pubsubplus-connector-opcua

User Guide

Solace Corporation

Version 4.0.0

solace.

Table of Contents

Preface . . . 1
Configuring Micro Integration . . . 2
  OPCUA Server Connection Details . . . 2
Configuring Multiple OPCUA NodeId’s . . . 5
  OPCUA Consumer . . . 5
    Option 1 — Inline Configuration . . . 5
    Option 2 — Centralized Node Configuration File . . . 5
  OPCUA Producer . . . 8
OPCUA Method Calls . . . 9
Message Mapping & Transforms . . . 10
Management and Monitoring Micro Integration . . . 11
Configuration . . . 12
  OPC UA Consumer Configuration Options . . . 12
  OPC UA Producer Configuration Options . . . 15
License . . . 18

Preface

OPCUA Connector

- Getting Started
- Resource Requirements for Self-Managed Micro-Integrations
- Enabling Workflows

solace.

Configuring Micro Integration

- Configuring the Datapath for Your Self-Managed Micro-Integration

OPCUA Server Connection Details

The Spring Cloud Stream Binder for OPCUA uses the following configuration to configure its session. The connector currently supports 1.03 OPCUA specifications.

Note: This connector is tested using a mock OPCUA server.

In the `application.yml`, this typically would be configured as follows:

These configuration options are prefixed by `opcua.java`

Config Option	Type	Valid Values	Default Value	Description
`endpointUrl`	`string`	`OPC UA Server url in the format opc.tcp://<host>:<port>/endpoint`	`empty`	OPCUA Server connection URL
`username`	`string`	`any`	`empty`	OPCUA Server username
`password`	`string`	`any`	`empty`	OPCUA Server password
`applicationName`	`string`	`any`	`Solace OPC UA connector`	Application Name for the connector instance
`applicationUri`	`string`	`any`	`urn:solace:opcua:connector`	Application URI for the connector instance
`sessionTimeout`	`integer`	`positive integer value`	`120000`	Requested session timeout value, in milliseconds.
`connectTimeout`	`integer`	`positive integer value`	`5000`	The timeout, in milliseconds, when opening a socket connection to a remote host.
`requestTimeout`	`integer`	`positive integer value`	`5000`	Maximum amount of time, in milliseconds, to wait for the response to a request.
`keepAliveInterval`	`integer`	`positive integer value`	`5000`	Interval, in milliseconds, between keep-alive requests.

Config Option	Type	Valid Values	Default Value	Description
keepAliveTimeout	integer	positive integer value	5000	Max duration, in milliseconds, to wait for a response to a keep-alive request.
maxResponseMessageSize	integer	positive integer value	0	The maximum allowable size of an OPC UA application layer message. 0 means no max size.
maxPendingPublishRequests	long	positive long value	4294967295L	The number of concurrent Publish Requests allowed to be pending at any given time.
userCertificate	string	absolute file path	empty	File path to user certificate.
userCertificate	string	absolute file path	empty	Absolute path to the client certificate file used for certificate-based authentication. Required when certificate authentication is enabled.
userCertificatePrivateKey	string	absolute file path	empty	Absolute path to the PKCS#8 encoded RSA private key used for OPC UA client certificate authentication. The private key must correspond to the configured client certificate.
certificate.certificatePath	string	absolute file path	empty	Absolute path to the X509 certificate. OPC UA client will present this certificate to the server during the secure channel establishment.

