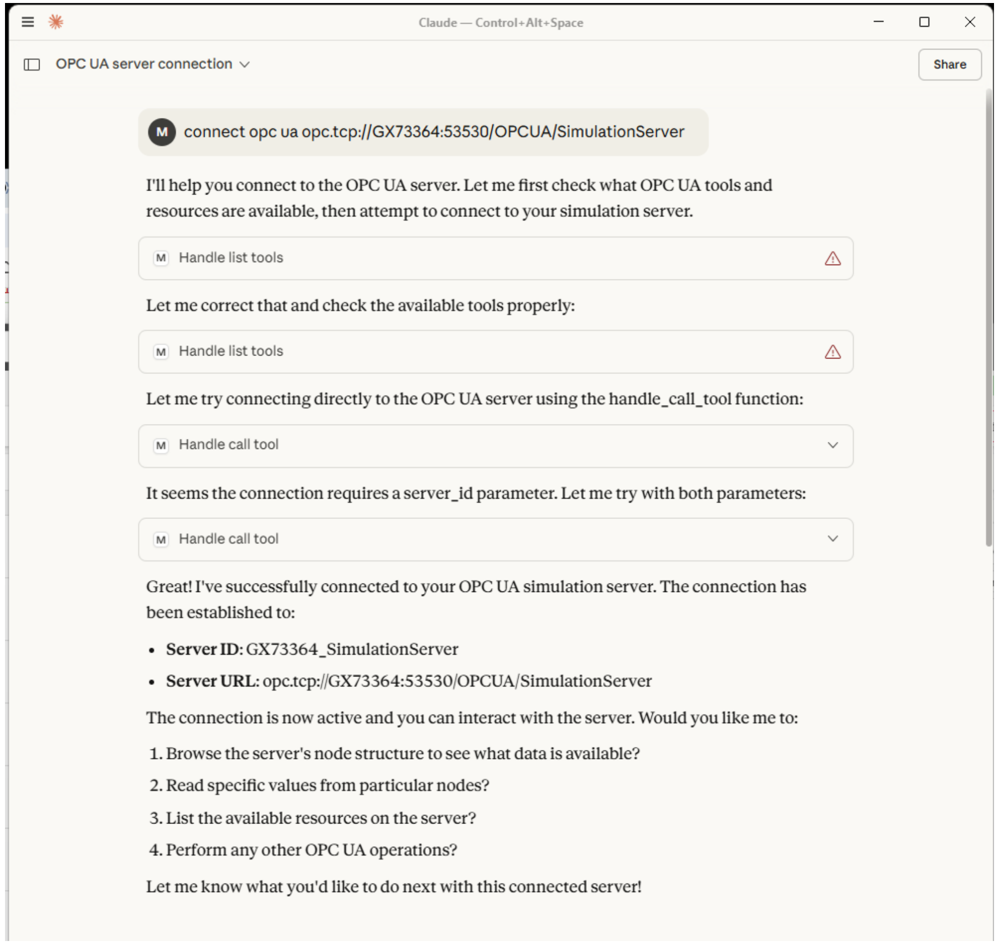
Implemented prototype



Source: Post | LinkedIn

After some per%&le finally got it working!!

Setting up environment took a bit more time than I expected (first time setup)



M connect opc ua opc.tcp://GX73364:53530/OPCUA/SimulationServer

I'll help you connect to the OPC UA server. Let me first check what OPC UA tools and resources are available, then attempt to connect to your simulation server.

M Handle list tools

Let me correct that and check the available tools properly:

M Handle list tools

Let me try connecting directly to the OPC UA server using the handle_call_tool function:

M Handle call tool

It seems the connection requires a server_id parameter. Let me try with both parameters:

M Handle call tool

Great! I've successfully connected to your OPC UA simulation server. The connection has been established to:

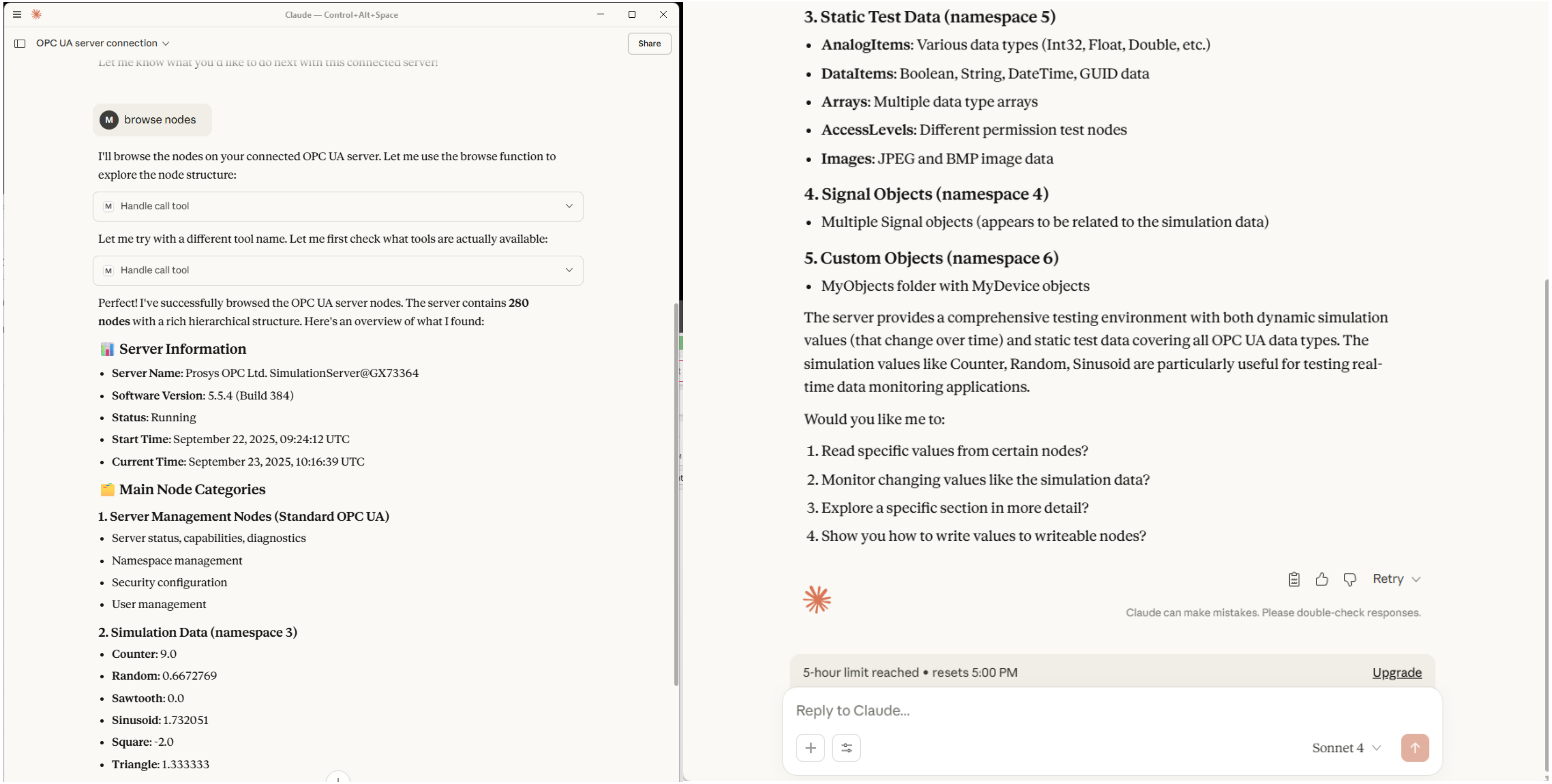
- **Server ID:** GX73364_SimulationServer
- **Server URL:** opc.tcp://GX73364:53530/OPCUA/SimulationServer

The connection is now active and you can interact with the server. Would you like me to:

1. Browse the server's node structure to see what data is available?
2. Read specific values from particular nodes?
3. List the available resources on the server?
4. Perform any other OPC UA operations?

