pubsubplus-connector-file-events

User Guide

Solace Corporation

Version 3.1.0

solace.

Table of Contents

Preface

File Mesh, a purpose-built solution that bridges the gap between legacy file systems and modern event-driven architectures. Powered by Solace, File Mesh enables files to seamlessly participate in the event mesh ecosystem, ensuring businesses can modernize their workflows without losing the value locked in file-based processes.

With support for multiple protocols such as SFTP, FTP, FTPs, Google Cloud Storage, and S3, File Mesh provides unparalleled flexibility, enabling organizations to leverage various storage platforms as part of their event-driven architecture.

`The File Events Micro-Integration enables us to stream events from files seamlessly to Solace Event Mesh.` There are two types of File Events Micro-Integrations

`File Events Source` or `File-to-Events`: File Events can process files in formats such as CSV, delimited, JSON, and XML. It reads individual records from these files and publishes events dynamically across the event mesh on dynamic topics. This allows businesses to extract valuable data from file-based workflows and inject it into real-time streams, triggering immediate actions or processes based on that data.

`File Events Sink` or `` `Events-to-File ``: Events can be consumed from a Solace Queue, and based on dynamic, configurable settings, these events can be written to a file. This is ideal for organizations that need to store events in a persistent manner, ensuring that event data can be easily accessed or processed later as part of a file-based workflow.

Getting Started

Assuming you're using the default `application.yml` within this package, following one of the below quick start guides will result in a connector that will connect to the PubSub+ broker and File To Events using default credentials, with 2 workflows enabled, workflow 0 and workflow 1. Where:

- Workflow 0 is consuming messages from the Solace PubSub+ queue, `Solace/Queue/0`, and publishing them to the File To Events producer destination, `producer-destination`.
- Workflow 1 is consuming messages from the File To Events consumer destination, `consumer-destination`, and publishing them to the Solace PubSub+ topic, `Solace/Topic/1`.

A workflow is the configuration of a flow of messages from a source to a target. The connector supports up to 20 concurrent workflows per instance.

The connector will not provision queues which do not exist.

Prerequisites

- Solace PubSub+ Event Broker
- File To Events

Quick Start common steps

These are the steps that are required to run all quick-start examples:

1. Update the provided `samples/config/application.yml` with the values for your deployment.

Quick Start: Running the connector via command line

Run:

```
java -jar pubsubplus-connector-file-events-3.1.0.jar --spring.config.additional-location=file:samples/config/
```

By default, this command detects any Spring Boot configuration files as per the Spring Boot's default locations.

For more information, see Configure Locations to Find Spring Property Files.

Quick Start: Running the connector via `start.sh` script

For convenience, you can start the connector through the shell script using the following command:

```
chmod 744 ./bin/start.sh
```

```
./bin/start.sh [-n NAME] [-l FOLDER] [-p PROFILE] [-c FOLDER] [-ch HOST] [-cp PORT] [
-j FILE] [-cm] [-cmh HOST] [-cmp PORT] [-mh HOST] [-mp PORT] [-o OPTIONS] [-b]
```

The script shows you all errors at the same time:

```
./bin/start.sh -l dummy_folder -c dummy_folder -j dummy_file.jar
```

The script shows you all errors at the same time:

```
pubsubplus-connector-file-events

Connector startup failed:

Following folder doesn't exists on your filesystem:     'dummy_folder'
Following folder doesn't exists on your filesystem:     'dummy_folder'
Following file doesn't exists on your filesystem:       'dummy_file.jar'
```

In situations where you don't provide a parameter, the script runs with the predefined values as follows:

Parameter	Default Value	Description
`-n, --name`	`application`	The name of the connector instance, that is configured in [spring.application.name]. This name impacts on grouping connectors only.
`-l, --libs`	`./libs`	The directory that contains the required and optional dependency JAR files, such as Micrometer metrics export dependencies (if configured). If this option is not specified, it will use the current `./libs/` directory.
`-p, --profile`	`empty, no profile is used`	The profile to be used with the connector's configuration. The configuration file named 'application-<profile>.yml' is used. If this option is not specified, no profile is used.

Parameter	Default Value	Description
`-c, --config`	`./ or current folder`	The path to the folder containing the configuration files to be applied when the connector starts up the chosen profile. If not specified, the current directory is used.
`-H, --host`	`127.0.0.1`	Specifies the host where the connector runs.
`-P, --port`	`8090`	Specifies the port where connector runs.
`-mp, --mgmt_port`	`9009`	Specifies the management port for back calls of current connector from PubSub+ Connector Manager. This parameter is ignored if the `-cm` parameter is not provided.
`-j, --jar`	`pubsubplus-connector-file-events-3.1.0.jar`	The path to the specified JAR file to start the connector. If the option is not specified, the default JAR file is used from the current directory.
`-cm, --manager`	`application`	Specifies PubSub+ Connector Manager to use the configuration storage and allows you to enable the cloud configuration for the connector. When this parameter is enabled, you can specify the `-mp` or `--mgmt_port`, `-H` or `--host`, and `-cmh` with the `-cmp` parameters, unless you want to use default values for those parameters. Be aware, this option disable listed parameters to be read from configuration file. In this case, the operator must explicitly specify the parameters for the script, otherwise defaultdefault values are used.
`-cmh, --cm_host`	`127.0.0.1`	Specifies the host where Connector Manager is running. This parameter is ignored if the `-cm` parameter is not provided.

Parameter	Default Value	Description
`-cmp, --cm_port`	`9500`	Specifies the port where Connector Manager is running. This parameter is ignored if `-cm` parameter is not provided.
`-o, --options`	`no default values`	Specifies the JVM options used on when the connector starts. For example, `-Xms64M -Xmx1G`.
`-tls`	`N/A`	Specifies to use HTTPS instead of HTTP. . When this parameter is used, the configuration file must contain an additional section with the preconfigured paths for the key store and trust store files.
`-s, --show`	`N/A`	Performs a dry run (does nothing). The output prints the start CLI command and its raw output and exits. This parameter is useful to check your parameters without running the connector.
`-b, --background`	`N/A`	Runs the connector in the background. No logs are shown and the connector continues running in detached mode.
`-h, --help`	`N/A`	Prints the help information and exits.

Script also provides that help information from command line using parameter `-h`.

More configuration example of starting Connector together with Connector Manager are provided by the Connector Manager samples.

Quick Start: Running the connector as a Container

See the connector's container documentation for instructions on how to run this connector as a container.

Enabling Workflows

The provided `application.yml` enables workflow 0 and 1. To enable additional workflows, define the following properties in the `application.yml`, where `<workflow-id>` is a value between `[0-19]`:

```
spring:
  cloud:
    stream:
      bindings: # Workflow bindings
        input-<workflow-id>:
          destination: <input-destination> # Queue name
          binder: (solace|file-events) # Input system
        output-<workflow-id>:
          destination: <output-destination> # Topic name
          binder: (solace|file-events) # Output system

solace:
  connector:
    workflows:
      <workflow-id>:
        enabled: true
```

The connector only supports workflows in the directions of:

- `solace` → `File To Events`
- `File To Events` → `solace`

For more information about Spring Cloud Stream and the Solace PubSub+ binder, see:

- Spring Cloud Stream Reference Guide
- Spring Cloud Stream Binder for Solace PubSub+

Configuring Connection Details

Solace PubSub+ Connection Details

The Spring Cloud Stream Binder for PubSub+ uses Spring Boot Auto-Configuration for the Solace Java API to configure its session.

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

```
solace:
  java:
    host: tcp://localhost:55555
    msg-vpn: default
    client-username: default
    client-password: default
```

For more information and options to configure the PubSub+ session, see Spring Boot Auto-Configuration for the Solace Java API.

Preventing Message Loss when Publishing to Topic-to-Queue Mappings

If the connector is publishing to a topic that is subscribed to by a queue, messages may be lost if they are rejected. For example, if queue ingress is shutdown.

To prevent message loss, configure `reject-msg-to-sender-on-discard` with the `including-when-shutdown` flag.

Connecting to Multiple Systems

To connect to multiple systems of a same type, use the multiple binder syntax.

For example:

```
spring:
  cloud:
    stream:
      binders:

        # 1st solace binder in this example
        solace1:
          type: solace
          environment:
            solace:
              java:
                host: tcp://localhost:55555

        # 2nd solace binder in this example
```

```
        solace2:
          type: solace
          environment:
            solace:
              java:
                host: tcp://other-host:55555

        # The only file-events binder
        file-events1:
          type: file-events
          # Add `environment` property map here if you need to customize this binder.
          # But for this example, we'll assume that defaults are used.

        # Required for internal use
        undefined:
          type: undefined
      bindings:
        input-0:
          destination: <input-destination>
          binder: file-events1
        output-0:
          destination: <output-destination>
          binder: solace1 # Reference 1st solace binder
        input-1:
          destination: <input-destination>
          binder: file-events1
        output-1:
          destination: <output-destination>
          binder: solace2 # Reference 2nd solace binder
```

The configuration above defines two binders of type `solace` and one binder of type `file-events`, which are then referenced within bindings.

Each binder above is configured independently under `spring.cloud.stream.binders.<binder-name>.environment`.

- When connecting to multiple systems, all binder configuration must be specified using the multiple binder syntax for all binders. For example, under the `spring.cloud.stream.binders.<binder-name>.environment`.
- Do not use single-binder configuration (for example, `solace.java.*` at the root of your `application.yml`) while using the multiple binder syntax.

File Source Connection Details

The Spring Cloud Stream Binder for File uses the following configuration to configure Source Connector

```
spring:
```

```
  cloud:
    stream:
      bindings:
        input-0:
          destination: # Absolute file path of source and sink file/directory. In
general for Static, Directory replication we need to configure source and sink paths
in a separate config file since multiple files and directories are supported, provide
the absolute location of config file(Create a file with extension .cfg and add your
path as per format at the end. Each line represents a source and sink path). In case
of dynamic file we can configure the source and sink path without the need for
separate config file since only one dynamic file is supported for replication.
          # Format to configure file location(absolute-source-file-path|absolute-sink-
filepath). This format applies for Static, Directory and Dynamic files.
          binder: file
        output-0:
          destination: # configure solace topic - File events are published to this
topic. This topic should be added as subscription to Sink Connector queue
          binder: solace
```

File Sink Connection Details

The Spring Cloud Stream Binder for File uses the following configuration to configure Sink Connector

```
spring:
  cloud:
    stream:
      bindings:
        output-0:
          destination: # Absolute base destination path of Sink file or directory.
This value is used when dest_file_name_type property is set to 1, 4 or 5
          binder: file
        input-0:
          destination: # configure sink connector queue where file events sent by
source connector are spooled
          binder: solace
```

User-configured Header Transforms

Deprecated

This feature is deprecated and will be removed in a future release. Use Message Transforms instead.

