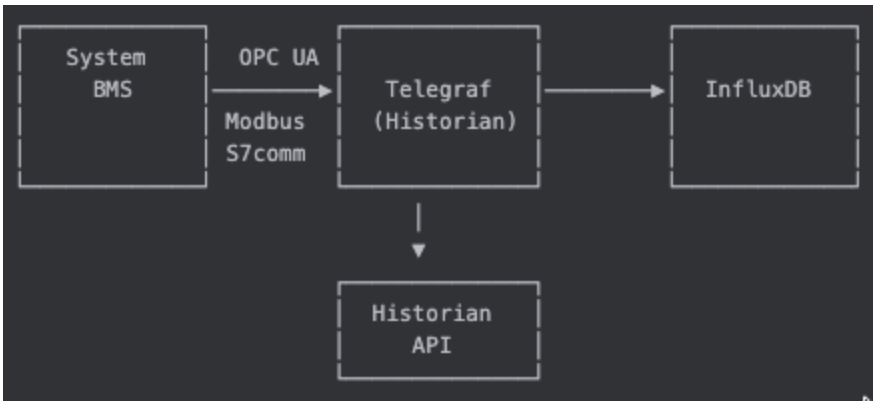
Dokumentacja integracji systemu BMS z Historian

1. Wprowadzenie

1.1 Przegląd systemu Historian

Historian to system zbierania i archiwizacji danych przemysłowych, który umożliwia integrację z systemami BMS (Building Management System) poprzez obsługę standardowych protokołów komunikacji przemysłowej. System wykorzystuje Telegraf jako agent zbierający dane oraz InfluxDB jako magazyn danych czasowych.

1.2 Komunikacja części systemu



1.3 Cel dokumentacji

Niniejsza dokumentacja ma na celu umożliwienie dostawcom systemów BMS integracji z systemem Historian poprzez:

- Opis obsługiwanych protokołów komunikacyjnych
- Szczegółowe wymagania techniczne
- Przykłady konfiguracji
- Dokumentację API
- Best practices i troubleshooting

1.4 Wymagania wstępne

- System BMS z obsługą jednego z protokołów: OPC UA, Modbus TCP lub S7comm
- Dostęp sieciowy do systemu Historian
- Poświadczenia dostępowe (JWT token lub API token) *opcja

2. Obsługiwane protokoły komunikacyjne

2.1 OPC UA (OPC Unified Architecture)

OPC UA jest zalecanym protokołem dla systemów wymagających wysokiej wydajności i niezawodności.

2.1.1 Tryby pracy

Subscription Mode (opcua_listener) - ZALECANY
- Tryb subskrypcji zapewnia najwyższą wydajność
- Dane są przesyłane tylko w przypadku zmiany wartości
- Domyślnie włączony dla nowych maszyn OPC UA
- Konfiguracja: useSubscription: true

Polling Mode (opcua)
- Tryb odpytywania jako alternatywa
- Dane są odczytywane w regularnych odstępach czasu
- Wymaga większej przepustowości sieci
- Konfiguracja: useSubscription: false

2.1.2 Wymagane parametry połączenia

Parametr	Typ	Wymagany	Domyślna wartość	Opis
endpoint	string	Tak	-	Adres URL serwera OPC UA (np. opc.tcp://192.168.1.100:4840)
connect_timeout	string	Nie	10s	Timeout połączenia
request_timeout	string	Nie	5s	Timeout żądania
security_policy	string	Nie	None	Polityka bezpieczeństwa (None, Basic128Rsa15, Basic256, Basic256Sha256)
security_mode	string	Nie	None	Tryb bezpieczeństwa (None, Sign, SignAndEncrypt)
certificate	string	Nie	-	Ścieżka do certyfikatu (wymagane dla trybów z szyfrowaniem)
private_key	string	Nie	-	Ścieżka do klucza prywatnego (wymagane dla

				trybów z szyfrowaniem)
subscription_interval	string	Nie	1s	Interwał subskrypcji (tylko dla subscription mode)
sampling_interval	string	Nie	-	Interwał próbkowania (tylko dla polling mode)
startup_error_behavior	string	Nie	retry	Zachowanie przy błędzie startowym (error, retry, ignore)

2.1.3 Format adresów węzłów

Każdy sygnał OPC UA wymaga określenia węzła poprzez:

- namespace: Numer przestrzeni nazw (np. "3", "4")
- identifier_type: Typ identyfikatora:
  - "s" - String identifier
  - "i" - Numeric identifier
  - "g" - GUID identifier
  - "b" - ByteString identifier
- identifier: Wartość identyfikatora

Przykłady:

namespace: "3"
identifier_type: "s"
identifier: "\"DB500 MES-Daten\".\"Glas_Process_Data\".\"Glass_Data\".\"Glass_ID\""

namespace: "4"
identifier_type: "i"
identifier: "741"

2.1.4 Deadband configuration

Deadband pozwala na zmniejszenie ilości przesyłanych danych poprzez ignorowanie małych zmian wartości.

- deadband_type: "absolute" (obecnie jedyny obsługiwany typ)
- deadband_value: Wartość progowa (np. 0.1, 1.0)

Przykład:

```
{
  "protocolSpecific": {
    "namespace": "3",
    "identifier_type": "s",
```

```
    "identifier": "Temperature",
    "deadband": 0.5
  }
}
```

2.2 Modbus TCP

Modbus TCP jest szeroko stosowanym protokołem w automatyce przemysłowej.

2.2.1 Wymagane parametry połączenia

Parametr	Typ	Wymagany	Domyślna wartość	Opis
controller	string	Tak	-	Adres kontrolera Modbus (np. tcp://192.168.1.100:502)
slave_id	number	Nie	1	ID urządzenia Modbus (1-247)
timeout	string	Nie	1s	Timeout połączenia
startup_error_behavior	string	Nie	retry	Zachowanie przy błędzie startowym

2.2.2 Typy rejestrów

- holding: Rejestry holding (odczyt/zapis) - adresy 0-65535
- input: Rejestry wejściowe (tylko odczyt) - adresy 0-65535
- coil: Cewki (odczyt/zapis) - adresy 0-65535
- discrete: Wejścia dyskretne (tylko odczyt) - adresy 0-65535

2.2.3 Typy danych

Obsługiwane typy danych Modbus:

Typ danych	Rozmiar	Opis
INT8L	1 bajt	8-bit integer (low byte)
INT8H	1 bajt	8-bit integer (high byte)
UINT8L	1 bajt	8-bit unsigned integer (low byte)
UINT8H	1 bajt	8-bit unsigned integer (high byte)
INT16	2 bajty	16-bit integer
UINT16	2 bajty	16-bit unsigned integer
INT32	4 bajty	32-bit integer
UINT32	4 bajty	32-bit unsigned integer
FLOAT32	4 bajty	32-bit floating point
FLOAT64	8 bajty	64-bit floating point

2.2.4 Byte Order

Kolejność bajtów określa sposób interpretacji wielobajtowych wartości:

- INT16/UINT16: "AB" (domyślnie)
- INT32/UINT32/FLOAT32: "ABCD" (domyślnie)
- FLOAT64: "ABCDEFGH" (domyślnie)

Inne opcje: "BA", "DCBA", "BADC", "CDAB", "BAFE", "FEDC", "HGFEDCBA"

2.2.5 Scale Factor

Współczynnik skalowania pozwala na przeliczanie wartości surowych na wartości rzeczywiste.
Przykład:

```
{
  "protocolSpecific": {
    "registerType": "holding",
    "address": 0,
    "dataType": "INT16",
    "byteOrder": "AB",
    "scale": 0.1
  }
}
```

Wartość surowa 100 zostanie przeliczona na 10.0 (100 × 0.1).

2.3 S7comm (Siemens S7)

S7comm umożliwia komunikację z sterownikami Siemens S7.

2.3.1 Wymagane parametry połączenia

Parametr	Typ	Wymagany	Domyślna wartość	Opis
server	string	Tak	-	Adres serwera S7 (np. 192.168.1.100:102)
rack	number	Nie	0	Numer szafy (rack)
slot	number	Nie	2	Numer slotu
timeout	string	Nie	10s	Timeout połączenia
startup_error_behavior	string	Nie	retry	Zachowanie przy błędzie startowym

2.3.2 Format adresów

Nowy format (zalecany):

DB7.I0 - Data Block 7, Input byte 0
DB10.Q4 - Data Block 10, Output byte 4
DB5.M2 - Data Block 5, Memory byte 2

Stary format (obsługiwany):

- db_number: Numer bloku danych (np. 7)
- start: Offset początkowy (np. 0)
- data_type: Typ danych (np. INT, REAL)

3. Wymagania techniczne

Ustawienia bezpośrednio w panelu historian.