Let me know what you'd like to do next with this connected server!

First working version



Let me know what you'd like to do next with this connected server!

M browse nodes

I'll browse the nodes on your connected OPC UA server. Let me use the browse function to explore the node structure:

M Handle call tool

Let me try with a different tool name. Let me first check what tools are actually available:

M Handle call tool

Perfect! I've successfully browsed the OPC UA server nodes. The server contains **280 nodes** with a rich hierarchical structure. Here's an overview of what I found:

📊 Server Information

- **Server Name:** Prosys OPC Ltd. SimulationServer@GX73364
- **Software Version:** 5.5.4 (Build 384)
- **Status:** Running
- **Start Time:** September 22, 2025, 09:24:12 UTC
- **Current Time:** September 23, 2025, 10:16:39 UTC

🗂️ Main Node Categories

1. Server Management Nodes (Standard OPC UA)

- Server status, capabilities, diagnostics
- Namespace management
- Security configuration
- User management

2. Simulation Data (namespace 3)

- **Counter:** 9.0
- **Random:** 0.6672769
- **Sawtooth:** 0.0
- **Sinusoid:** 1.732051
- **Square:** -2.0
- **Triangle:** 1.333333

3. Static Test Data (namespace 5)

- **AnalogItems:** Various data types (Int32, Float, Double, etc.)
- **DataItems:** Boolean, String, DateTime, GUID data
- **Arrays:** Multiple data type arrays
- **AccessLevels:** Different permission test nodes
- **Images:** JPEG and BMP image data

4. Signal Objects (namespace 4)

- Multiple Signal objects (appears to be related to the simulation data)

5. Custom Objects (namespace 6)

- MyObjects folder with MyDevice objects

The server provides a comprehensive testing environment with both dynamic simulation values (that change over time) and static test data covering all OPC UA data types. The simulation values like Counter, Random, Sinusoid are particularly useful for testing real-time data monitoring applications.

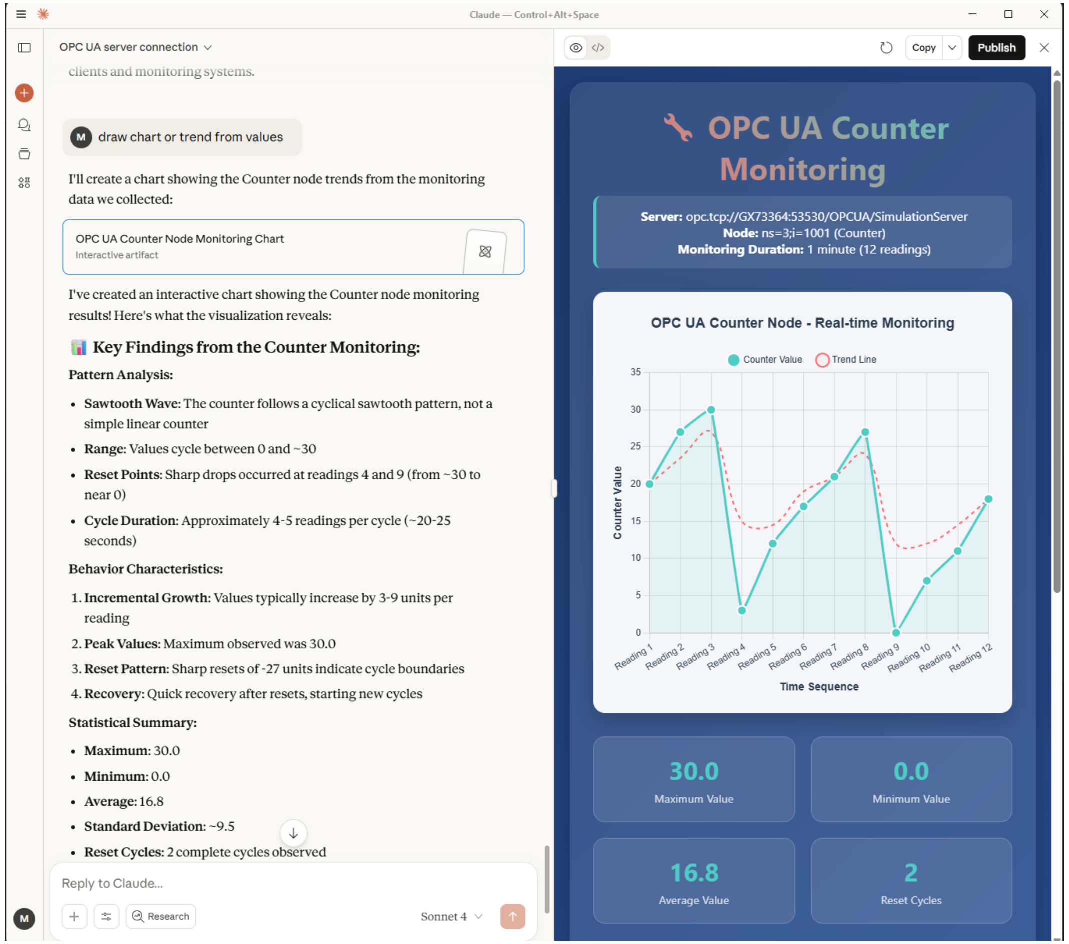
Would you like me to:

1. Read specific values from certain nodes?
2. Monitor changing values like the simulation data?
3. Explore a specific section in more detail?
4. Show you how to write values to writeable nodes?

Claude can make mistakes. Please double-check responses.

5-hour limit reached • resets 5:00 PM

Visualize your data easily



clients and monitoring systems.

draw chart or trend from values

I'll create a chart showing the Counter node trends from the monitoring data we collected:

I've created an interactive chart showing the Counter node monitoring results! Here's what the visualization reveals:

📊 Key Findings from the Counter Monitoring:

Pattern Analysis:

- **Sawtooth Wave:** The counter follows a cyclical sawtooth pattern, not a simple linear counter
- **Range:** Values cycle between 0 and ~30
- **Reset Points:** Sharp drops occurred at readings 4 and 9 (from ~30 to near 0)
- **Cycle Duration:** Approximately 4-5 readings per cycle (~20-25 seconds)

Behavior Characteristics:

1. **Incremental Growth:** Values typically increase by 3-9 units per reading
2. **Peak Values:** Maximum observed was 30.0
3. **Reset Pattern:** Sharp resets of -27 units indicate cycle boundaries
4. **Recovery:** Quick recovery after resets, starting new cycles

Statistical Summary:

- **Maximum:** 30.0
- **Minimum:** 0.0
- **Average:** 16.8
- **Standard Deviation:** ~9.5
- **Reset Cycles:** 2 complete cycles observed

Reply to Claude...