Generic Migration Rules:

1. Configuration Structure Changes:
   a. Replace `solace.connector.workflows.<n>.transform-headers` with `solace.connector.workflows.<n>.transform` and within it:
      i. Add `enabled: true` to enable transforms
      ii. Move `transform-headers.expressions` to the `transform.expressions` list property
2. Expression Syntax Changes:
   - Replace direct header references, `headers.<headerName>`, with `source['headers']['<headerName>']` for reading
   - Use `target['headers']['<headerName>'] = ⋯` for writing
   - Replace Java methods (`T(String)`, etc.) with built-in functions:
     - `T(String).join()` → `#joinString()`
     - `split()` → `#splitString()`
     - `toUpperCase()` → `#upperCaseString()`
     - `toLowerCase()` → `#lowerCaseString()`
     - etc

Example Migration:

Old Configuration:

```
solace:
  connector:
    workflows:
      0:
        transform-headers:
          expressions:
            route: "T(String).format('%s/%s', headers.region,
headers.status)"
            count: "headers.count.toString()"
```

New Configuration:

```
solace:
```

```
  connector:
    workflows:
      0:
        transform:
          enabled: true
          expressions:
            - transform: "target['headers']['route'] = #joinString('/',
source['headers']['region'], source['headers']['status'])"
            - transform: "target['headers']['count'] =
#convertNumberToString(source['headers']['count'])"
```

Generally, the consumed message's headers are propagated through the connector to the output message. If you want to transform the headers, then you can do so as follows:

```
# <workflow-id> : The workflow ID ([0-19])
# <header> : The key for the outbound header
# <expression> : A SpEL expression which has "headers" as parameters

solace.connector.workflows.<workflow-id>.transform-
headers.expressions.<header>=<expression>
```

Example 1: To create a new header, `new_header`, for workflow `0` that is derived from the headers `foo` & `bar`:

```
solace.connector.workflows.0.transform-headers.expressions.new_header
="T(String).format('%s/abc/%s', headers.foo, headers.bar)"
```

Example 2: To remove the header, `delete_me`, for workflow `0`, set the header transform expression to `null`:

```
solace.connector.workflows.0.transform-headers.expressions.delete_me="null"
```

For more information about Spring Expression Language (SpEL) expressions, see Spring Expression Language (SpEL).

User-configured Payload Transforms

Deprecated

This feature is deprecated and will be removed in a future release. Use Message Transforms instead.

Message payloads going through a workflow can be transformed using a Spring Expression Language (SpEL) expression as follows:

```
# <workflow-id> : The workflow ID ([0-19])
# <expression> : A SpEL expression

solace.connector.workflows.<workflow-id>.transform-payloads.expressions[0].transform
=<expression>
```

A SpEL expression may reference:

- `payload`: To access the message payload.
- `headers.<header_name>`: To access a message header value.
- Registered functions.

While the syntax uses an array of expressions, only a single transform expression is supported in this release. Multiple transform expressions may be supported in the future.

Registered Functions

Registered functions are built-in and can be called directly from SpEL expressions. To call a registered function, use the `#` character followed by the function name. The following table describes the available registered functions:

Registered Function Signature	Description
`boolean isPayloadBytes(Object obj)`	Returns whether the object `obj` is an instance of `byte[]` or not. Sample usage of this function within a SpEL expression: `"#isPayloadBytes(payload) ? true : false"`

Example 1: To normalize `byte[]` and `String` payloads as upper-cased `String` payloads or leave payloads unchanged when of different types:

```
solace.connector.workflows.0.transform-payloads.expressions[0].transform
="#isPayloadBytes(payload) ? new String(payload).toUpperCase() : payload instanceof
```

```
T(String) ? payload.toUpperCase() : payload"
```

Example 2: To convert `String` payloads to `byte[]` payloads using a `charset` retrieved from a message header or leave payloads unchanged when of different types:

```
solace.connector.workflows.0.transform-payloads.expressions[0].transform="payload
instanceof T(String) ?
payload.getBytes(T(java.nio.charset.Charset).forName(headers.charset)) : payload"
```

For more information about Spring Expression Language (SpEL) expressions, see Spring Expression Language (SpEL).

Message Headers

Solace and file-events headers can be created or manipulated using the User-configured Header Transforms feature described above.

Solace Headers

Solace headers exposed to the connector are documented in the Spring Cloud Stream Binder for Solace PubSub+ documentation.

Reserved Message Headers

The following are reserved header spaces:

- `solace_`
- `scst_`
- Any headers defined by the core Spring messaging framework. See Spring Integration: Message Headers for more info.

Any headers with these prefixes (that are not defined by the connector or any technology used by the connector) may not be backwards compatible in future releases of this connector.

Dynamic Producer Destinations

To route messages to dynamic destinations at runtime, use the Message Transform feature to set the following headers:

Header Name	Type	Values	Applies To	Description
`scst_targetDestination`	`string`	Any valid destination name	Solace, File To Events	Specifies the name of the dynamic destination to publish to. Setting this header overrides the configured destination.
`solace_scst_targetDestinationType`	`string`	`(queue|topic)`	Solace	Specifies the destination type of the dynamic destination. When unspecified, the configured or default destination type is used.

Setting the `scst_targetDestination` header under `solace.connector.default.workflow.*` may not be viable if not all workflows follow the same direction.

Asynchronous Publishing

This connector does not support asynchronous publishing. Publish acknowledgments are resolved synchronously for all workflows regardless of the config option:

```
# <workflow-id> : The workflow ID ([0-19])

solace.connector.workflows.<workflow-id>.acknowledgment.publish-async=(true|false)
```

Enabling `publish-async` enable asynchronous publishing on the connector's core, but the effective publishing mode is still synchronous because there is no support for this feature on either the consumer binding or the producer binding.

Management and Monitoring Connector

Monitoring Connector's States

The connector provides an ability to monitor its internal states through exposed endpoints provided by Spring Boot Actuator.

An Actuator shares information through the endpoints reachable over HTTP/HTTPS. The endpoints that are available are configured in the connector configuration file.

What endpoints are available is configured in the connector configuration file:

```
management:
  simple:
    metrics:
      export:
        enabled: true
  endpoints:
    web:
      exposure:
        include:
"health,metrics,loggers,logfile,channels,env,workflows,leaderelection,bindings,info"
```

The above sample configuration enables metrics collection through the configuration parameter of `management.simple.metrics.export.enabled` set to `true` and then shares them through the HTTP/HTTPS endpoint together with other sections configured for the current connector.

Exposed HTTP/HTTPS Endpoints

The set of endpoints exposed through the HTTP/HTTPS endpoint.

- Exposed endpoints are available if you query the endpoints using the web interface (for example `https://localhost:8090/actuator/<some_endpoint>`) and also available in PubSub+ Connector Manager.
- The operator may choose to not expose all or some of these endpoints. If so, the Actuator endpoints that are not exposed are not visible if you query the endpoints (for example, `https://localhost:8090/actuator/<some_endpoint>`) nor in PubSub+ Connector Manager.

The simple metrics registry is only to be used for testing. It is not a production-ready means of collecting metrics. In production, use a dedicated monitoring system (for example, Datadog, Prometheus, etc.) to collect metrics.

The Actuator endpoint now contains information about Connector's internal states shared over the following HTTP/HTTPS endpoint:

```
GET: /actuator/
```

The following shows an example of the data shared with the configuration above:


```
{
  "_links": {
    "self": {
      "href": "/actuator",
      "templated": false
    },
    "workflows": {
      "href": "/actuator/workflows",
      "templated": false
    },
    "workflows-workflowId": {
      "href": "/actuator/workflows/{workflowId}",
      "templated": true
    },
    "leaderelection": {
      "href": "/actuator/leaderelection",
      "templated": false
    },
    "health-path": {
      "href": "/actuator/health/{*path}",
      "templated": true
    },
    "health": {
      "href": "/actuator/health",
      "templated": false
    },
    "metrics": {
      "href": "/actuator/metrics",
      "templated": false
    },
    "metrics-requiredMetricName": {
      "href": "/actuator/metrics/{requiredMetricName}",
      "templated": true
    }
  }
}
```

Health

The connector reports its health status using the Spring Boot Actuator health endpoint.

To configure the information returned by the `health` endpoint, configure the following properties:

- `management.endpoint.health.show-details`
- `management.endpoint.health.show-components`

For more information, about health endpoints, see Spring Boot documentation.

Health for the workflow, Solace binder, and file-events binder components are exposed when `management.endpoint.health.show-components` is enabled. For example:

```
management:
  endpoint:
   health:
    show-components: always
    show-details: always
```

This configuration would always show the full details of the health check including the workflows and binders. The default value is `never`.

Workflow Health

A `workflows` health indicator is provided to show the health status for each of a connector's workflows. This health indicator has the following form:

```
{
  "status": "(UP|DOWN)",
  "components": {
    "<workflow-id>": {
      "status": "(UP|DOWN)",
      "details": {
        "error": "<error message>"
      }
    }
  }
}
```

Health Status	Description
`UP`	A status that indicates the workflow is functioning as expected.
`DOWN`	A status that indicates the workflow is unhealthy. Operator intervention may be required.