3.1 Machine Code (machine_code)

Machine Code jest wymaganym tagiem dla każdej maszyny w systemie Historian. Może być ustalany przez operatora historian.

3.1.1 Wymagania

- Format: 4-8 znaków alfanumerycznych (tylko litery i cyfry)
- Znaki specjalne: Niedozwolone
- Unikalność: Powinien być unikalny w obrębie systemu
- Użycie: Identyfikator maszyny w systemie SCADA

3.1.2 Przykłady poprawnych Machine Codes

Glass_cutter	NOK
Glasscutter	NOK
0241	OK
Sample_Modbus	NOK
MACH001	OK
SampleModbus	NOK

3.1.3 Walidacja

Machine Code jest automatycznie walidowany przy tworzeniu/aktualizacji maszyny. Jeśli nie zostanie podany, system użyje nazwy maszyny jako fallback (jeśli spełnia wymagania formatu).

3.2 Konwencje nazewnictwa

3.2.1 Nazwy maszyn

- Zalecane: Opisowe nazwy bez znaków specjalnych
- Maksymalna długość: Brak ścisłego limitu, ale zalecane < 100 znaków
- Unikalność: Nazwy maszyn powinny być unikalne w obrębie konfiguracji

3.2.2 Nazwy sygnałów

- Zalecane: Opisowe nazwy reprezentujące zmierzoną wartość
- Ograniczenia: Unikaj znaków specjalnych, które mogą powodować problemy w systemach docelowych
- Przykłady: Temperature, Pressure, Flow_Rate, Status

3.3 Interwały odczytu

3.3.1 Format

Interwały są określane w formacie string z jednostką czasu:

- "1s" - 1 sekunda
- "10s" - 10 sekund
- "1m" - 1 minuta
- "5m" - 5 minut
- "1h" - 1 godzina

3.3.2 Wpływ na wydajność

- Krótsze interwały (1s-10s): Większa dokładność, większe obciążenie sieci i systemu
- Dłuższe interwały (1m+): Mniejsze obciążenie, mniejsza dokładność czasowa

Zalecenia:

- OPC UA z subscription mode: 1s - 10s
- Modbus TCP: 1s - 30s (w zależności od liczby rejestrów)
- S7comm: 1s - 30s

4. Struktura danych i formaty

4.1 Machine Configuration

4.1.1 Wymagane pola

```
{
  "id": "uuid",
  "protocol": "opcua" | "modbus" | "s7comm",
  "name": "string",
  "connection": {
    // Protokół-specyficzne parametry
  },
  "defaultInterval": "string",
  "signals": [],
  "tags": {
    "machine_code": "string"  // WYMAGANY
  }
```

```
}
```

4.1.2 Opcjonalne pola

```
{
  "useSubscription": true,  // Tylko dla OPC UA
  "healthCheck": {
    "enabled": true,
    "interval": "30s",
    "failureThreshold": 3,
    "notifyOnRecovery": true,
    "allowToFail": false
  }
}
```

4.2 Signal Configuration

4.2.1 Pola wspólne

```
{
  "id": "uuid",
  "name": "string",
  "address": "string",  // Opcjonalne
  "filters": ["filter_id_1", "filter_id_2"],
  "protocolSpecific": {
    // Protokół-specyficzne parametry
  }
}
```

4.2.2 OPC UA - protocolSpecific

```
{
  "registerType": "holding",
  "address": 0,
  "dataType": "INT16",
  "byteOrder": "AB",
  "scale": 1.0
}
```

4.2.3 Modbus - protocolSpecific

```
{
  "registerType": "holding",
  "address": 0,
  "dataType": "INT16",
  "byteOrder": "AB",
  "scale": 1.0
}
```

4.2.4 S7comm - protocolSpecific

Nowy format:

```
{
  "address": "DB7.I0"
}
```

Stary format:

```
{
  "db_number": 7,
  "start": 0,
  "data_type": "INT"
}
```

5. API Endpoints

5.1 Authentication

5.1.1 Login

Endpoint: POST /api/auth/login
Request:

```
{
  "email": "user@example.com",
  "password": "password"
}
```