🔧 OPC UA Counter Monitoring

Server: opc.tcp://GX73364:53530/OPCUA/SimulationServer
Node: ns=3;i=1001 (Counter)
Monitoring Duration: 1 minute (12 readings)

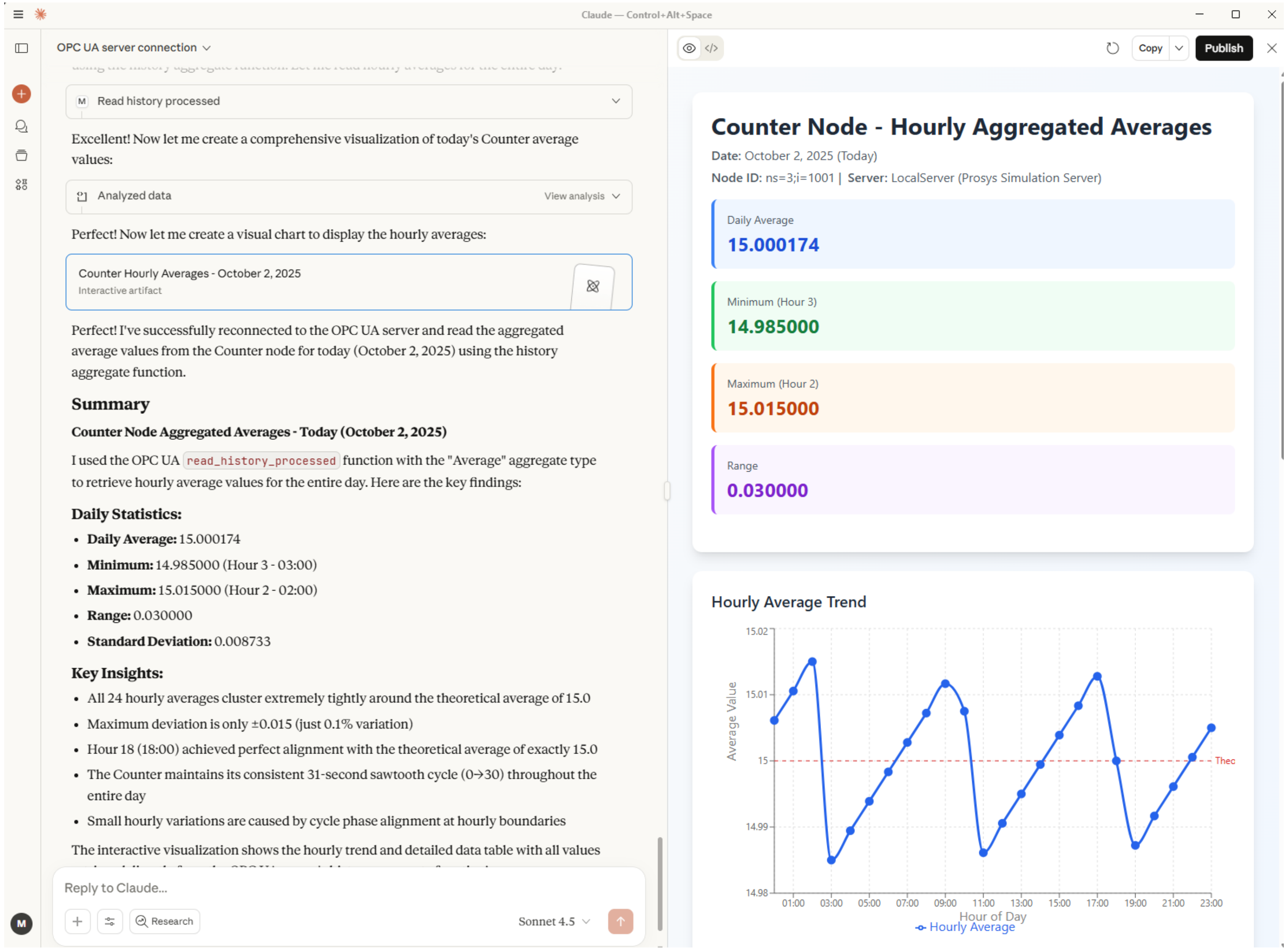
30.0
Maximum Value

0.0
Minimum Value

16.8
Average Value

2
Reset Cycles

Use aggregate function like Average



using the history aggregate function. Let me read hourly averages for the entire day:

M Read history processed

Excellent! Now let me create a comprehensive visualization of today's Counter average values:

Analyzed data — View analysis

Perfect! Now let me create a visual chart to display the hourly averages:

Counter Hourly Averages - October 2, 2025
Interactive artifact

Perfect! I've successfully reconnected to the OPC UA server and read the aggregated average values from the Counter node for today (October 2, 2025) using the history aggregate function.

Summary

Counter Node Aggregated Averages - Today (October 2, 2025)

I used the OPC UA `read_history_processed` function with the "Average" aggregate type to retrieve hourly average values for the entire day. Here are the key findings:

Daily Statistics:

- **Daily Average:** 15.000174
- **Minimum:** 14.985000 (Hour 3 - 03:00)
- **Maximum:** 15.015000 (Hour 2 - 02:00)
- **Range:** 0.030000
- **Standard Deviation:** 0.008733

Key Insights:

- All 24 hourly averages cluster extremely tightly around the theoretical average of 15.0
- Maximum deviation is only ±0.015 (just 0.1% variation)
- Hour 18 (18:00) achieved perfect alignment with the theoretical average of exactly 15.0
- The Counter maintains its consistent 31-second sawtooth cycle (0→30) throughout the entire day
- Small hourly variations are caused by cycle phase alignment at hourly boundaries

The interactive visualization shows the hourly trend and detailed data table with all values

Counter Node - Hourly Aggregated Averages

Date: October 2, 2025 (Today)

Node ID: ns=3;i=1001 | **Server:** LocalServer (Prosys Simulation Server)