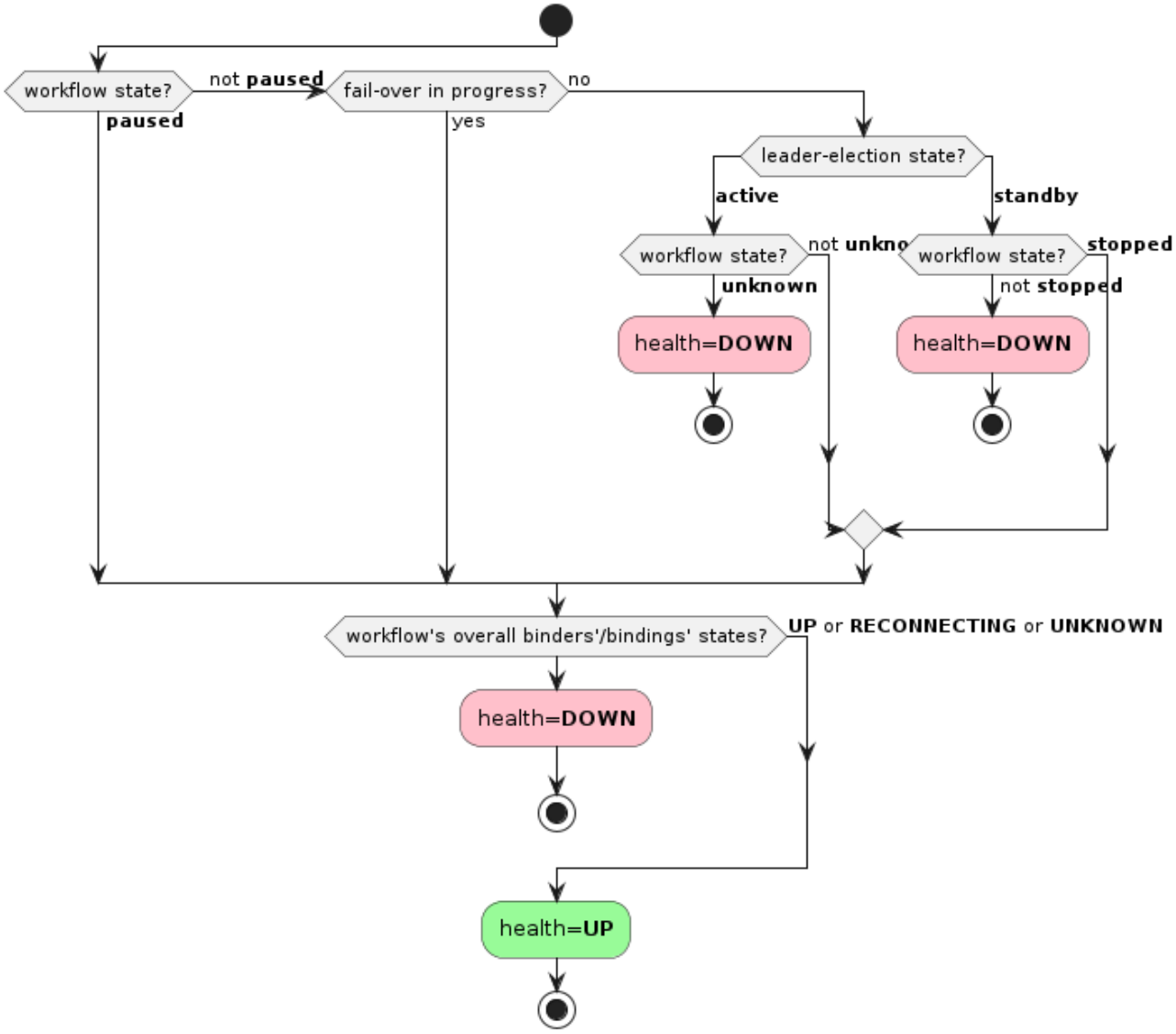

Figure 1. Workflow Health Resolution Diagram

This health indicator is enabled default. To disable it, set the property as follows:

```
management.health.workflows.enabled=false
```

Solace Binder Health

For details, see the Solace binder documentation.

Leader Election

The connector has three leader election modes for redundancy:

Leader Election Mode	Description
Standalone (Default)	A single instance of a connector without any leader election capabilities.
Active-Active	A participant in a cluster of connector instances where all instances are active.
Active-Standby	A participant in a cluster of connector instances where only one instance is active (i.e. the leader), and the others are standby.

Operators can configure the leader election mode by setting the following configuration:

```
solace.connector.management.leader-election.mode
=(standalone|active_active|active_standby)
```

Leader Election Modes: Standalone / Active-Active

When the connector starts, all enabled workflows start at the same time. The connector itself is considered as always active.

Leader Election Mode: Active-Standby

If the connector is in active-standby mode, a PubSub+ management session and management queue must be configured as follows:

```
solace.connector.leader-election.mode=active_standby

# Management session
# Exact same interface as solace.java.*
solace.connector.management.session.host=<management-host>
solace.connector.management.session.msgVpn=<management-vpn>
solace.connector.management.session.client-username=<client-username>
solace.connector.management.session.client-password=<client-password>
solace.connector.management.session.<other-property-name>=<value>

# Management queue name accessible by the management session
# Must have exclusive access type
solace.connector.management.queue=<management-queue-name>
```

To determine if the connector is `active` or `standby`, it creates a flow to the management queue. If this flow is active, then the connector's state is `active` and will start its enabled workflows. Otherwise, if this flow is inactive, then the connector's state is `standby` and will stop its enabled workflows.

At a macro level for a cluster of connectors, failover only happens when there are infrastructure failures (for example, the JVM goes down or networking failures to the management queue).

If a workflow fails to start or stop during failover, it will retry up to some maximum defined by the configuration option, `solace.connector.management.leader-election.fail-over.max-attempts`.

During failover, the connector attempts to start or stop all enabled workflows. After an attempt has been made to start or stop each workflow, the connector transitions to the active/standby mode regardless of the status of the workflows.

Leader Election Management Endpoint

A custom `leaderelection` management endpoint was provided using Spring Actuator.

Operators can navigate to the connector's `leaderelection` management endpoint to view its leader election status.

Endpoint	Operation	Payloads
`/leaderelection`	Read (HTTP `GET`)	**Request:** None. **Response:** `{ "mode": { "type": "(standalone | active_active | active_standby)", ① "state": "(active | standby)", ② "source": { ③ "queue": "<management-queue-name>", "host": "<management-host>", "msgVpn": "<management-vpn>" } } }` ① Mandatory parameter in output ② Mandatory parameter in output ③ Optional section. Appears only when `type` is set to `active_standby`.

Workflow Management

Workflow Management Endpoint

A custom `workflows` management endpoint using Spring Actuator is provided to manage workflows.

To enable the `workflows` management endpoint:

```
management:
  endpoints:
    web:
      exposure:
        include: "workflows"
```

Once the `workflows` management endpoint is enabled, the following operations can be performed:

Endpoint	Operation	Payloads
`/workflows`	Read (HTTP `GET`)	**Request:** None. **Response:** Same payload as the `/workflows/{workflowId}` read operation, but as a list of all workflows.
`/workflows/{workflowId}`	Read (HTTP `GET`)	**Request:** None. **Response:** `{ "id": "<workflowId>", "enabled": (true|false), "state": "(running|stopped|paused|unknown)", "inputBindings": [ "<input-binding>" ], "outputBindings": [ "<output-binding>" ] }`
`/workflows/{workflowId}`	Write (HTTP `POST`)	**Request:** `{ "state": "STARTED|STOPPED|PAUSED|RESUMED" }` **Response:** None.

Only workflows with Solace PubSub+ consumers (where the `solace` binder is defined in the `input-#`) support pause/resume.

Some features require for the connector to manage workflow lifecycles. There's no guarantee that workflow states continue to persist when write operations are used to change the workflow states while such features are in use.

For example: When the connector is configured in the active-standby leader election mode, workflows will automatically transition from `running` to `stopped` when the connector fails over from `active` to `standby`. Vice-versa for a failover in the opposite direction.

Workflow States

A workflow's state is defined as the aggregate states of its bindings (see the `bindings` management endpoint) as follows:

Workflow State	Condition
`running`	All bindings have `state="running"`.
`stopped`	All bindings have `state="stopped"`.
`paused`	All consumer bindings and all pausable producer bindings have `state="paused"`.
`unknown`	None of the other states. Represents an inconsistent aggregate binding state.

When the producer or consumer binding is not implementing Spring's Lifecycle interface, Spring always reports the bindings as `state=N/A`. The `state=N/A` is ignored when deciding the overall state of the workflow. For example, if the consumer's binding is `state=running` and producer's binding `state=N/A` (or vise-versa), the workflow state would be `running`.

For more information about binding states, see Spring Cloud Stream: Binding visualization and control.

Metrics

This connector uses Spring Boot Metrics that leverages Micrometer to manage its metrics.

Connector Meters

In addition to the meters already provided by the Spring framework, this connector introduces the following custom meters:

Name	Type	Tags	Description	Notes

`solace.connector.process`	`Timer`	`type: channel`
		
		`name: <bindingName>`
		
		`result: (success|failure)`
		
		`exception: (none|exception simple class name)`
			The processing time.	This meter is a rename of `spring.integration.send` whose `name` tag matches a binding name.

`solace.connector.error.process`	`Timer`	`type: channel`
		
		`name: <bindingNames>`
		
		`result: (success|failure)`
		
		`exception: (none|exception simple class name)`
			The error processing time.	This meter is a rename of `spring.integration.send` whose `name` tag matches an input binding's error channel name (`<destination>.<group>.errors`).
				
				Meters might be merged under the same `name` tag (delimited by `|`) if multiple bindings have the same error channel name (for example, bindings can have a matching `destination`, `group`, or both). **NOTE: Setting a binding's `group` is not supported.**

`solace.connector.message.size.payload`	`DistributionSummary` Base Units: `bytes`	`name: <bindingName>`	The message payload size.	

Name	Type	Tags	Description	Notes
solace.connector.message.size.total	DistributionSummary Base Units: bytes	name: <bindingName>	The total message size.	
solace.connector.publish.ack	Counter Base Units: acknowledgments	name: <bindingName> result: (success|failure) exception: (none|exception simple class name)	The publish acknowledgment count.	
solace.connector.transform.expressions.count	Gauge Base Units: expressions	workflow.id: <workflowId>	Transformation expressions count	
solace.connector.transform.time	Timer	workflow.id: <workflowId> result: (success|failure)	Transformation execution time	This is the aggregate time for all expressions used to transform a single message.

The solace.connector.process meter with result=failure is not a reliable measure of tracking the number of failed messages. It only tells you how many times a step processed a message (or batch of messages), how long it took to process that message, and if that step completed successfully.

Instead, we recommend that you use a combination of solace.connector.error.process and solace.connector.publish.ack to track failed messages.

Add a Monitoring System

By default, this connector includes the following monitoring systems:

- Datadog
- Dynatrace
- Influx
- JMX
- OpenTelemetry (OTLP)

- StatsD

To add additional monitoring systems, add the system's `micrometer-registry-<system>` JAR file and its dependency JAR files to the connector's classpath. The included systems can then be individually enabled/disabled by setting `management.<system>.metrics.export.enabled=true` in the `application.yml`.

Security

Securing Endpoints

Exposed Management Web Endpoints

There are many endpoints that are automatically enabled for this connector. For a comprehensive list, see Management and Monitoring Connector.

The `health` endpoint only returns the root status by default (i.e. no health details).

To enable other management endpoints, see Spring Actuator Endpoints.

Authentication & Authorization

This release of the connector only supports basic HTTP authentication.

By default, no users are created unless the operator configures them in their configuration file. The configuration parameters responsible for security are as follows:

```
solace:
  connector:
    security:
      enabled: true
      users:
      - name: user1
        password: pass
      - name: admin1
        password: admin
        roles:
        - admin
```

In the above example, we have created two users:

- **user1**: Has access to perform GET (Read) requests.
- **admin1**: Has access to perform GET and POST (Read & Write) requests.

To fully disable security and permit anyone to access the connector's web endpoints, operators can configure the `solace.connector.security.enabled` parameter `false`.

While these properties could be defined in an `application.yml` file, we recommend that you use environment variables to set secret values.

The following example shows you how to define users using environment variables:

```
# Create user with no role (i.e. read-only)
SOLACE_CONNECTOR_SECURITY_USERS_0_NAME=user1
```

```
SOLACE_CONNECTOR_SECURITY_USERS_0_PASSWORD=pass

# Create user with admin role
SOLACE_CONNECTOR_SECURITY_USERS_1_NAME=admin1
SOLACE_CONNECTOR_SECURITY_USERS_1_PASSWORD=admin
SOLACE_CONNECTOR_SECURITY_USERS_1_ROLES_0=admin
```

In the above example, we have created two users:

- **user1**: Has access to perform GET (Read) requests.
- **admin1**: Has access to perform GET and POST (Read & Write) requests.