Config Option	Type	Valid Values	Default Value	Description
certificate.clientAlias	string	absolute file path	empty	Alias used to retrieve the client certificate and private key from the keystore. The certificate and key pair identified by this alias are used by the OPC UA client to establish a secure session with the server.
certificate.password	string	absolute file path	empty	Password used to retrieve the client certificate and private key from the keystore. The certificate and key pair identified by this password are used by the OPC UA client to establish a secure session with the server.
certificate.certificateInstance	string	Certificate instance type	PKCS12	This property cannot be overridden, and it determines the certificate instance type.
certificate.securityPolicy	string	None | Basic128Rsa15 | Basic256 | Basic256Sha256 | Aes128_Sha256_RsaOaep | Aes256_Sha256_RsaPss	None	The type of security policy to be configured for secure authentication. Please refer to OPCUA specifications for more details on security policies.
certificate.validateCertificates	boolean	true or false	true	Enable or disable certificate validation.
certificate.useDefaultTrustStore	boolean	true or false	true	By default, the OPCUA client will load the default trust store during secure authentication.
certificate.useDefaultTrustStore	customTrustStorePath	absolute file path	empty	Absolute file path to custom trust store. This property is valid if certificate.useDefaultTrustStore is set to false

Configuring Multiple OPCUA NodeId's

OPCUA Consumer

Multiple OPC UA NodeIds can be configured per workflow when consuming data from an OPC UA server. NodeIds can be specified either inline in the binding configuration or via a centralized configuration file.

The MI creates one subscription per workflow to monitor the configured nodes. Ensure that the total number of monitored items does not exceed the OPC UA server's per subscription maximum limit.

This capability is still evolving and may impact performance or resource utilization under heavy load. It is recommended to thoroughly test it before deploying in a production environment and to determine the optimal number of OPCUA nodes per workflow for your specific use case.

Option 1 — Inline Configuration

```
spring:
  cloud:
    stream:
      bindings:  # Workflow bindings
        input-1:
          destination: ns=2;s=Demo.Dynamic.String,ns=2;s=Demo.Dynamic.Double
          binder: opcua
        output-1:
          destination: foo/bar  # OPCUA producer destination
          binder: solace
```

Option 2 — Centralized Node Configuration File

For workflows with a large number of nodes, NodeIds can be managed in a centralized configuration file. This avoids long inline destination strings and makes node management easier across environments. Supported formats: CSV, TXT

CSV Format

Headers are mandatory when using CSV format. The column containing the NodeIds must be specified via the node-column-identifier property, either by header name or by column index position.

```
S.No,NodeIds
1,ns=2;s=Demo.Dynamic.Boolean
2,ns=2;s=Demo.Dynamic.Byte
3,ns=2;s=Demo.Dynamic.DataValue
```

```
4,ns=2;s=Demo.Dynamic.DateTime
5,ns=2;s=Demo.Dynamic.ByteString
6,ns=2;s=Demo.Dynamic.Double
7,ns=2;s=Demo.Dynamic.Float
8,ns=2;s=Demo.Dynamic.Guid
9,ns=2;s=Demo.Dynamic.Int16
10,ns=2;s=Demo.Dynamic.Int32
```

```
spring:
  cloud:
    stream:
      opcua:
        default: # applies to all OPCUA input bindings. Each binding will read this
file and extract nodeids based on specified range.
          consumer:
            node-file-path: ./nodelist.csv
            node-file-format: CSV
            node-column-selector: BY_HEADER # read column by header name
            node-column-identifier: NodeIds # matches the 'NodeIds' column header in
CSV
      bindings:
        input-0:
          destination: 0-5
          binder: opcua
        output-0:
          destination: test/topic
          binder: solace
        input-1:
          destination: 5-10
          binder: opcua
        output-1:
          destination: test/topic
          binder: solace
```

For index-based access:

```
node-column-selector: BY_INDEX
node-column-identifier: 1 # 0-based index — column 1 = 'NodeIds'
```

node-column-selector defaults to BY_HEADER. When using BY_INDEX, the value of node-column-identifier must be a valid 0-based integer index corresponding to the column position in the CSV file.