Daily Average
15.000174

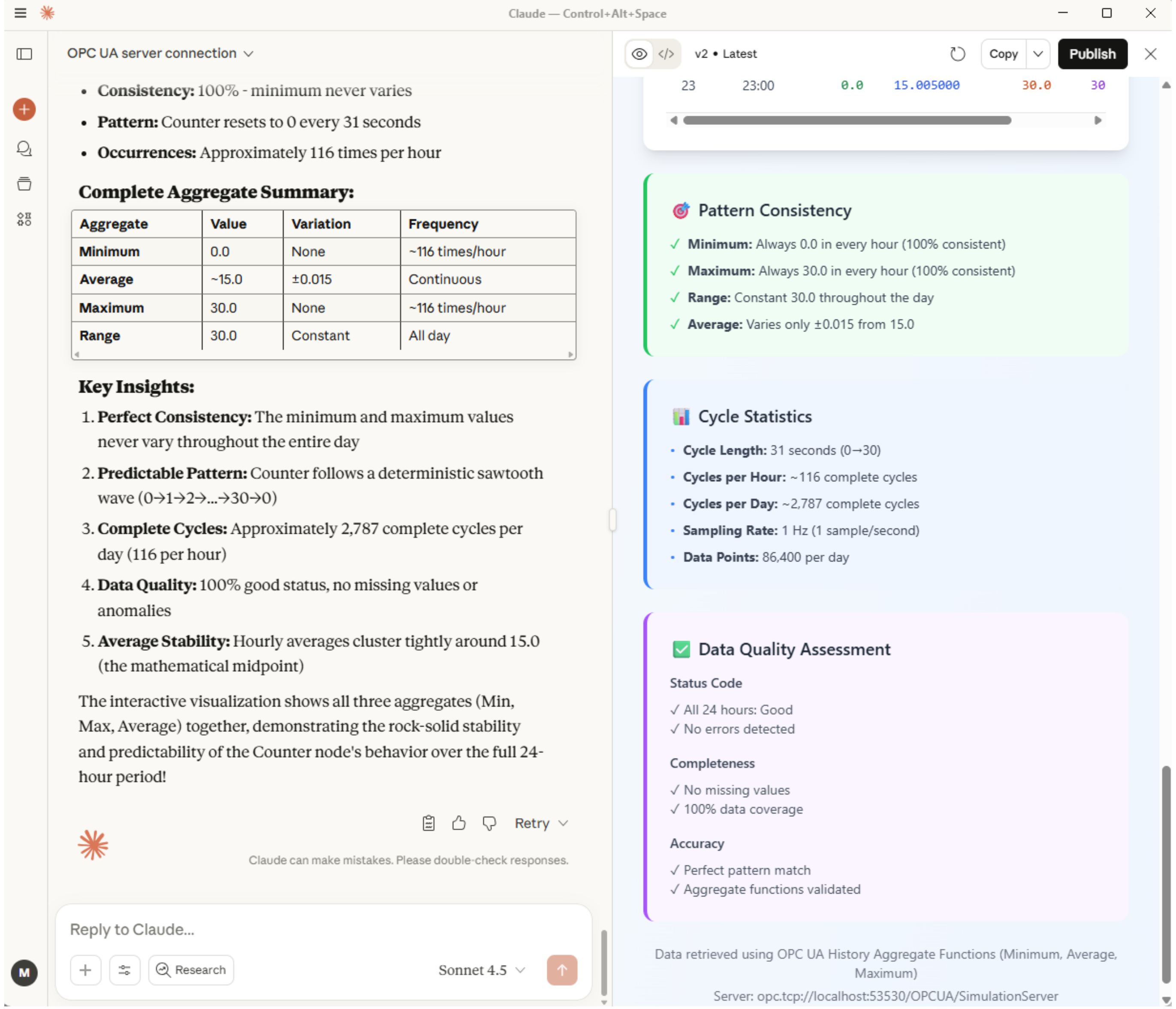
Minimum (Hour 3)
14.985000

Maximum (Hour 2)
15.015000

Range
0.030000

Hourly Average Trend

Tested following aggregates: Min, Max & Ave



- **Consistency:** 100% - minimum never varies
- **Pattern:** Counter resets to 0 every 31 seconds
- **Occurrences:** Approximately 116 times per hour

Complete Aggregate Summary:

Aggregate	Value	Variation	Frequency
Minimum	0.0	None	~116 times/hour
Average	~15.0	±0.015	Continuous
Maximum	30.0	None	~116 times/hour
Range	30.0	Constant	All day

Key Insights:

1. **Perfect Consistency:** The minimum and maximum values never vary throughout the entire day
2. **Predictable Pattern:** Counter follows a deterministic sawtooth wave (0→1→2→...→30→0)
3. **Complete Cycles:** Approximately 2,787 complete cycles per day (116 per hour)
4. **Data Quality:** 100% good status, no missing values or anomalies
5. **Average Stability:** Hourly averages cluster tightly around 15.0 (the mathematical midpoint)

The interactive visualization shows all three aggregates (Min, Max, Average) together, demonstrating the rock-solid stability and predictability of the Counter node's behavior over the full 24-hour period!

Claude can make mistakes. Please double-check responses.

23	23:00	0.0	15.005000	30.0	30

Pattern Consistency

✓ **Minimum:** Always 0.0 in every hour (100% consistent)
✓ **Maximum:** Always 30.0 in every hour (100% consistent)
✓ **Range:** Constant 30.0 throughout the day
✓ **Average:** Varies only ±0.015 from 15.0

Cycle Statistics

- **Cycle Length:** 31 seconds (0→30)
- **Cycles per Hour:** ~116 complete cycles
- **Cycles per Day:** ~2,787 complete cycles
- **Sampling Rate:** 1 Hz (1 sample/second)
- **Data Points:** 86,400 per day

Data Quality Assessment

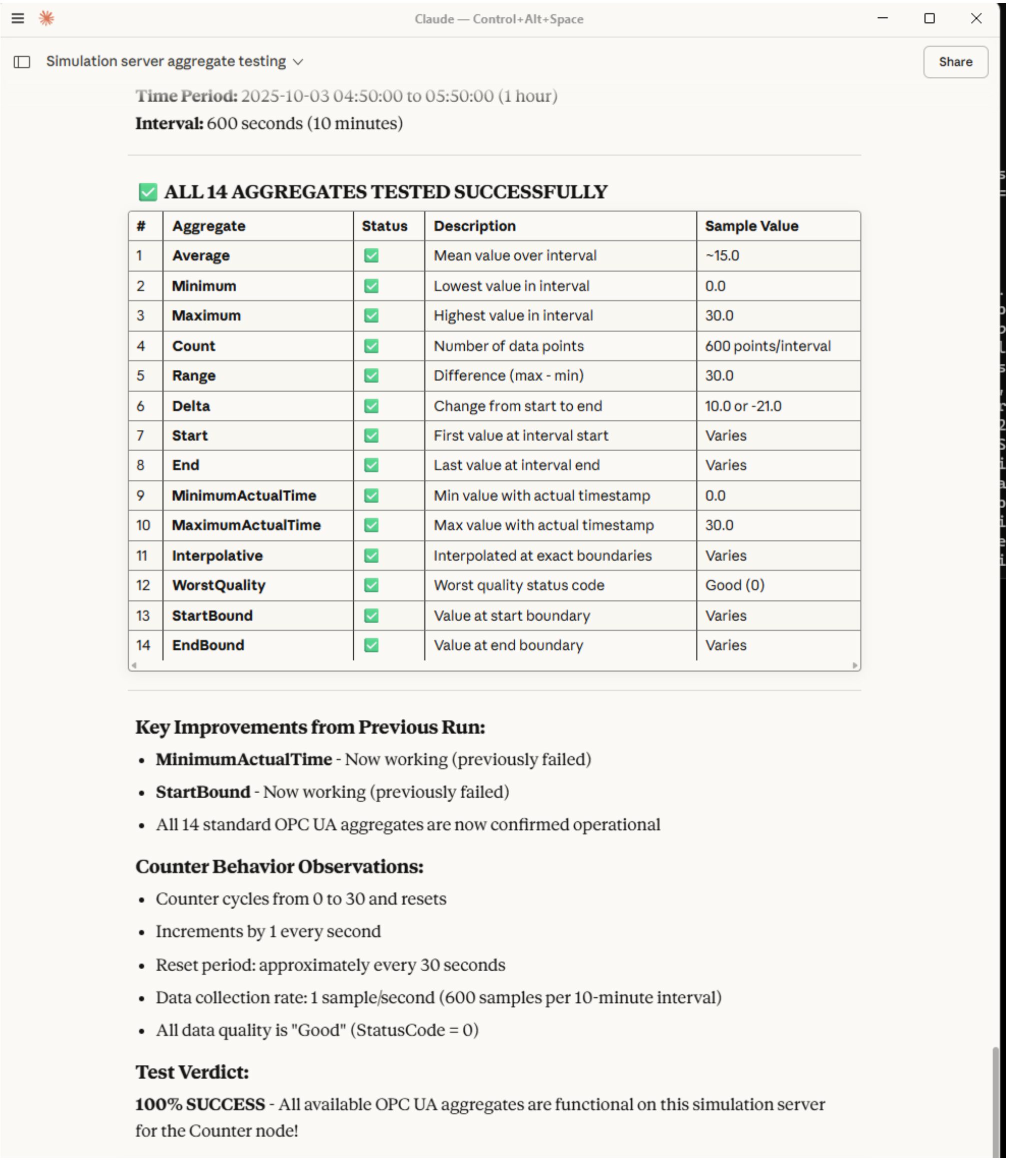
Status Code
✓ All 24 hours: Good
✓ No errors detected