`solace.connector.security.users` is a list. When users are defined in multiple sources (different `application.yml` files, environment variables, and so on), overriding works by replacing the entire list. In other words, you must pick one place to define all your users, whether in a **single** application properties file or as environment variables.

For more information, see Spring Boot - Merging Complex Types.

TLS

TLS is disabled by default.

To configure TLS, see Spring Boot - Configure SSL and TLS Setup in Spring.

Consuming Object Messages

For the connector to process object messages, it needs access to the classes which define the object payloads.

Assuming that your payload classes are in their own project(s) and are packaged into their own jar(s), place these jar(s) and their dependencies (if any) onto the connector's classpath.

It is recommended that these jars only contain the relevant payload classes to prevent any oddities.

In the jar(s), your class files must be archived in the same directory/classpath as the application that publishes them.

e.g. If the source application is publishing a message with payload type, `MySerializablePayload`, defined under classpath `com.sample.payload`, then when packaging the payload jar for the connector, the `MySerializablePayload` class must still be accessible under the `com.sample.payload` classpath.

Typically, build tools such as Maven or Gradle will handle this when packaging jars.

Adding External Libraries

The connector jar uses the `loader.path` property as the recommended mechanism for adding external libraries to the connector's classpath.

See Spring Boot - PropertiesLauncher Features for more info.

To add libraries to the connector's container image, see the connector's container documentation.

Configuration

Providing Configuration

For information about about how the connector detects configuration properties, see Spring Boot: Externalized Configuration.

Converting Canonical Spring Property Names to Environment Variables

For information about converting the Spring property names to environment variables, see the Spring documentation.

Spring Profiles

If multiple configuration files exist within the same configuration directory for use in different environments (development, production, etc.), use Spring profiles.

Using Spring profiles allow you to define different application property files under the same directory using the filename format, `application-{profile}.yml`.

For example:

- `application.yml`: The properties in non-specific files that always apply. Its properties are overridden by the properties defined in profile-specific files.
- `application-dev.yml`: Defines properties specific to the development environment.
- `application-prod.yml`: Defines properties specific to the production environment.

Individual profiles can then be enabled by setting the `spring.profiles.active` property.

See Spring Boot: Profile-Specific Files for more information and an example.

Configure Locations to Find Spring Property Files

By default, the connector detects any Spring property files as described in the Spring Boot's default locations.

- If you want to add additional locations, add `--spring.config.additional-location=file:<custom-config-dir>` (This parameter is similar to the example command in Quick Start: Running the connector via command line).
- If you want to exclusively use the locations that you've defined and ignore Spring Boot's default locations, add `--spring.config.location=optional:classpath:/,optional:classpath:/config/,file:<custom-config-dir>`.

For more information about configuring locations to find Spring property files, see Spring Boot documentation.

If you want configuration files for multiple, different connectors within the same `config` directory for use in different environments (such as development, production, etc.), we recommend that you use Spring Boot Profiles instead of child directories. For example:

- Set up your configuration like this:
  - `config/application-prod.yml`
  - `config/application-dev.yml`
- Do not do this:
  - `config/prod/application.yml`
  - `config/dev/application.yml`

Child directories are intended to be used for merging configuration from multiple sources of configuration properties. For more information and an example of when you might want to use multiple child directories to compose your application's configuration, see the Spring Boot documentation.

Obtaining Build Information

Build information, including version, build date, time and description is enabled by default via Spring Boot Actuator Info Endpoint. By default, every connector shares all information related to its build only.

Below is the structure of the output data:

```
{
  "build": {
    "version": "<connector version>",
    "artifact": "<connector artifact>",
    "name": "<connector name>",
    "time": "<connector build time>",
    "group": "<connector group>",
    "description": "<connector description>",
    "support": "<support information>"
  }
}
```

If you want to exclude build data from the output of the `info` endpoint, set `management.info.build.enabled` to `false`.

Alternatively, if you want to disable the info endpoint entirely, you can remove 'info' from the list of endpoints specified in `management.endpoints.web.exposure.include`.

Spring Configuration Options

This connector packages many libraries for you to customize functionality. Here are some

references to get started:

- Spring Cloud Stream
- Spring Cloud Stream Binder for Solace PubSub+
- Spring Logging
- Spring Actuator Endpoints
- Spring Metrics

Connector Configuration Options

The following table lists the configuration options. The following options in **Config Option** are prefixed with `solace.connector.`:

Config Option	Type	Default Value	Description
`management.leader-election.fail-over.max-attempts`	`int` **Constraint:** `> 0`	`3`	The maximum number of attempts to perform a fail-over.
`management.leader-election.fail-over.back-off-initial-interval`	`long` **Constraint:** `> 0`	`1000`	The initial interval (milliseconds) to back-off when retrying a fail-over.
`management.leader-election.fail-over.back-off-max-interval`	`long` **Constraint:** `> 0`	`10000`	The maximum interval (milliseconds) to back-off when retrying a fail-over.
`management.leader-election.fail-over.back-off-multiplier`	`double` **Constraint:** `>= 1.0`	`2.0`	The multiplier to apply to the back-off interval between each retry of a fail-over.
`management.leader-election.mode`	`enum` One of: • `standalone` • `active_active` • `active_standby`	`standalone`	The connector's leader election mode. **standalone:** A single instance of a connector without any leader election capabilities. **active_active:** A participant in a cluster of connector instances where all instances are active. **active_standby:** A participant in a cluster of connector instances where only one instance is active (i.e. the leader), and the others are standby.
`management.queue`	`string`	`null`	The management queue name.

Config Option	Type	Default Value	Description
`management.session.*`	See Spring Boot Auto-Configuration for the Solace Java API		Defines the management session. This has the same interface as that used by `solace.java.*`. See Spring Boot Auto-Configuration for the Solace Java API for more info.
`security.enabled`	`boolean`	`true`	If `true`, security is enabled. Otherwise, anyone has access to the connector's endpoints.
`security.users[<index>].name`	`string`	`null`	The name of the user.
`security.users[<index>].password`	`string`	`null`	The password for the user.
`security.users[<index>].roles`	`list<string>` Valid values: • `admin`	`empty list (i.e. read-only)`	The list of roles that the specified user has. It has read-only access if no roles are returned.

Workflow Configuration Options

These configuration options are defined under the following prefixes:

- `solace.connector.workflows.<workflow-id>.`: If the options support per-workflow configuration and the default prefixes.
- `solace.connector.default.workflow.`: If the options support default workflow configuration.

Config Option	Type	Default Value	Description
`enabled`	`boolean`	`false`	If `true`, the workflow is enabled. Cannot be set at the default workflow level.
`transform.enabled`	`boolean`	`false`	If `true`, message transformation is enabled for the workflow. Disables the legacy `transform-headers.*` and `transform-payload.*` options. See Message Transforms for more info.

Config Option	Type	Default Value	Description
`transform.source-payload.content-type`	`string` One of: • `application/vnd.solace.micro-integration.unspecified` • `application/json`	`application/vnd.solace.micro-integration.unspecified`	The content type to interpret the source payload as. See Content Type Interpretation for more info.
`transform.target-payload.content-type`	`string` One of: • `application/vnd.solace.micro-integration.unspecified` • `application/json`	`application/vnd.solace.micro-integration.unspecified`	The content type to interpret and serialize the target payload as. See Content Type Interpretation for more info.
`transform.expressions[<index>].transform`	`string` A SpEL expression		A SpEL expression at some `<index>` in the ordered list of expressions to transform the message. See Message Transform for more info.
`transform-headers.expressions`	`Map<string, string>` **Key:** A header name. **Value:** A SpEL string that accepts `headers` as parameters.	`empty map`	⚠ Deprecated. Use Message Transforms (`transform.*`) instead. A mapping of header names to header value SpEL expressions. The SpEL context contains the `headers` parameter that can be used to read the input message's headers.
`acknowledgment.publish-async`	`boolean`	`false`	If `true`, publisher acknowledgment processing is done asynchronously. The workflow's consumer and producer bindings must support this mode, otherwise the publisher acknowledgments are processed synchronously regardless of this setting.

Config Option	Type	Default Value	Description
`acknowledgment.back-pressure-threshold`	int **Constraint:** `>= 1`	`255`	The maximum number of outstanding messages with unresolved acknowledgments. Message consumption is paused when the threshold is reached to allow for producer acknowledgments to catch up.
`acknowledgment.publish-timeout`	int **Constraint:** `>= -1`	`600000`	The maximum amount of time (in millisecond) to wait for asynchronous publisher acknowledgments before considering a message as failed. A value of `-1` means to wait indefinitely for publisher acknowledgments.

File Events Source (File to Events)

Configuring Input Destination

The configuration property `spring.cloud.stream.bindings.input-0.destination` defines the input destination for the source micro integration (MI) instance.

This property varies based on the fileType being processed. The behavior and required setup differ for each type, as detailed below.

For dynamic expressions used in topic configurations (e.g., ${FNAME}), refer to the Dynamic Expressions section for more details.

FileType 1: Static Files

For static files (processed sequentially), create a new configuration file with the following format for each entry:

```
SourceFilePath|OptionalDestinationFilePath|OptionalCustomBaseDestinationTopic|Optional
CustomDynamicDestinationTopic
```

The configuration file can contain multiple lines (entries), and the MI will process each source path sequentially.

Column	Description
`SourceFilePath`	Absolute path of the static source file.
`OptionalDestinationFilePath`	Absolute path of the destination file. This value is included in the header metadata. The sink MI instance can use this to reference the original file name provided by the source MI.
`OptionalCustomBaseDestinationTopic`	Custom base topic for each file path, if needed. If not defined, the default output-0.destination value is used.
`OptionalCustomDynamicDestinationTopic`	Fine-grained topic per file, if needed. Metadata events are published to the base topic (if configured), while actual file events are published to this custom dynamic topic. Dynamic expressions can be used to populate topic values at runtime. For more detailed usage and to know about more dynamic expressions that can be used, please refer to the section Dynamic Expressions

Once the file is created provide the absolute path of this new configuration file to destination

```
spring.cloud.stream.bindings.input-0.destination: /path/to/static_files.cfg
```

Routing Logic:

If the dynamic topic is not defined, all events are sent to the optional custom base topic. If neither the dynamic nor base topic is defined, the MI uses the default output-0.destination.

Example

A sample configuration file entry for a static file:

```
/home/centos/local-source/sftp|/home/centos/local-
sink/sftp|solace/fc/source/sftp|solace/fc/source/sftp/${FNAME}
```

This example processes the source file at

```
/home/centos/local-source/sftp,
with destination /home/centos/local-sink/sftp,
base topic solace/fc/source/sftp,
and dynamic topic solace/fc/source/sftp/${FNAME} (where ${FNAME} resolves to the file
name).
```

The file can have multiple such lines for sequential processing.