TXT Format

For plain text files, each NodeId must be on a separate line. No headers or delimiters are required.

```
ns=2;s=Demo.Dynamic.Boolean
ns=2;s=Demo.Dynamic.Byte
ns=2;s=Demo.Dynamic.DataValue
ns=2;s=Demo.Dynamic.DateTime
ns=2;s=Demo.Dynamic.ByteString
ns=2;s=Demo.Dynamic.Double
ns=2;s=Demo.Dynamic.Float
ns=2;s=Demo.Dynamic.Guid
ns=2;s=Demo.Dynamic.Int16
ns=2;s=Demo.Dynamic.Int32
```

```
spring:
  cloud:
    stream:
      opcua:
        default: # applies to all OPCUA input bindings. Each binding will read this
file and extract nodeids based on specified range.
          consumer:
            node-file-path: ./nodelist.txt
      bindings:
        input-0:
          destination: 0-5
          binder: opcua
        output-0:
          destination: test/topic
          binder: solace
        input-1:
          destination: 5-10
          binder: opcua
        output-1:
          destination: test/topic
          binder: solace
```

Configuring a Node File Per Binding

Each OPC UA input binding can be independently configured with its own node configuration file, allowing different workflows to consume NodeIds from separate files. This is useful when managing large node lists across multiple workflows or when different workflows require nodes from different sources.

```
spring:
  cloud:
    stream:
      opcua:
        bindings:
          input-0:
            consumer:
              node-file-path: ./nodelist-0.csv
```

```
              node-file-format: CSV
              node-column-selector: BY_HEADER # read column by header name
              node-column-identifier: NodeIds # matches the 'NodeIds' column header in
CSV
          input-1:
            consumer:
              node-file-path: ./nodelist-1.txt
      bindings:
        input-0:
          destination: 0-5 # read NodeIds from line 0 to line 5 in nodelist-0.csv
          binder: opcua
        output-0:
          destination: test/topic
          binder: solace
        input-1:
          destination: 5-10 # read NodeIds from line 5 to line 10 in nodelist-1.txt
          binder: opcua
        output-1:
          destination: test/topic
          binder: solace
```

OPCUA Producer

When publishing to OPC UA, a dynamic destination can be configured so that each message is sent to its corresponding NodeId. The configuration structure is shown below. For more information on configuring header transforms and dynamically resolving values from source headers, refer to Message Transforms section for more details.

```
solace:
  connector:
    workflows:  # Workflow configuration
      0:
        enabled: true  # If true, the workflow is enabled.
        acknowledgment:
          publish-async: true
        transform:
          expressions: # An ordered list of transformation expressions to apply to the
message
            - transform: "target['headers']['scst_targetDestination'] =
#joinString('/', 'ns=2;s=CTT.Static.AllProfiles.Scalar.String')"
```

OPCUA Method Calls

Support for OPC UA methods is an evolving capability. Users should ensure they understand the expected behavior and validate outcomes when configuring and using this feature. Currently, method invocation is supported only through event-driven triggers (e.g., Solace to OPC UA flow). Refer to Producer Configuration for supported options.

Message Mapping & Transforms

Mapping Message Headers and Payloads

Management and Monitoring Micro Integration

Managing and Deploying Your Self-Managed Micro-Integration

Configuration

Configuring Self-Managed Micro-Integrations

Workflow Configuration

OPC UA Consumer Configuration Options

These configuration options are all prefixed by `spring.cloud.stream.opcua.bindings.<binding-channel-name>.consumer.`:

Config Option	Type	Valid Values	Default Value	Description
`publishingInterval`	`double`	`positive double value`	`1000.0`	Interval at which notification messages are sent by subscription.
`samplingInterval`	`double`	`positive double value`	`1000.0`	The interval that defines the fastest rate at which the MonitoredItem(s) should be accessed and evaluated. This interval is defined in milliseconds.
`queueSize`	`int`	`positive integer value`	`10`	The requested size of the MonitoredItem queue
`clientHandle`	`int`	`positive integer value`	`0`	Client-supplied id of the MonitoredItem. This id is used in Notifications generated for the list Node. By default, the connector generates unique client handle.