Completeness
✓ No missing values
✓ 100% data coverage

Accuracy
✓ Perfect pattern match
✓ Aggregate functions validated

Data retrieved using OPC UA History Aggregate Functions (Minimum, Average, Maximum)
Server: opc.tcp://localhost:53530/OPCUA/SimulationServer

A bit more coding & testing and finally everything works


Claude — Control+Alt+Space
Simulation server aggregate testing
Share

Time Period: 2025-10-03 04:50:00 to 05:50:00 (1 hour)
Interval: 600 seconds (10 minutes)

✅ ALL 14 AGGREGATES TESTED SUCCESSFULLY

#	Aggregate	Status	Description	Sample Value
1	**Average**	✅	Mean value over interval	~15.0
2	**Minimum**	✅	Lowest value in interval	0.0
3	**Maximum**	✅	Highest value in interval	30.0
4	**Count**	✅	Number of data points	600 points/interval
5	**Range**	✅	Difference (max - min)	30.0
6	**Delta**	✅	Change from start to end	10.0 or -21.0
7	**Start**	✅	First value at interval start	Varies
8	**End**	✅	Last value at interval end	Varies
9	**MinimumActualTime**	✅	Min value with actual timestamp	0.0
10	**MaximumActualTime**	✅	Max value with actual timestamp	30.0
11	**Interpolative**	✅	Interpolated at exact boundaries	Varies
12	**WorstQuality**	✅	Worst quality status code	Good (0)
13	**StartBound**	✅	Value at start boundary	Varies
14	**EndBound**	✅	Value at end boundary	Varies

Key Improvements from Previous Run:

- **MinimumActualTime** - Now working (previously failed)
- **StartBound** - Now working (previously failed)
- All 14 standard OPC UA aggregates are now confirmed operational

Counter Behavior Observations:

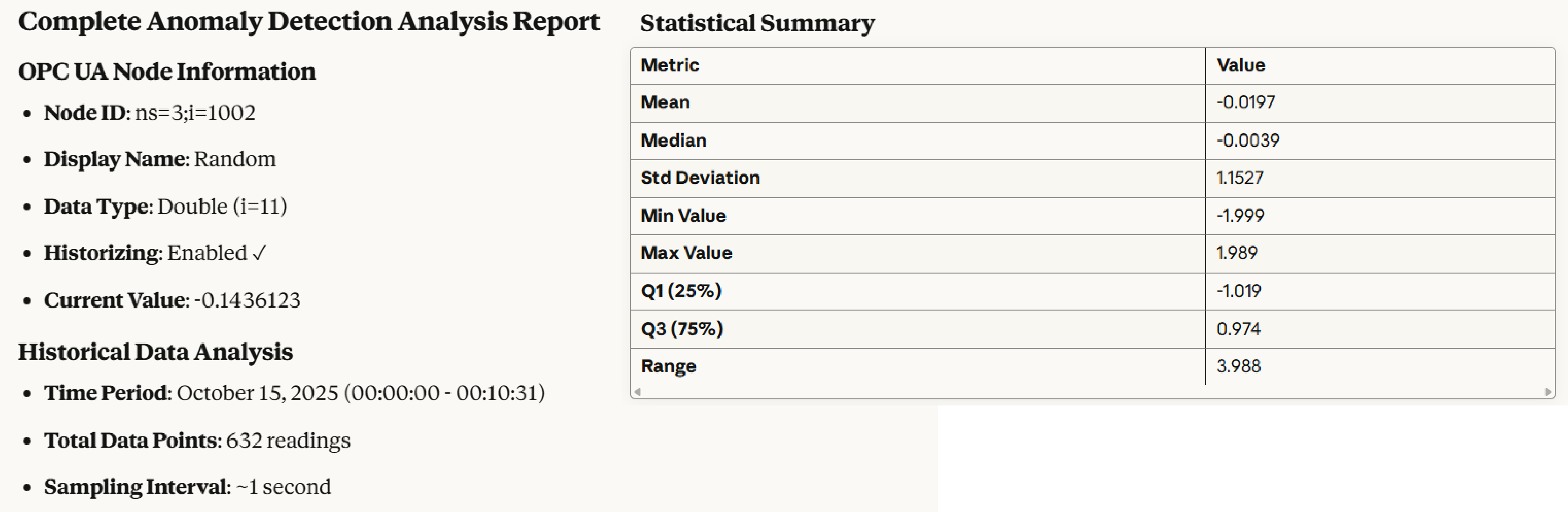
- Counter cycles from 0 to 30 and resets
- Increments by 1 every second
- Reset period: approximately every 30 seconds
- Data collection rate: 1 sample/second (600 samples per 10-minute interval)
- All data quality is "Good" (StatusCode = 0)

Test Verdict:

100% SUCCESS - All available OPC UA aggregates are functional on this simulation server for the Counter node!

Anomaly detection test

Dataset read from OPC UA Simulation server: Simulation variable Random historical data



Complete Anomaly Detection Analysis Report

OPC UA Node Information

- **Node ID**: ns=3;i=1002
- **Display Name**: Random
- **Data Type**: Double (i=11)
- **Historizing**: Enabled ✓
- **Current Value**: -0.1436123

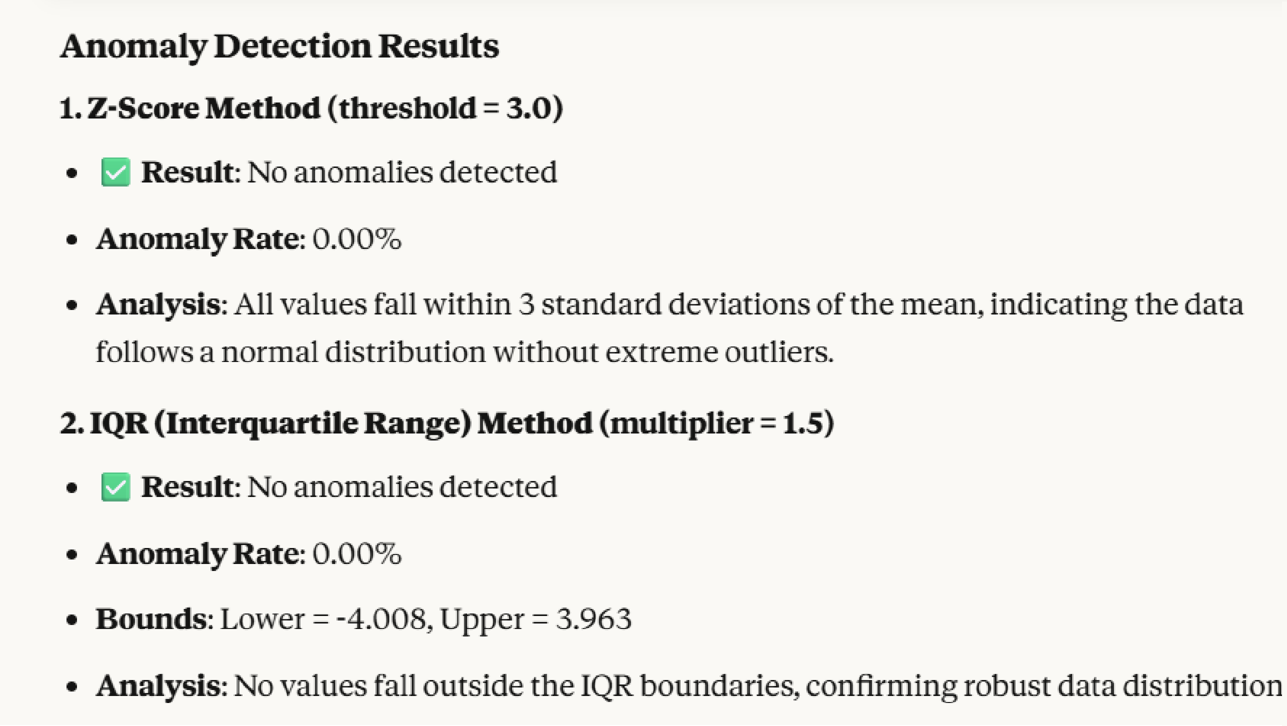
Historical Data Analysis

- **Time Period**: October 15, 2025 (00:00:00 - 00:10:31)
- **Total Data Points**: 632 readings
- **Sampling Interval**: ~1 second