FileType 2: Dynamic Files (Continuously Written)

For dynamic files that are continuously written to, only one file is supported per source MI instance. No new configuration file is required. Instead, assign the single line entry directly to the property:

```
spring.cloud.stream.bindings.input-0.destination:
SourceFilePath|OptionalDestinationFilePath|OptionalCustomBaseDestinationTopic|Optional
CustomDynamicDestinationTopic
```

The format follows the same structure as FileType 1.

Column	Description
`SourceFilePath`	Absolute path of the dynamic source file.
`OptionalDestinationFilePath`	Absolute path of the destination file. This value is included in the header metadata.
`OptionalCustomBaseDestinationTopic`	Custom base topic, if needed. If not defined, the default output-0.destination value is used.
`OptionalCustomDynamicDestinationTopic`	Custom dynamic topic, if needed. Supports dynamic expressions.

Routing Logic:

Same as FileType 1.

Example

Direct property assignment:

```
    `spring.cloud.stream.bindings.input-0.destination`: /var/log/dynamic-
app.log|/var/log/dynamic-sink.log|solace/dynamic/base|solace/dynamic/events/${FNAME}
This monitors the continuously written file at /var/log/dynamic-app.log, routing
metadata to solace/dynamic/base and events to solace/dynamic/events/${FNAME}.
```

FileType 3: Directory Single Level (Non-Recursive)

For single-level directories (non-recursive), use the same approach as static files, but specify directory paths instead of file paths.

Create a new configuration file with the following format:

```
SourceDirectoryPath|OptionalDestinationDirectoryPath|OptionalCustomBaseDestinationTopi
c|OptionalCustomDynamicDestinationTopic
```

The configuration file can contain multiple lines, processed sequentially. All files in the specified directories (one level deep) will be streamed.

The column descriptions are analogous to FileType 1, with "FilePath" replaced by "DirectoryPath".

Once the file is created provide the absolute path of this new configuration file to destination

```
spring.cloud.stream.bindings.input-0.destination: /path/to/dir_paths.cfg
```

Routing Logic:

Same as FileType 1.

Example

A sample configuration file entry for a single-level directory:

```
/home/centos/local-source/docs|/home/centos/local-
sink/docs|solace/fc/source/docs|solace/fc/source/docs/${FNAME}
```

This streams all files from /home/centos/local-source/docs (non-recursively), with destination directory /home/centos/local-sink/docs, base topic solace/fc/source/docs, and dynamic topic solace/fc/source/docs/${FNAME}. The file can have multiple such lines for multiple directories.

FileType 4: Directory Nested (Recursive)

For nested directories, the setup is identical to FileType 3, but the search is recursive—all subdirectories and their files will be streamed.

Use the same configuration file format:

```
SourceDirectoryPath|OptionalDestinationDirectoryPath|OptionalCustomBaseDestinationTopi
c|OptionalCustomDynamicDestinationTopic
```

The column descriptions are analogous to FileType 3.

Once the file is created provide the absolute path of this new configuration file to destination

```
spring.cloud.stream.bindings.input-0.destination: /path/to/dir_paths.cfg
```

Routing Logic:

Same as FileType 1.

Example

A sample configuration file entry for a nested directory:

```
/home/centos/local-source/projects|/home/centos/local-
sink/projects|solace/fc/source/projects|solace/fc/source/projects/${FNAME}
```

This recursively streams all files from /home/centos/local-source/projects and subdirectories, with destination directory /home/centos/local-sink/projects, base topic solace/fc/source/projects, and dynamic topic solace/fc/source/projects/${FNAME}. The file can have multiple such lines for multiple root directories.

Adding Dynamic Dates in Paths

Dynamic date expressions can be added in Source/Destination file/directory paths or topics

Example 1

```
/home/centos/local-source/${DATE(%1$tY%1$tm%1$td)}/file
```

The above configured path will resolve to basic the system's current date

```
/home/centos/local-source/20251001/file
```

Example 2

```
/home/centos/local-source/${DATE[-1d](%1$tY%1$tm%1$td)}/file
```

The above configured path will resolve to basic the system's current date minus 1 day

```
/home/centos/local-source/20250930/file
```

For more details on the usage of these dynamic expressions refer to the sections Dynamic Expressions and Date Format Specifiers

Understanding Topic Configurations

The topics all events will publish on are defined at different levels to achieve flexible routing per file/dir/path

Here's the list of different topics defined in the configuration

Topic Type	Description	Configuration
`Default Base Topic Global`	`Mandatory`	Default topic is used to stream all events - meta data events as well as file events. Configured at `spring.cloud.stream.bindings.output-0.destination`
`Default Dynamic Topic Global`	`Optional`	Default dynamic topic is used to stream file events on a topic expression that's dynamically resolved in real time. Configured at `file-events.source.advanced.dynamicTopic`
`Custom Base Topic per File/Dir Path`	`Optional`	Default Topic can be overwritten if each configured path for static/directory fileType needs to be published on different topic. Configured as Optional Custom Base Topic (Column 3) in the configuration file created for configuring paths. or In case of dynamic file type - `spring.cloud.stream.bindings.input-0.destination`
`Custom Dynamic Topic per File/Dir Path`	`Optional.`	Default dynamic Topic can be overwritten if each configured path for static/directory fileType needs to be published on different topic. Configured as Optional Custom Dynamic Topic Expression (Column 4) in the configuration file created for configuring paths. or In case of dynamic file type - `spring.cloud.stream.bindings.input-0.destination`

Important Note

The Meta Data events are always published on the base Topics (Default Base Topic Global/ Custom Base Topic per Path)

The File Events can be published on any of the topics based on how the topic priority is resolved

Topic Priority

For Meta Data events:

```
`Default Topic per File/Dir Path` >
`Default Topic`
```

For File Events

```
`Custom Dynamic Topic per File/Dir Path` >
`Custom Base Topic per File/Dir Path` >
`Default Dynamic Topic Global` >
`Default Base Topic Global`
```

Which means if `Custom Dynamic Topic per File/Dir Path` is configured it will supersede all other topic and if no other topic is configured `Default Base Topic Global` will be used

Last Value Queue (LVQ Checkpoint) Topic

The checkpoint event meant for LVQ is used by the source MI to track the its state.

This is important for preventing retransmission of files in case of scheduled or unexpected restarts

The LVQ checkpoint topic will be created by the source MI itself and will resolve to `` `${Default Base Topic Global}/checkpoint ``

To `override` the base topic for LVQ checkpoint - configure `file-events.source.lvq.destination`.

If configured the source MI will publish all checkpoints to `` `${file-events.source.lvq.destination}/checkpoint ``

`Note`: The source MI will internally append the lvq topic with "/checkpoint"

Configuring Queues on Solace

Apart from the management queues(optional), Two mandatory queues are required to be created on Solace Broker for the very least

1. The File Events Queue - For streaming file events and data events(Optional). The Subscriber can consume file events from this queue
2. The Last Value Queue - For storing the intermediate state of source MI

Topic Subscriptions for File Types

Topic subscriptions define the queues and associated topics used for message routing.

Static and Dynamic File Types

For static and dynamic file types, the following queue and topic subscriptions apply:

Queue Name	Topic Subscriptions Examples
q.file.connector.lvq (Last Value Queue)	solace/fc/source/test/checkpoint (LVQ Checkpoint topic) solace/fc/source/test (Resolved File Events topic) solace/fc/source/test (Optional Resolved Meta Data Events topic)
q.file.event.queue (File Events Queue)	solace/fc/source/test (Resolved File Events topic) solace/fc/source/test (Optional Resolved Meta Data Events topic)

Directory and Recursive Directory Options

For directory (single-level and recursive) file types, the following queue and topic subscriptions apply:

Queue Name	Topic Subscriptions Examples
q.file.connector.lvq (Last Value Queue)	solace/fc/source/test/checkpoint (LVQ Checkpoint topic)
q.file.event.queue (File Events Queue)	solace/fc/source/test (Resolved File Events topic) solace/fc/source/test (Optional Resolved Meta Data Events topic)

File Events Source (File to Events) Configuration Options

These configuration options are all prefixed by `file-events.source.`:

Scheduler Configuration

Config Option	Type	Valid Values	Default Value	Description
`scheduler.restart_time_sec`	`int`	`any`	`0`	This property enables source connector to run at periodic intervals. This is needed when you want to replicate your modified files periodically under a directory. Integer value representing number of seconds to wait and start the next replication. Set to Zero if scheduler is not required(Recommended for Dynamic file, since connector first replication runs till EOD is reached). Following is scheduler behaviour w.r.t eod_time_sec 1. restart_time_sec: 10 and eod_time_sec: 104400(5 A.M converted to number of seconds 24+5)*3600 , source connector will stop at 5 A.M 2. restart_time_sec: 10 and eod_time_sec: -1 , source connector will keep on running 3. restart_time_sec: 10 and eod_time_sec: 0 , source connector will exit at midnight 4. restart_time_sec: 0 and eod_time_sec: -1 , source connector will exit after first replication

General Configuration

Config Option	Type	Valid Values	Default Value	Description
general.miID	string	any	empty	Unique ID for the connector instance
general.file_type	int	1,2,3,4	empty	Type of file (1 → static 2 → Dynamic 3 → Directory first level 4 → Directory recursive)
general.state_backup_path	string	any	empty	Absolute file location to store Source Connector's state information. Only used in case file_to_events is enabled. The file should end in .cfg format(Ex:<path>/source_connector_state_backup.cfg. The file will be created by connector if it doesn't exist.
general.max_files_allowed	int	any	99999	Set this value to limit the number of files in a replication. Connector will consider the files for replication until the limit is reached
general.clear_state_on_eod	int	0, 1	0	Connector preserves the last successful file state in backup file, so that it resumes from checkpoint on next restart. If this property is set to 1 connector will clear the file state in backup file when End of Day configured time is reached. Set to 0 if file state in backup file need not be cleared

Config Option	Type	Valid Values	Default Value	Description
general.eod_time_sec	int	-1,0, any number > 0	-1	Connector will exit once configured EOD is elapsed. For example if connector need to be exited every day at 5 A.M EOD should be configured to 104400. The formula to calculate EOD is (24 Hours + (number of hours(24 hour format) when connector should exit))*3600. In our example (24+5)*3600 = 104400. Set this to 0 is connector need to exit at midnight every day and -1 to disable this check

Date Events Configuration

Config Option	Type	Valid Values	Default Value	Description
data_events.connectorStartEvent	boolean	true, false	false	If Set to true, An additional event with Header(EVENT_TYPE → "START") will be published to the base destination topic name once a new location path is processed Additional Header included with Data Events are are follows SRC_PATH - Source File Path. The file being read DEST_PATH - Destination File Path if configured in the source configuration

Config Option	Type	Valid Values	Default Value	Description
`data_events.connectorCompleteEvent`	`boolean`	`true, false`	`false`	If Set to true, An additional event with Header(EVENT_TYPE → "COMPLETE") will be published to the base destination topic name once all the files are completely processed in the configured location path Additional Header included with Data Events are are follows SRC_PATH - Source File Path. The file being read DEST_PATH - Destination File Path if configured in the source configuration TOTAL_FILES - Total Files Processed TOTAL_EVENTS - Total Events Processed for all files TOTAL_DATA_PROCESSED - Total Data Processed for all files