Response:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "uuid",
    "email": "user@example.com"
  }
}
```

5.1.2 API Token (opcjonalny)

Jeśli serwer jest skonfigurowany z PUBLIC_API_TOKEN, można użyć:
Header: X-API-Token: <token>
lub
Header: Authorization: Bearer <token>

5.2 Configuration Management

5.2.1 Lista konfiguracji

Endpoint: GET /api/configurator/config
Headers:

Authorization: Bearer <jwt_token>

Response:

```
[
  {
    "id": "uuid",
    "name": "Configuration Name",
    "machines": [],
    "filters": [],
    "outputs": [],
    "isDraft": true,
    "lastModifiedAt": "2024-01-01T00:00:00Z"
  }
]
```

5.2.2 Pobierz konfigurację

Endpoint: GET /api/configurator/config/:id
Response: Pełna konfiguracja z wszystkimi maszynami, sygnałami, filtrami i wyjściami.

5.2.3 Utwórz konfigurację

Endpoint: POST /api/configurator/config
Request:

```
{
  "name": "New Configuration",
  "machines": [],
  "filters": [],
  "outputs": [],
  "agentSettings": {
    "interval": "1s"
  }
}
```

5.2.4 Aktualizuj konfigurację

Endpoint: PUT /api/configurator/config/:id
Request: Częściowa aktualizacja (tylko zmienione pola).

5.3 Machine Management

5.3.1 Dodaj maszynę

Endpoint: POST /api/configurator/config/:configId/machines
Request:

```
{
  "name": "Machine Name",
  "protocol": "opcua",
  "connection": {
    "endpoint": "opc.tcp://192.168.1.100:4840"
  },
  "defaultInterval": "10s",
  "tags": {
    "machine_code": "MACH001"
  },
  "useSubscription": true
}
```

5.3.2 Aktualizuj maszynę

Endpoint: PUT /api/configurator/config/:configId/machines/:machineId

Request: Częściowa aktualizacja maszyny.

5.3.3 Usuń maszynę

Endpoint: DELETE /api/configurator/config/:configId/machines/:machineId

5.4 Signal Management

5.4.1 Dodaj sygnał

Endpoint: POST /api/configurator/config/:configId/machines/:machineId/signals
Request (OPC UA):

```
{
  "name": "Temperature",
  "protocolSpecific": {
    "namespace": "3",
    "identifier_type": "s",
    "identifier": "\"DB500\".\"Temperature\""
  },
  "filters": []
}
```

Request (Modbus):

```
{
  "name": "Pressure",
  "protocolSpecific": {
    "registerType": "holding",
    "address": 0,
    "dataType": "INT16",
    "byteOrder": "AB",
    "scale": 0.1
  },
  "filters": []
}
```

Request (S7comm):

```
{
  "name": "Status",
  "protocolSpecific": {
    "address": "DB7.I0"
  },
  "filters": []
```

```
}
```

5.4.2 Aktualizuj sygnał

Endpoint: PUT /api/configurator/config/:configId/machines/:machineId/signals/:signalId

5.4.3 Usuń sygnał

Endpoint: DELETE /api/configurator/config/:configId/machines/:machineId/signals/:signalId

5.5 Validation i Deployment

5.5.1 Waliduj konfigurację

Endpoint: POST /api/configurator/config/:id/validate
Response:

```
{
  "valid": true,
  "errors": [],
  "warnings": []
}
```

5.5.2 Generuj TOML

Endpoint: GET /api/configurator/config/:id/generate
Response: Tekst TOML gotowy do użycia w Telegraf.

5.5.3 Wdróż wersję

Endpoint: POST /api/configurator/config/:id/deploy/:versionId
Wdraża określoną wersję konfiguracji do produkcji.

6. Przykłady konfiguracji

Zobacz pliki w katalogu docs/examples/:

- opcua_subscription_example.json
- modbus_example.json
- s7comm_example.json
- health_check_example.json

7. Health Check i Monitoring

7.1 Health Check Settings

Health Check umożliwia monitorowanie stanu połączenia z maszyną.

7.1.1 Konfiguracja