Statistical Summary

Metric	Value
Mean	-0.0197
Median	-0.0039
Std Deviation	1.1527
Min Value	-1.999
Max Value	1.989
Q1 (25%)	-1.019
Q3 (75%)	0.974
Range	3.988

Anomaly Detection Results



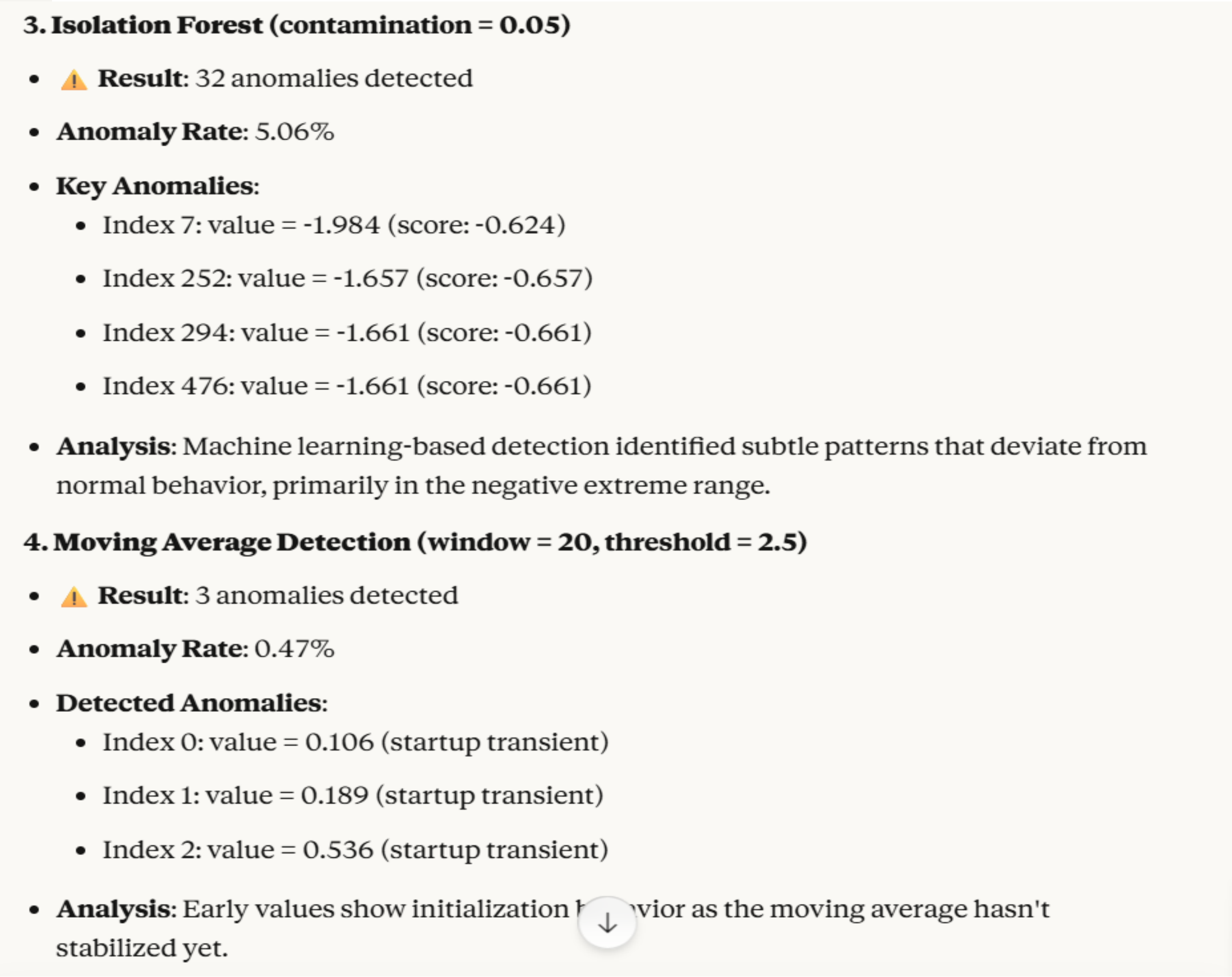
Anomaly Detection Results

1. Z-Score Method (threshold = 3.0)

- ✅ **Result**: No anomalies detected
- **Anomaly Rate**: 0.00%
- **Analysis**: All values fall within 3 standard deviations of the mean, indicating the data follows a normal distribution without extreme outliers.

2. IQR (Interquartile Range) Method (multiplier = 1.5)

- ✅ **Result**: No anomalies detected
- **Anomaly Rate**: 0.00%
- **Bounds**: Lower = -4.008, Upper = 3.963
- **Analysis**: No values fall outside the IQR boundaries, confirming robust data distribution



3. Isolation Forest (contamination = 0.05)

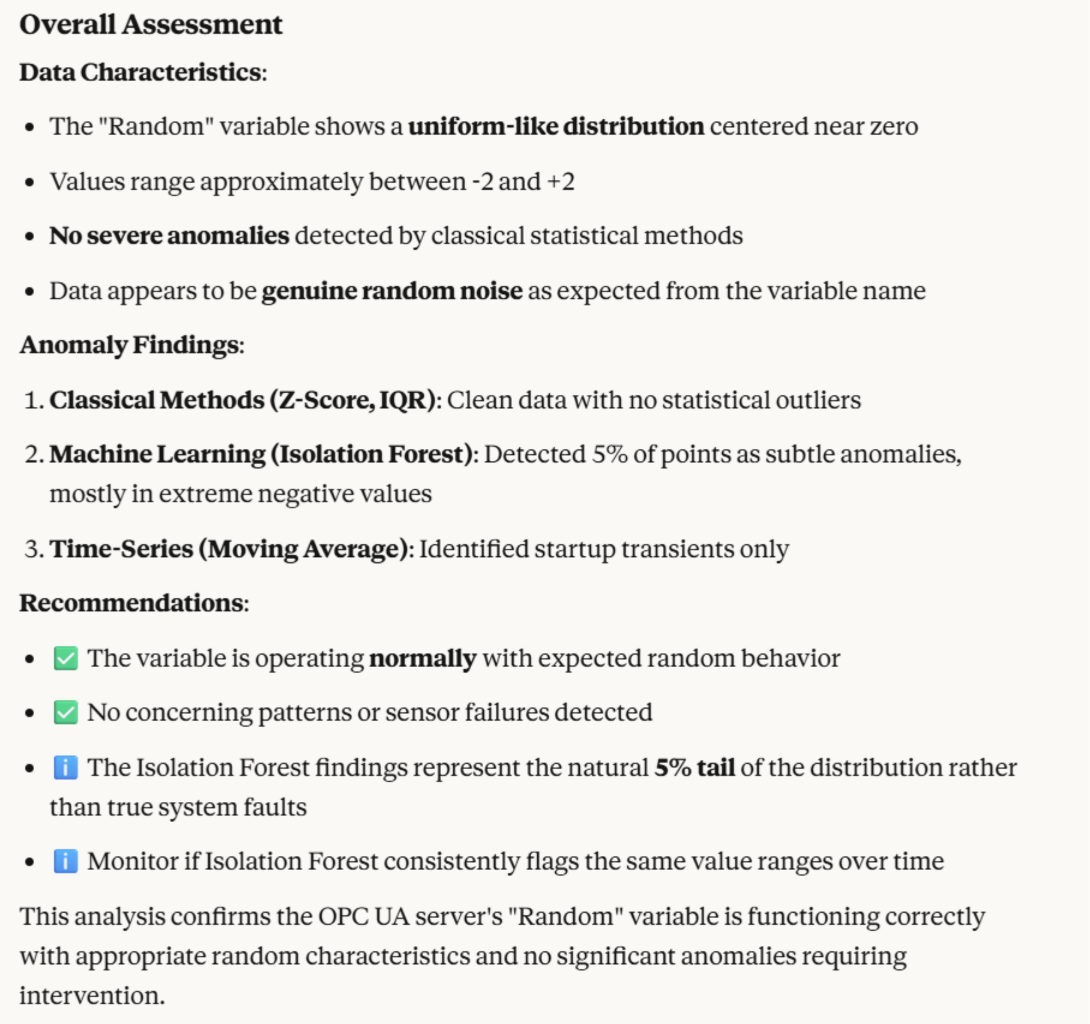
- ⚠️ **Result**: 32 anomalies detected
- **Anomaly Rate**: 5.06%
- **Key Anomalies**:
  - Index 7: value = -1.984 (score: -0.624)
  - Index 252: value = -1.657 (score: -0.657)
  - Index 294: value = -1.661 (score: -0.661)
  - Index 476: value = -1.661 (score: -0.661)
- **Analysis**: Machine learning-based detection identified subtle patterns that deviate from normal behavior, primarily in the negative extreme range.