Config Option	Type	Valid Values	Default Value	Description
data_events.fileStartEvent	boolean	true, false	false	If Set to true, An additional event with Header(EVENT_TYPE → "FILE_START") will be published to the base destination topic name once a new file processing starts Additional Header included with Data Events are are follows SRC_PATH - Source File Path. The file being read DEST_PATH - Destination File Path if configured in the source configuration
data_events.fileCompleteEvent	boolean	true, false	false	If Set to true, An additional event with Header(EVENT_TYPE → "FILE_COMPLETE") will be published to the base destination topic name once the file is completely processed. Additional Header included with Data Events are are follows SRC_PATH - Source File Path. The file being read DEST_PATH - Destination File Path if configured in the source configuration TOTAL_EVENTS - Total Events Processed for the current file TOTAL_DATA_PROCESSED - Total Data Processed for the current file

Advanced Configuration

Config Option	Type	Valid Values	Default Value	Description
advanced.fileFormat	int	1,2,3,4,5,6,11,12	1	1 = Line by line CSV file streaming. (Configure advanced.delimited) 2 = Delimited Lines file streaming. Delimiter can be defined in the parameter eventDelimiter. (Configure advanced.delimited) 3 = JSON File Streaming. 4 = XML File Streaming. 5 = Fixed Length File (Configure advanced.fixed) 6 = Line Events File. Stream entire line as event. Custom Line Separator or eventDelimiter can be configured. Default is New line (Configure advanced.line) 11 = Line by line CSV file streaming Version 2. (Configure advanced.delimited) 12 = Delimited Lines file streaming. Delimiter can be defined in the parameter eventDelimiter. (Configure advanced.delimited)

Config Option	Type	Valid Values	Default Value	Description
advanced.dynamicTopic	String	any	empty	For FileFormats value 1 and 2 use when columnDelimiter and columnHeaderMap are defined. Add dynamic column values of the delimited file in the topic Structure. For example if columnDelimiter is "," and columnHeaderMap is "employee_id,first_name,last_name,city,country" Dynamic topic can be set as 'solace/dynamic/topic/${column1}/${column5}/${city}'. You can use column<position> or the column name defined in the header as the variable expression
advanced.jsonPath	String	any	empty	provide a jsonPath filter to stream json objects from a json file. Use with fileFormat value 3 i.e. JSON File Streaming
advanced.xPath	String	any	empty	provide a xPath filter to stream xml objects from a xml file. Use with fileFormat value 4 i.e. XML File Streaming
advanced.postProcessFileMovePathExpr	String	any	empty	provide a path which can be a regular absolute path for the new file or use Dynamic Expressions in the path as follows postProcessFileMovePathExpr: `"/home/archived/done_${DATE(%1$td%1$tm%1$tY)}/${FNAME}"` For more detailed usage and to know about more dynamic expressions that can be used within this file move path expression, please refer to the section Dynamic Expressions

Lines as Events Configuration

Config Option	Type	Valid Values	Default Value	Description
`advanced.line.eventDelimiter`	`Char`	`any`	`'New line'`	This setting would define a delimiter for separating events in the file format 6 (Entire line as event). If enabled, it would likely be used to distinguish different events within a single file.
`advanced.line.rootObjectName`	`String`	`any`	`'Empty'`	If the output needs to be enclosed in a root object - Name of the root element. Typically used in `JSON`, `XML` or `Text` Output format. Will not work if the output format is `raw`

CSV/Delimited Configuration

Config Option	Type	Valid Values	Default Value	Description
`advanced.delimited.eventDelimiter`	`Char`	`any`	`'\n'`	This setting would define a delimiter for separating events. If enabled, it would likely be used to distinguish different events within a single file.
`advanced.delimited.columnDelimiter`	`String`	`any`	`','`	Specifies the character or sequence of characters used to separate columns in the file. In case the fileFormat values are 1 and 2, use a column Delimiter to map the columns to headers and create a final json payload. Used with columnHeaderMap or if useFirstEventAsHeader is set to true

Config Option	Type	Valid Values	Default Value	Description
advanced.delimited.columnHeaderMap	String	any	empty	Specifies the column headers when they are not explicitly present in the file. Provide a header map to use dynamicTopic OR to create a JSON payload from a delimited file if sendDelimitedEventPayloadAsBinary is set to false. Example - 'SNO,Employee_name,Employee_address,city'. To send payload as raw binary, set sendDelimitedEventPayloadAsBinary to true
advanced.delimited.ignoreFirstEventOfDelimitedFile	boolean	true, false	false	Determines whether the first event (row) should be ignored. It is set to false, meaning all rows are processed.
advanced.delimited.useFirstEventAsHeader	boolean	true, false	false	When set to true, the first row of the file is treated as the header, defining column names, this map will then be used to create dynamicTopic Or a JSON payload. Value for columnDelimiter should be set for using this property.
advanced.delimited.trimColumns	boolean	true, false	false	When true, whitespace around column values is removed.
advanced.delimited.quoteCharacter	Char	any	'"'	Defines the character used for quoting values, default value set to a double quote (")

Config Option	Type	Valid Values	Default Value	Description
advanced.delimited.eventFilterExpression	String	any	empty	Filters events based on conditions on the column values present in the event. For example, if one of the columns is Age. Use "Age < 30" to filter and publish events only which pass this condition. For more detailed usage, please refer to the section List of CSV/Delimited Events Filter Expressions
advanced.delimited.columnFilter	String	any	empty	Specifies which columns are included in the output event. Example - 'SNO,Employee_name,Employee_address'. Provide columnDelimiter separated list of column headers that needs to be included in the payload
advanced.delimited.sendDelimitedEventPayloadAsBinary	boolean	true, false	true	If Set to true, the event/record payload for Delimited file (fileFormat values 1 and 2) will be published as binary. If set to false, the payload will be sent as json if advanced.delimited.columnDelimiter and advanced.delimited.columnHeaderMap are set
advanced.delimited.outputColumnDelimiter	String	any	Same as advanced.delimited.columnDelimiter	Specifies the character or sequence of characters used to separate columns in the output. Used only when advanced.delimited.sendDelimitedEventPayloadAsBinary is set to true and columHeaderMap or useFirstEventAsHeader is set

Directory Wildcard Configuration

Config Option	Type	Valid Values	Default Value	Description
directory_wildcard.wildcard_type	int	0, 1, 2	0	Set this property to apply regular expression on files. 0 → disabled 1 → whitelist files or directory 2 → blacklist files or directory
directory_wildcard.config_path	string	any	empty	provide absolute file path location containing regular expression. The file should be in .cfg format
directory_replication.start_time	int	-1, 0, any number > 0	-1	Set this property to filter files in directory based on modified time. -1 will consider the timestamp in checkpoint if available or will replicate all files again. 0 will override the timestamp in checkpoint and will consider files modified since midnight. Any value(epoch time) other than 0 or -1 will consider files modified after the epoch time

LVQ Configuration

Config Option	Type	Valid Values	Default Value	Description
solace_out.lvq	string	any	empty	Provide the LVQ name configured on Solace broker. Connector will connect to this queue on start to fetch checkpoint information.

Config Option	Type	Valid Values	Default Value	Description
solace_out.destination	string	any	empty	LVQ topic - Connector will publish checkpoint information and the base topic will be same as output destination topic configured in connection details section. LVQ topic is built as follows 1. In case of Static or Directory replication connector will append /checkpoint to base topic(<output-destination-topic>) and will publish data. The LVQ created on Solace broker should have this subscription <output-destination-topic>/checkpoint 2. In case of Dynamic file connector will not append /checkpoint to base topic(<output-destination-topic>) and will publish data. The LVQ created on Solace broker should have this subscription <output-destination-topic>.

Dynamic Expressions

Introduction

The **Dynamic Expressions Resolver** allows configuration files, file paths, topic names, and event payloads to include **dynamic, self-resolving placeholders**. At runtime, these placeholders are replaced with values derived from context — such as date/time, file attributes, topics, headers, or user-provided properties.

This makes configurations adaptive without manual edits, enabling automation in file naming, routing, and message enrichment.

Example

```
input.path = /data/incoming/${DATE(%1$tY%1$tm%1$td)}/input.csv
output.file = ${FILE(FBASE)}_${DATE[+1d](%1$tY%1$tm%1$td)}.done
encoded.file = #ENCODE_BASE64(${FILE(FBASE)}_${DATE(%1$tY%1$tm%1$td)})
```

Resolved example:

Input Expression	Resolved Value
`/data/incoming/${DATE(%1$tY%1$tm%1$td)}/input.csv`	`/data/incoming/20251004/input.csv`
`${FILE(FBASE)}_${DATE[+1d](%1$tY%1$tm%1$td)}.done`	`orders_20251005.done`
`#ENCODE_BASE64(${FILE(FBASE)}_${DATE(%1$tY%1$tm%1$td)})`	`b3JkZXJzXzIwMjUxMDA0`

Expression Syntax

Every **Level 1 expression** follows this general format:

```
${CATEGORY[OFFSET](PARAM)}
```

Component	Description	Example
`CATEGORY`	Expression type — e.g., `DATE`, `FILE`, `TOPIC`	`${DATE}`, `${FILE(FNAME)}`
`[OFFSET]`	(Optional) For date/time adjustments	`${DATE[+1d]}` adds one day
`(PARAM)`	(Optional) Format or key parameter	`${DATE(%1$tY-%1$tm-%1$td)}`

A **Level 2 operator** can then wrap the entire Level 1 expression:

```
#FUNCTION(${CATEGORY(args...)})
```

Important: - Only one Level 2 operator is allowed per expression (no chaining). - The Level 1 expression must be enclosed in `${…}` inside the Level 2 wrapper.

DATE Expressions

The `DATE` expression inserts a formatted timestamp. It supports arithmetic offsets and uses `java.util.Formatter` syntax.

Offset Options

Offset	Meaning	Example
`[+3d]`	Add 3 days	`${DATE[+3d](%1$tY-%1$tm-%1$td)}`
`[-2h]`	Subtract 2 hours	`${DATE[-2h](%1$tH:%1$tM)}`
`[+1w]`	Add 1 week	`${DATE[+1w](%1$tY%1$tm%1$td)}`
`[+15m]`	Add 15 minutes	`${DATE[+15m](%1$tH:%1$tM)}`

Examples

Expression	Output Example
`${DATE(%1$tY-%1$tm-%1$td)}`	`2025-10-04`
`${DATE[+1d](%1$tY%1$tm%1$td_%1$tH%1$tM)}`	`20251005_1425`
`${DATE[-1w](Week_%1$tU)}`	`Week_39`

FILE Expressions

The `FILE` category extracts metadata from file names or paths. Useful for generating output files, logs, or routing keys dynamically.