```
{
  "healthCheck": {
    "enabled": true,
    "interval": "30s",
    "failureThreshold": 3,
    "notifyOnRecovery": true,
    "allowToFail": false
  }
}
```

7.1.2 Parametry

- enabled: Włącz/wyłącz monitoring (domyślnie: false)
- interval: Częstotliwość sprawdzania połączenia (domyślnie: "30s")
- failureThreshold: Liczba kolejnych błędów przed wysłaniem alertu (domyślnie: 3)
- notifyOnRecovery: Wysyłaj powiadomienie gdy połączenie zostanie przywrócone (domyślnie: true)
- allowToFail: Wycisz alerty dla tej maszyny - monitoring działa, ale alerty nie są wysyłane (domyślnie: false)

7.2 Connection Status

Status połączenia jest dostępny w czasie rzeczywistym przez WebSocket.
Endpoint: ws://<host>/connection-status/socket.io
Statusy:

- connected - Maszyna jest połączona
- disconnected - Maszyna jest rozłączona
- unknown - Status nieznany

7.3 Alerting

System Historian obsługuje integrację z następującymi systemami alertowania:

7.3.1 Zabbix

```
{
  "zabbix": {
    "enabled": true,
    "server": "http://zabbix-server",
    "apiUrl": "http://zabbix-server/api_jsonrpc.php",
    "username": "user",
    "password": "password",
    "hostGroup": "Historian Machines"
  }
}
```

7.3.2 Mattermost

```
{
  "mattermost": {
    "enabled": true,
    "webhookUrl": "https://mattermost.example.com/hooks/xxx",
    "channel": "#alerts"
  }
}
```

7.3.3 SMTP/Email

```
{
  "smtp": {
    "enabled": true,
    "host": "smtp.example.com",
    "port": 587,
    "secure": true,
    "username": "user@example.com",
    "password": "password",
    "from": "historian@example.com",
    "recipients": ["admin@example.com"]
  }
```

```
}
```

8. Best Practices

8.1 Wydajność

8.1.1 OPC UA - Subscription Mode

- Zawsze używaj subscription mode (useSubscription: true) zamiast polling mode
- Subscription mode przesyła dane tylko przy zmianie wartości, co znacznie zmniejsza obciążenie sieci
- Domyślny interwał subskrypcji: 1s jest optymalny dla większości zastosowań

8.1.2 Deadband dla OPC UA

- Używaj deadband dla sygnałów analogowych, które mogą mieć małe fluktuacje
- Deadband zmniejsza ilość przesyłanych danych bez utraty istotnych informacji
- Przykład: deadband: 0.5 dla temperatury oznacza, że zmiany mniejsze niż 0.5°C są ignorowane

8.1.3 Optymalizacja interwałów

- Używaj najdłuższych interwałów, które są akceptowalne dla danego zastosowania
- Dla sygnałów szybkozmiennych: 1s - 10s
- Dla sygnałów wolnozmiennych: 30s - 5m

8.2 Niezawodność

8.2.1 Startup Error Behavior

- retry (zalecane): System będzie próbował ponownie połączyć się w tle
- error: System zatrzyma się przy błędzie (używaj tylko do debugowania)
- ignore: System zignoruje błąd (niezalecane)

8.2.2 Health Check

- Zawsze włączaj health check dla maszyn krytycznych
- Ustaw failureThreshold odpowiednio do charakterystyki sieci (3-5 dla stabilnych sieci, 5-10 dla niestabilnych)
- Używaj allowToFail: true tylko dla maszyn testowych lub niekrytycznych

8.2.3 Retry Logic

System automatycznie ponawia próby połączenia zgodnie z konfiguracją startup_error_behavior: "retry".

8.3 Bezpieczeństwo

8.3.1 OPC UA Security

- Dla środowisk produkcyjnych używaj security_policy: "Basic256Sha256" i security_mode: "SignAndEncrypt"
- Wymaga skonfigurowania certyfikatów (certificate i private_key)
- Dla środowisk testowych można użyć security_policy: "None" i security_mode: "None"

8.3.2 API Token

- Używaj API token dla integracji systemowych
- Przechowuj tokeny bezpiecznie (nie w kodzie źródłowym)
- Rotuj tokeny regularnie

9. Troubleshooting

9.1 Typowe problemy

9.1.1 Błędy połączenia

Problem: Maszyna nie może się połączyć z systemem BMS.
Rozwiązanie:

1. Sprawdź dostępność sieciową (ping, telnet)
2. Zweryfikuj parametry połączenia (endpoint, port, adres IP)
3. Sprawdź konfigurację firewall
4. Dla OPC UA: sprawdź security policy i mode
5. Sprawdź logi systemowe

9.1.2 Problemy z walidacją

Problem: Konfiguracja nie przechodzi walidacji.
Rozwiązanie:

1. Sprawdź czy machine_code jest poprawny (4-8 znaków alfanumerycznych)
2. Zweryfikuj format adresów sygnałów
3. Sprawdź czy wszystkie wymagane pola są wypełnione
4. Użyj endpointu /api/configurator/config/:id/validate aby zobaczyć szczegóły błędów

9.1.3 Błędy w formatowaniu adresów

OPC UA:

- Sprawdź czy namespace jest poprawny (string z numerem)
- Zweryfikuj identifier_type (s, i, g, b)
- Dla string identifiers: upewnij się, że identyfikator jest poprawnie escapowany

Modbus:

- Sprawdź czy address jest w zakresie 0-65535

- Zweryfikuj registerType (holding, input, coil, discrete)
- Sprawdź czy dataType jest obsługiwany

S7comm:

- Dla nowego formatu: sprawdź składnię DB<number>.<type><offset>
- Dla starego formatu: zweryfikuj db_number, start, data_type

9.2 Diagnostyka

9.2.1 Logi systemowe

Logi systemu Historian zawierają szczegółowe informacje o:

- Próbach połączenia
- Błędach komunikacji
- Statusie maszyn
- Operacjach API

9.2.2 Endpointy diagnostyczne

- GET /api/healthcheck - Status systemu
- GET /api/configurator/config/:id/validate - Walidacja konfiguracji
- WebSocket connection status - Status połączeń w czasie rzeczywistym

9.2.3 WebSocket Connection Status

Połącz się z WebSocket endpoint aby monitorować status połączeń w czasie rzeczywistym:

```
const socket = io('http://localhost:3001/connection-status', {
  path: '/socket.io'
});

socket.on('connection-status', (data) => {
  console.log('Machine:', data.machineId, 'Status:', data.status);
});
```

10. Załączniki

10.1 Schematy danych

10.1.1 Machine JSON Schema

```
{
  "type": "object",
  "required": ["id", "protocol", "name", "connection", "defaultInterval",
"signals", "tags"],
```

```
  "properties": {
    "id": { "type": "string", "format": "uuid" },
    "protocol": { "type": "string", "enum": ["opcua", "modbus", "s7comm"]
},
    "name": { "type": "string" },
    "connection": { "type": "object" },
    "defaultInterval": { "type": "string", "pattern": "^\\d+[smh]$" },
    "signals": { "type": "array", "items": { "type": "object" } },
    "tags": {
      "type": "object",
      "required": ["machine_code"],
      "properties": {
        "machine_code": { "type": "string", "pattern": "^[a-zA-Z0-9]{4,8}$"
}
      }
    },
    "useSubscription": { "type": "boolean" },
    "healthCheck": { "type": "object" }
  }
}
```

10.2 Tabele referencyjne

10.2.1 Typy danych Modbus

Typ	Rozmiar (bajty)	Zakres	Byte Order
INT8L	1	-128 do 127	-
INT8H	1	-128 do 127	-
UINT8L	1	0 do 255	-
UINT8H	1	0 do 255	-
INT16	2	-32768 do 32767	AB
UINT16	2	0 do 65535	AB
INT32	4	-2147483648 do 2147483647	ABCD
UINT32	4	0 do 4294967295	ABCD

FLOAT32	4	±3.4E±38	ABCD
FLOAT64	8	±1.7E±308	ABCDEFGH

10.2.2 Security Policies OPC UA

Policy	Opis	Zalecane użycie
None	Brak szyfrowania	Tylko środowiska testowe
Basic128Rsa15	Podstawowe szyfrowanie RSA	Przestarzałe, niezalecane
Basic256	Szyfrowanie 256-bit	Środowiska produkcyjne (starsze)
Basic256Sha256	Szyfrowanie 256-bit z SHA-256	Środowiska produkcyjne (zalecane)

10.2.3 Security Modes OPC UA

Mode	Opis	Zalecane użycie
None	Brak zabezpieczeń	Tylko środowiska testowe
Sign	Podpis cyfrowy	Środowiska wewnętrzne
SignAndEncrypt	Podpis i szyfrowanie	Środowiska produkcyjne (zalecane)