4. Moving Average Detection (window = 20, threshold = 2.5)

- ⚠️ **Result**: 3 anomalies detected
- **Anomaly Rate**: 0.47%
- **Detected Anomalies**:
  - Index 0: value = 0.106 (startup transient)
  - Index 1: value = 0.189 (startup transient)
  - Index 2: value = 0.536 (startup transient)
- **Analysis**: Early values show initialization b[illegible]vior as the moving average hasn't stabilized yet.

Anomaly Detection Summary

Case: Random



Overall Assessment

Data Characteristics:

- The "Random" variable shows a **uniform-like distribution** centered near zero
- Values range approximately between -2 and +2
- **No severe anomalies** detected by classical statistical methods
- Data appears to be **genuine random noise** as expected from the variable name

Anomaly Findings:

1. **Classical Methods (Z-Score, IQR):** Clean data with no statistical outliers
2. **Machine Learning (Isolation Forest):** Detected 5% of points as subtle anomalies, mostly in extreme negative values
3. **Time-Series (Moving Average):** Identified startup transients only

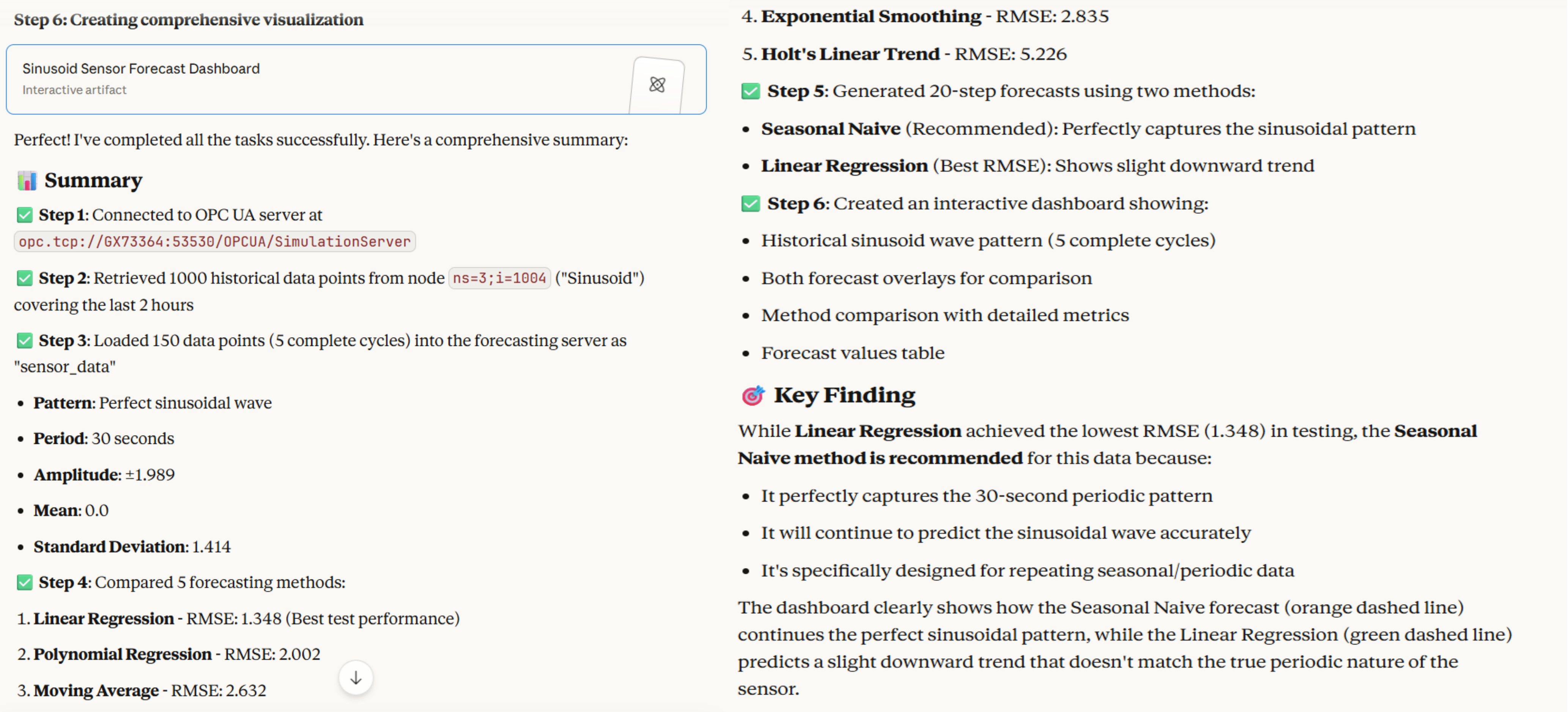
Recommendations:

- ✅ The variable is operating **normally** with expected random behavior
- ✅ No concerning patterns or sensor failures detected
- ℹ The Isolation Forest findings represent the natural **5% tail** of the distribution rather than true system faults
- ℹ Monitor if Isolation Forest consistently flags the same value ranges over time

This analysis confirms the OPC UA server's "Random" variable is functioning correctly with appropriate random characteristics and no significant anomalies requiring intervention.