Parameters

Parameter	Description	Example	Output
`FNAME`	File name with extension	`${FILE(FNAME)}`	`orders.csv`
`FBASE`	File base name (no extension)	`${FILE(FBASE)}`	`orders`
`FEXT`	File extension	`${FILE(FEXT)}`	`csv`
`FPATH_A`	Absolute path	`${FILE(FPATH_A)}`	`/data/in/orders.csv`

FPATH_R	Relative path	${FILE(FPATH_R)}	incoming/orders.csv
FPATH_P	Parent folder path	${FILE(FPATH_P)}	/data/in
FNAME_P	Parent folder name	${FILE(FNAME_P)}	incoming

Case Variants

- ${FNAME_U} → ORDERS.CSV
- ${FBASE_L} → orders
- ${FEXT_U} → CSV

Examples

Expression	Output
${FILE(FBASE)}_${DATE(%1$td%1$tm%1$tY)}.done	orders_04102025.done
${FILE(FEXT_U)}	CSV
${FILE(FPATH_P)}	/mnt/data/input

TOPIC Expressions

Extract topic hierarchy components by index.

Topic	Expression	Result
acme/sales/canada/order/created	${TOPIC(1)}	acme
acme/sales/canada/order/created	${TOPIC(3)}	canada
acme/sales/canada/order/created	${TOPIC(-1)}	created

UMAP and HEADER Expressions

Reference user-defined properties or message headers.

Property Map	Expression	Output
{ "region": "NA", "type": "invoice" }	${UMAP(region)}	NA
{ "Content-Type": "application/json" }	${HEADER(Content-Type)}	application/json

Legacy Placeholders

Legacy forms remain supported for backward compatibility.

Legacy	Equivalent	Example
`${fname}`	`${FILE(FNAME)}`	`data.csv`
`${fpath_r}`	`${FILE(FPATH_R)}`	`incoming/data.csv`
`${fext_u}`	`${FILE(FEXT_U)}`	`CSV`
`${fpath_p}`	`${FILE(FPATH_P)}`	`/data/incoming`

Level 2 Operators (Post-Processing Tools)

Level 2 operators transform the **result of a single Level 1 expression**. They are always used in this form:

```
#FUNCTION(${EXPR})
```

Available Operators

Operator	Description	Example	Output Example
`ENCODE_BASE64`	Base64-encode the result	`#ENCODE_BASE64(${DATE(%1$tY-%1$tm-%1$td)})`	`MjAyNS0xMC0wNA==`
`DECODE_BASE64`	Decode Base64 text	`#DECODE_BASE64(${UMAP(encodedValue)})`	Decoded text
`ENCODE_URL`	URL-encode the result	`#ENCODE_URL(${FILE(FNAME)})`	`orders.csv` → `orders.csv`
`DECODE_URL`	Decode URL text	`#DECODE_URL(${UMAP(urlValue)})`	Decoded string
`MD5`	MD5 hash	`#MD5(${FILE(FBASE)})`	`e1cbb0c3879af8347246f12c559a86b5`
`SHA1`	SHA-1 hash	`#SHA1(${DATE(%1$tY)})`	`d3486ae9136e7856bc42212385ea797094475802`
`SHA256`	SHA-256 hash	`#SHA256(${FILE(FNAME)})`	`b94d27b9934d3e08…`
`SUBSTR(start | end)`	Extract substring (end optional)	`#SUBSTR(${FILE(FBASE)}	0	3)`	`ord`	`UPPER`
Uppercase conversion	`#UPPER(${FILE(FBASE)})`	`ORDERS`	`LOWER`
Lowercase conversion	`#LOWER(${FILE(FBASE)})`	`orders`	`UUID`
Generate a random UUID (ignores inner expr)	`#UUID(${ANYTHING})`	`b7e9a84d-f23a-4e4a-9cc3-ec7f27cd3f65`	`RANDOM(min

max)`	Generate random integer	`#RANDOM(${DATE(% 1$tY)}	10
99)`	45	TRIM	Trim spaces
#TRIM(${UMAP(name)})	Deepak	`REPLACE(target	replacement)`
Replace substring	`#REPLACE(${FILE(FB ASE)}	ord	buy)`

Example Scenarios

Scenario	Expression	Output
Base64 encode timestamp	#ENCODE_BASE64(${DATE(%1$tY-%1$tm-%1$td)})	MjAyNS0xMC0wNA==
Create MD5 of file base	#MD5(${FILE(FBASE)})	e1cbb0c3879af8347246f12c559a86b5
Convert topic region to upper case	#UPPER(${TOPIC(2)})	SALES
Random suffix for file	`${FILE(FBASE)}_#RANDOM(${DATE(%1$tY)}	100
999)`	orders_457	Replace substring in name
`#REPLACE(${FILE(FBASE)}	ord	sale)`
sales	Trim header value	#TRIM(${HEADER(User-Name)})

Troubleshooting and Best Practices

Issue	Possible Cause	Resolution
Expression not replaced	Unsupported syntax or null context	Verify ${…} and pass context parameters
Level 2 not applied	Missing ${} inside #FUNCTION()	Use #FUNC(${EXPR}) format
Wrong substring result	Index out of range	Ensure valid positions for SUBSTR
Unexpected Base64 result	Expression already encoded	Check if data is pre-encoded

Summary

Dynamic Expressions provide a powerful, declarative way to build adaptable configuration strings. With Level 1 categories (DATE, FILE, TOPIC, etc.) and Level 2 post-processors (ENCODE_BASE64, MD5, UPPER, etc.), users can construct dynamic paths, identifiers, and names **without modifying code or restarting systems**. They are especially useful for operations teams automating event file naming,

topic routing, and integration workflows.

Date Format Specifiers

These specifiers can be used in the DATE expressions such as ${DATE(<specifier>)} wherever allowed -

1. Dynamic Topic the source MI publishes events on
2. Source Paths from where the files will be read
3. Output binding destination. (Destination File Path) `spring.cloud.stream.bindings.output-<>.destination`
4. Advanced option `advanced.emptyEventsSubstitute`
5. File Pre Processors defined under configuration pre_process.<> - `prependToFile`, `prependToFirstEvent`, `prependToEvents`
6. File Post Processors defined under configuration post_process.<> - `appendToEvents`

Character	Specifier %1$t<c>	Description	Example
c	%1$tc	Complete date and time	Mon May 04 09:51:52 CDT 2009
F	%1$tF	ISO 8601 date	2004-02-09
D	%1$tD	U.S. formatted date (month/day/year)	02/09/2004
T	%1$tT	24-hour time	18:05:19
r	%1$tr	12-hour time	06:05:19 pm
R	%1$tR	24-hour time, no seconds	18:05
Y	%1$tY	Four-digit year (with leading zeroes)	2004
y	%1$ty	Last two digits of the year (with leading zeroes)	04
C	%1$tC	First two digits of the year (with leading zeroes)	20
B	%1$tB	Full month name	February
b	%1$tb	Abbreviated month name	Feb
m	%1$tm	Two-digit month (with leading zeroes)	02
d	%1$td	Two-digit day (with leading zeroes)	03
e	%1$te	Two-digit day (without leading zeroes)	9
A	%1$tA	Full weekday name	Monday
a	%1$ta	Abbreviated weekday name	Mon
j	%1$tj	Three-digit day of the year (with leading zeroes)	069
H	%1$tH	Two-digit hour (with leading zeroes), between 00 and 23	18

Character	Specifier %1$t<c>	Description	Example
k	%1$tk	Two-digit hour (without leading zeroes), between 0 and 23	18
I	%1$tI	Two-digit hour (with leading zeroes), between 01 and 12	06
l	%1$tl	Two-digit hour (without leading zeroes), between 1 and 12	6
M	%1$tM	Two-digit minutes (with leading zeroes)	05
S	%1$tS	Two-digit seconds (with leading zeroes)	19
L	%1$tL	Three-digit milliseconds (with leading zeroes)	047
N	%1$tN	Nine-digit nanoseconds (with leading zeroes)	047000000
p	%1$tp	Lowercase morning or afternoon marker	pm
z	%1$tz	RFC 822 numeric offset from GMT	-0800
Z	%1$tZ	Time zone	PST
s	%1$ts	Seconds since 1970-01-01 00:00:00 GMT	1078884319
Q	%1$tQ	Milliseconds since 1970-01-01 00:00:00 GMT	1078884319047

Examples

`${DATE(%1$tF %1$tT.%1$tL)}` will give output as `"2024-08-30 12:00:05.878"`

`${DATE(%1$tF %1$tr)}` will give output as `"2024-08-30 12:00:05 PM"`

`${DATE(%1$tT.%1$tN)}` will give output as `"12:00:05.047000000"`

CSV/Delimited Events Filter Expressions

This Micro Integration provides various expressions for filtering data based on different conditions that can be used in the delimited eventFilterExpression Below is a categorized list of expressions you can use in your Java application.

1. Basic Comparisons

Expression	Description
`age > 30`	Matches rows where age is greater than 30
`salary < 50000`	Matches rows where salary is less than 50,000
`status == "active"`	Matches rows where the status is "active"
`age >= 18 && age <= 60`	Matches rows where age is between 18 and 60
`name == "Alice"`	Matches rows where the name is "Alice"
`isPremiumMember == true`	Matches rows where the user is a premium member

2. Logical Operators

Expression	Description
`age > 30 && salary > 50000`	Matches rows where age is greater than 30 and salary is above 50,000
`status == "active"		
status == "pending"`	Matches rows where status is "active" or "pending"
`!(age < 18)`	Matches rows where age is NOT less than 18
`(age > 25		
salary > 40000) && status == "active"`	Complex conditions with multiple logical operations

3. String Operations

Expression	Description
`name.startsWith("A")`	Matches rows where the name starts with "A"
`name.endsWith("son")`	Matches rows where the name ends with "son"
`name.contains("Smith")`	Matches rows where the name contains "Smith"
`name.length() > 5`	Matches rows where the name has more than 5 characters
`name.toUpperCase() == "JOHN"`	Matches rows where the uppercase version of name is "JOHN"
`name.toLowerCase().contains("admin")`	Case-insensitive check if the name contains "admin"

4. Numeric Operations

Expression	Description
`(salary * 1.1) > 55000`	Matches rows where a 10% salary increase exceeds 55,000
`(age + 5) > 30`	Matches rows where age + 5 is greater than 30
`age % 2 == 0`	Matches rows where age is even
`Math.sqrt(salary) > 200`	Uses Math functions (square root of salary is greater than 200)

5. Null and Empty Checks

Expression	Description
`name != null`	Matches rows where name is not null
`name == null`	Matches rows where name is null
`name != null && name != ""`	Matches rows where name is not null and not empty
`status.isEmpty()`	Matches rows where the status field is empty

6. Date Operations

Expression	Description
`registrationDate.after("2023-01-01")`	Matches rows where registration date is after Jan 1, 2023
`registrationDate.before("2022-12-31")`	Matches rows where registration date is before Dec 31, 2022
`registrationDate.year == 2024`	Matches rows where registration happened in 2024
`registrationDate.month == 5`	Matches rows where registration happened in May

7. List and Collection Filtering

Expression	Description
`roles.contains("admin")`	Matches rows where the roles list contains "admin"
`tags.size() > 3`	Matches rows where the tags list has more than 3 elements
`categories.contains("electronics") && price > 100`	Matches rows where categories include "electronics" and price is greater than 100

8. Using Column Numbers Instead of Headers

If headers are not available, use column indices instead.