Config Option	Type	Valid Values	Default Value	Description
discardOldest	boolean	true or false	true	A boolean parameter that specifies the discard policy when the queue is full and a new Notification is to be queued. It has the following values: TRUE - the oldest (first) Notification in the queue is discarded. The new Notification is added to the end of the queue. FALSE - the last Notification added to the queue gets replaced with the new Notification.
enablePublishingMode	boolean	true or false	true	A Boolean parameter with the following values: TRUE - publishing of NotificationMessages is enabled for the Subscription. FALSE - publishing of NotificationMessages is disabled for the Subscription.
subscriptionType	string	DATA_SUBSCRIPTION | EVENT_SUBSCRIPTION | MANAGED_DATA_SUBSCRIPTION | MANAGED_EVENT_SUBSCRIPTION	DATA_SUBSCRIPTION	Choose the type of subscription Data or Event subscription. In case of Managed subscription above configuration will not be considered since they operate with inbuilt default values. In case if more control is required use regular subscriptions.

Config Option	Type	Valid Values	Default Value	Description
nodeFilePath	string	valid file system path	empty	Specifies the absolute path to the centralized node configuration file containing the list of OPC UA NodeIds to monitor. When configured, the binding destination should specify the line range of NodeIds to read from the file using a 0-based index range in the format startIndex-endIndex.
nodeFileFormat	string	TEXT, CSV	TEXT	Specifies the format of the node configuration file defined in node-file-path. Set to TEXT for plain text files where each NodeId is on a separate line. Set to CSV for comma-separated files where NodeIds are stored in a specific column alongside other metadata.
nodeColumnSelector	string	BY_HEADER, BY_INDEX	BY_HEADER	Required when nodeFileFormat is set to CSV. Specifies how the connector identifies the column containing NodeIds in the CSV file. When set to BY_HEADER, the column is identified by its header name as specified in node-column-identifier. When set to BY_INDEX, the column is identified by its 0-based integer index position as specified in node-column-identifier.

Config Option	Type	Valid Values	Default Value	Description
nodeColumnIdentifier	string	valid column identifier(header name or index position)	empty	Required when nodeFileFormat is set to CSV. Specifies the identifier used to locate the column containing NodeIds in the CSV file. The interpretation of this value depends on the node-column-selector setting. When node-column-selector is set to BY_HEADER, this value must match the column header name in the CSV file. When set to BY_INDEX, this value must be a valid 0-based integer index corresponding to the column position. IMPORTANT: When node-column-selector is BY_HEADER, the value of node-column-identifier is case-sensitive and must match the column header in the CSV file exactly. For example, NodeIds and nodeids are treated as different values. Ensure the configured value matches the header in the CSV file precisely.

OPC UA Producer Configuration Options

These configuration options are all prefixed by `spring.cloud.stream.opcua.bindings.<binding-channel-name>.producer.`:

Config Option	Type	Valid Values	Default Value	Description
destinationType	DestinationType	METHOD, NODE	NODE	Type of destination when streaming data to OPCUA. If set to METHOD, the destination name in spring.cloud.stream.bindings.<binding-channel-name>.destination should be configured as <opcua-parentnode-address>,<method-name>. Example: ns=2;s=CTT.Methods,MethodIO
inputParams	List	list of string values	empty	Set this property when the destination type is METHOD and the method requires input parameters. The MI determines the input data type by fetching the type definition configured on the method node. It then casts the input parameters before invoking the method. Input values can be configured using message headers, payload, and/or custom static values. These sources can be combined, and their order must match the order of the method arguments defined on the OPC UA method. See detailed examples below.

inputParams Examples

Headers must be explicitly mapped using the transformation feature; otherwise, they will resolve to null. Refer to Headers Propagation section for more details. .

```
input-params:
  - headers.<header-name-1>
  - headers.<header-name-2>
```

Payload Example - Since OPC UA methods support standard data types, the MI casts the entire payload based on the method type definition.Even if the method defines multiple input arguments, the payload is treated as a single input argument. This approach is valid only when the method invocation expects a single effective input derived from the payload.

```
input-params:
  - payload
```

Custom Static values can also be provided directly as input parameters.

```
input-params:
  - 10
  - 20
  - 30
```

License

This project is licensed under the Solace Community License, Version 1.0. - See the `LICENSE` file for details.