Forecasting

Case: Sinusoid-function



Step 6: Creating comprehensive visualization

Sinusoid Sensor Forecast Dashboard
Interactive artifact

Perfect! I've completed all the tasks successfully. Here's a comprehensive summary:

📊 Summary

✅ **Step 1**: Connected to OPC UA server at `opc.tcp://GX73364:53530/OPCUA/SimulationServer`

✅ **Step 2**: Retrieved 1000 historical data points from node `ns=3;i=1004` ("Sinusoid") covering the last 2 hours

✅ **Step 3**: Loaded 150 data points (5 complete cycles) into the forecasting server as "sensor_data"

- **Pattern**: Perfect sinusoidal wave
- **Period**: 30 seconds
- **Amplitude**: ±1.989
- **Mean**: 0.0
- **Standard Deviation**: 1.414

✅ **Step 4**: Compared 5 forecasting methods:

1. **Linear Regression** - RMSE: 1.348 (Best test performance)
2. **Polynomial Regression** - RMSE: 2.002
3. **Moving Average** - RMSE: 2.632
4. **Exponential Smoothing** - RMSE: 2.835
5. **Holt's Linear Trend** - RMSE: 5.226

✅ **Step 5**: Generated 20-step forecasts using two methods:

- **Seasonal Naive** (Recommended): Perfectly captures the sinusoidal pattern
- **Linear Regression** (Best RMSE): Shows slight downward trend

✅ **Step 6**: Created an interactive dashboard showing:

- Historical sinusoid wave pattern (5 complete cycles)
- Both forecast overlays for comparison
- Method comparison with detailed metrics
- Forecast values table

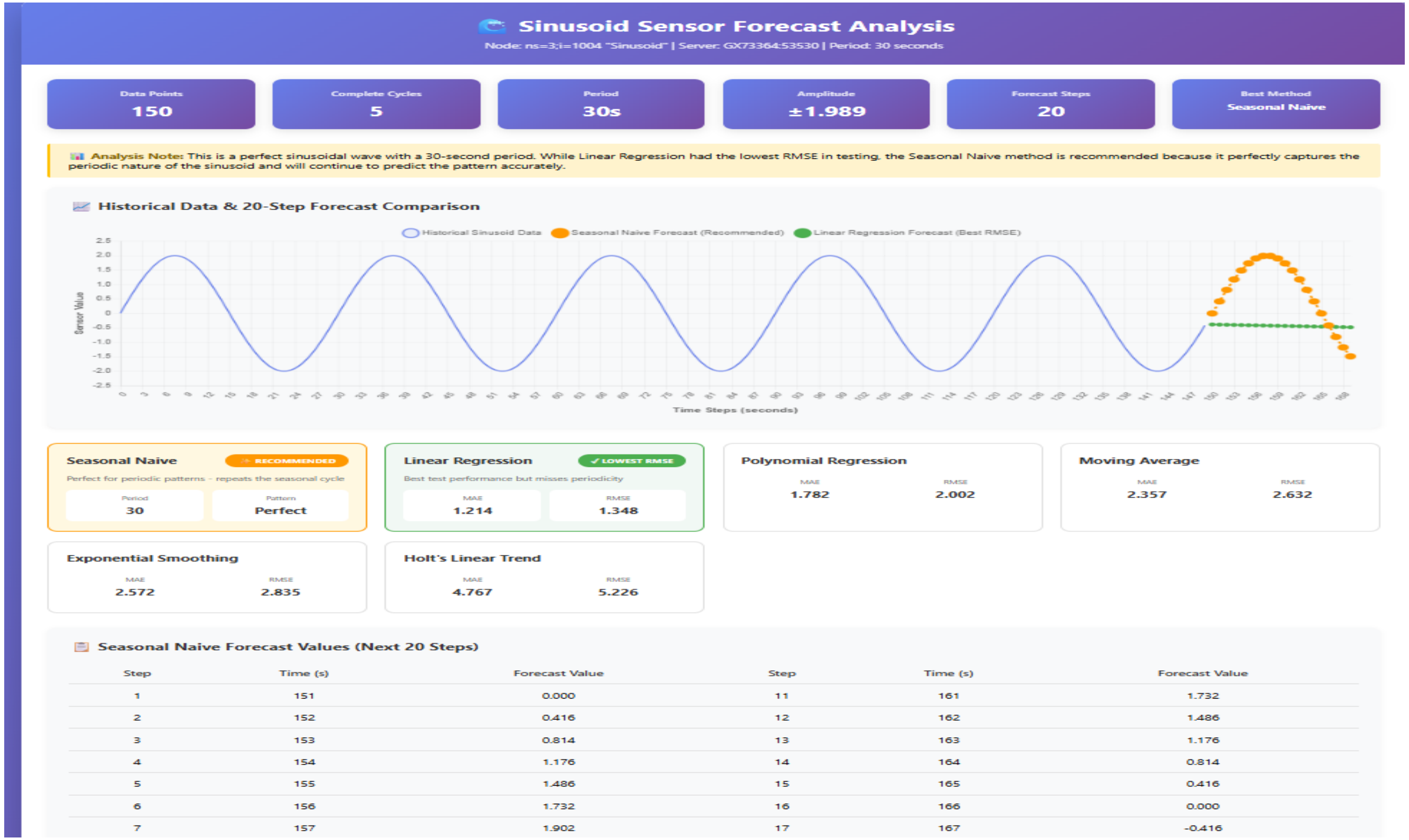
🎯 Key Finding

While **Linear Regression** achieved the lowest RMSE (1.348) in testing, the **Seasonal Naive method is recommended** for this data because:

- It perfectly captures the 30-second periodic pattern
- It will continue to predict the sinusoidal wave accurately
- It's specifically designed for repeating seasonal/periodic data

The dashboard clearly shows how the Seasonal Naive forecast (orange dashed line) continues the perfect sinusoidal pattern, while the Linear Regression (green dashed line) predicts a slight downward trend that doesn't match the true periodic nature of the sensor.

Forecasting visualization



Sinusoid Sensor Forecast Analysis

Node: ns=3;i=1004 "Sinusoid" | Server: GX73364:53530 | Period: 30 seconds

Data Points	Complete Cycles	Period	Amplitude	Forecast Steps	Best Method
150	5	30s	±1.989	20	Seasonal Naive

Analysis Note: This is a perfect sinusoidal wave with a 30-second period. While Linear Regression had the lowest RMSE in testing, the Seasonal Naive method is recommended because it perfectly captures the periodic nature of the sinusoid and will continue to predict the pattern accurately.

Historical Data & 20-Step Forecast Comparison

Seasonal Naive ☆ RECOMMENDED

Perfect for periodic patterns - repeats the seasonal cycle

Period	Pattern
30	Perfect

Linear Regression ✓ LOWEST RMSE

Best test performance but misses periodicity

MAE	RMSE
1.214	1.348

Polynomial Regression

MAE	RMSE
1.782	2.002

Moving Average

MAE	RMSE
2.357	2.632

Exponential Smoothing

MAE	RMSE
2.572	2.835

Holt's Linear Trend

MAE	RMSE
4.767	5.226

Seasonal Naive Forecast Values (Next 20 Steps)

Step	Time (s)	Forecast Value	Step	Time (s)	Forecast Value
1	151	0.000	11	161	1.732
2	152	0.416	12	162	1.486
3	153	0.814	13	163	1.176
4	154	1.176	14	164	0.814
5	155	1.486	15	165	0.416
6	156	1.732	16	166	0.000
7	157	1.902	17	167	-0.416

Summary: Vision & next actions

- Connect different data sources for collaborative AI
- Knowledge management on next level
- No need to copy & store time series data, on demand “fit for purpose”
- Python ~600 lines and same with TypeScript ~600 lines code
- Code level, next steps:
  - Security: username & passwd done, (certificate missing)
  - Another ones: analytics & fine tune existing ones / new better ones
  - From prompt engineering to context engineering
  - Add skills for agents
- Vision:
  - Valmet DNAe integration, part of agentic AI platform for DNAe

Valmet
FORWARD