Expression	Equivalent With Headers
`column3 > 30`	`age > 30`

Expression	Equivalent With Headers
`column4 == "active"`	`status == "active"`
`column2.startsWith("A")`	`name.startsWith("A")`
`column5 * 1.1 > 50000`	`salary * 1.1 > 50000`

9. Regex Matching

Expression	Description
`name ~= "^[A-Z].*"`	Matches names starting with an uppercase letter
`email ~= ".*@gmail.com$"`	Matches emails ending with `@gmail.com`

File Streaming Modes

The `fileFormat` property defines how file content is read and streamed as events. Supported modes include:

fileFormat	Description
1	CSV Line by Line File Streaming (Version 1 - Deprecated)
2	CSV - Custom Delimited Events File Streaming (Version 1 - Deprecated)
3	JSON File Streaming
4	XML File Streaming
5	Fixed Length File Streaming
11	CSV Line by Line File Streaming (Version 2 - Enhanced)
12	CSV - Custom Delimited Events File Streaming (Version 2 - Enhanced)

File Format 1: CSV Line by Line File Streaming (Version 1)

This mode supports **line-by-line streaming of delimited files (e.g., CSV)**, where each row is read as an event. This format is now **deprecated** and is maintained only for backward compatibility.

Events are parsed row-wise with optional transformation and filtering using a lightweight configuration.

Key Features

- Line-by-line streaming
- Supports CSV or any delimited format (e.g., pipe `|`, comma `,`, semicolon `;`)
- Column mapping using header row or explicit definition
- Event filtering with expression language
- Dynamic topic generation using column placeholders

- Column trimming and selective output

Example CSV File Content

```
1|123|John|'Doe'|25|OpenAI|Toronto|Canada|john.doe@example.com|1234567890
2|124|Jane|'Smith'|35|Google|New York|USA|jane.smith@example.com|0987654321
```

Configuration Properties

Field	Type	Allowed Values	Default	Description
`advanced.delimited.columnDelimiter`	String	Column Separator or Delimiter	","	Defines how the columns are separated in the file. Examples: ",", "|", "\t", "||" Must be provided when using `columnHeaderMap` or `useFirstEventAsHeader`.
`advanced.delimited.columnHeaderMap`	String	Delimited string	*(empty)*	Specifies column names when the header is not in the file. Required for JSON transformation and dynamic topic generation. Example: `"Index|Customer Id|First Name|Last Name|Age|Company|City|Country|Email|Phone"`
`advanced.delimited.ignoreFirstEventOfDelimitedFile`	Boolean	`true`, `false`	`false`	Skips the first row of the file. Useful when the file starts with a non-data row (e.g., metadata or comment).
`advanced.delimited.useFirstEventAsHeader`	Boolean	`true`, `false`	`false`	Uses the first row in the file as the header row. If enabled, `columnHeaderMap` is ignored.
`advanced.delimited.trimColumns`	Boolean	`true`, `false`	`false`	Trims whitespace around column values. Helpful for cleaner matching and filtering.
`advanced.delimited.quoteCharacter`	Char	Any single character	`"` (double quote)	Character used to quote column values. Example: `'John Doe'` or `"New York"`
`advanced.delimited.escapeCharacter`	Char	Any single character	`\0` (null)	Character used as an escape character
`advanced.delimited.eventFilterExpression`	String	Any valid expression	*(empty)*	Filters rows using expression logic. Example: `"Age < 31"` will allow only events where Age < 31.

Field	Type	Allowed Values	Default	Description
advanced.delimited.columnFilter	String	Delimited string	*(empty)*	Restricts the output JSON payload to specific columns. Example: "Index|First Name|Age|Country"
advanced.delimited.parsingFailBehavior	String	THROW, SKIP_ROW	THROW	THROW - throws error and stops processing more file data, SKIP_ROW skips the failed event and logs it
advanced.dynamicTopic	String	Topic template with placeholders	*(empty)*	Enables event publishing to dynamic Solace topics using column values. Example: "file/events/${column2}/${Country}"

Configuration Example 1 – Basic (with explicit headers)

```
file-events:
  source:
    advanced:
      fileFormat: 1
      delimited:
        columnDelimiter: "|"
        columnHeaderMap: "Index|Customer Id|First Name|Last
Name|Age|Company|City|Country|Email|Phone"
      dynamicTopic: file/events/${column2}/${Country}
```

Explanation: - The CSV rows are parsed using | as delimiter. - Explicit header names are provided for mapping columns. - Dynamic topic is generated using column2 and Country.

Configuration Example 2 – Use first row as header

```
file-events:
  source:
    advanced:
      fileFormat: 1
      delimited:
        columnDelimiter: "|"
        useFirstEventAsHeader: true
        trimColumns: true
      dynamicTopic: file/events/${Customer Id}/${Country}
```

Explanation: - The first row of the file is used as headers. - Trimming is enabled. - Dynamic topic uses named placeholders based on the header.

Configuration Example 3 – Filtering and selective columns

```
file-events:
  source:
    advanced:
      fileFormat: 1
      delimited:
        columnDelimiter: "|"
        columnHeaderMap: "Index|Customer Id|First Name|Last
Name|Age|Company|City|Country|Email|Phone"
        eventFilterExpression: "Age < 31"
        columnFilter: "Index|First Name|Age|Country"
        quoteCharacter: "'"
        escapeCharacter: '\\'
      dynamicTopic: file/events/${column2}/${Country}
```

Explanation: - Events are filtered to only include those with Age < 31. - Only a subset of columns is included in the final event. - Quote character is customized to `'`.

Sample Input Payload

Given row:

```
1|123|John|'Doe'|25|OpenAI|Toronto|Canada|john.doe@example.com|1234567890
```

And `columnFilter`: `"Index|First Name|Age|Country"`

The output JSON would be:

```
{
  "Index": "1",
  "First Name": "John",
  "Age": 25,
  "Country": "Canada"
}
```

Best Practices

- Always define `columnDelimiter` explicitly to avoid ambiguity.
- Prefer `useFirstEventAsHeader` when headers are present in the file.
- Avoid this mode for complex CSV parsing – migrate to `fileFormat: 11` or `12`.

- Validate `eventFilterExpression` syntax with sample data.
- Use `columnFilter` to reduce payload size and publish only required data.

File Format 2: CSV - Custom Delimited Events File Streaming (Version 1)

The configuration for File Format 2 and 1 works in the similar way except we can define a custom event delimiter inside the configuration as below:

Configuration Properties

Field	Type	Allowed Values	Default	Description
`advanced.delimited.eventDelimiter`	Char	any	'\n'	This setting would define a delimiter for separating events. If enabled, it would likely be used to distinguish different events within a single file.

Example File with multiple rows separate by "@"

Given row:

```
1|123|John|'Doe'|25|OpenAI|Toronto|Canada|john.doe@example.com|1234567890@2|124|Jane|'
Smith'|35|Google|New York|USA|jane.smith@example.com|0987654321
```

File Format 11: CSV Line by Line (Version 2 - Enhanced)

This mode is an **enhanced replacement for the deprecated fileFormat: 1**, providing better validation, column typing, and expression-based filtering.

It allows advanced control over how columns are parsed, validated, and included in the output. Suitable for high-integrity CSV parsing and structured event generation.

⚙️ Configuration Properties

Field	Type	Allowed Values	Default	Description
`advanced.delimited.columnDelimiter`	String	Column Separator or Delimiter	","	Defines how the columns are separated in the file. Examples: ",", "|", "\t", "||" Must be provided when using `columnHeaderMap` or `useFirstEventAsHeader`.
`advanced.delimited.columnHeaderMap`	String	Delimited string of headers	*(empty)*	Explicit header list if headers are not in the file.
`advanced.delimited.ignoreFirstEventOfDelimitedFile`	Boolean	`true`, `false`	`false`	If true, skips the first row entirely.
`advanced.delimited.useFirstEventAsHeader`	Boolean	`true`, `false`	`false`	Treats the first row of the file as the header row.
`advanced.delimited.trimColumns`	Boolean	`true`, `false`	`false`	Trims leading and trailing spaces of all column values.
`advanced.delimited.quoteCharacter`	Char	Any character	`"` (double quote)	Column values can be quoted. This sets the quote character.
`advanced.delimited.escapeCharacter`	Char	Any single character	`\0` (null)	Character used as an escape character
`advanced.delimited.eventFilterExpression`	String	Expression language	*(empty)*	Allows rows to be included or excluded based on values (e.g., `Age < 31`).
`advanced.delimited.restrictToColumnSpecs`	Boolean	`true`, `false`	`false`	If true, only includes columns defined in `columnSpecs` in output.
`advanced.delimited.columnSpecs`	List	Defined column constraints	*(empty)*	Optional. Allows regex, defaulting, coercion, and validation per column.

Example Configuration – Enhanced CSV Parsing

```
file-events:
  source:
```

```
    advanced:
      fileFormat: 11
      dynamicTopic: solace/fc/source/demo/dynamic/${Country}/${City}/${param-
3}/${column5}
      delimited:
        columnDelimiter: "|"
        columnHeaderMap: Index,Customer Id,First Name,Last
Name,Age,Company,City,Country,Phone 1,Phone 2,Email,Subscription Date,Website
        ignoreFirstEventOfDelimitedFile: false
        useFirstEventAsHeader: true
        trimColumns: true
        quoteCharacter: '"'
        escapeCharacter: '\\'
        strictValidation: true
        eventFilterExpression: "Age < 31"
        restrictToColumnSpecs: false
        columnSpecs:
          - name: "First Name"
            type: STRING
            regex: "[A-Za-z]+"
            onRegexFail: SKIP_COLUMN
            defaultValue: "Unknown"
            required: OPTIONAL
          - name: "Customer Id"
            type: STRING
            required: REQUIRED
          - name: Age
            type: INT
            regex: "\\d+"
            onRegexFail: COERCE
            required: REQUIRED
          - name: Company
            type: STRING
            required: OPTIONAL
          - name: Country
            type: STRING
            required: OPTIONAL
          - name: City
            type: STRING
            required: OPTIONAL
```

Explanation: - Strongly typed configuration with validation rules and defaults - Topic is dynamically generated using column values - Filters out rows where Age >= 31 - Allows optional or required columns, validation failure handling

File Format 12: CSV - Custom Delimited Events File Streaming (Version 2 - Enhanced)

The configuration for File Format 2 and 1 works in the similar way except we can define a custom event delimiter inside the configuration as below:

Configuration Properties

Field	Type	Allowed Values	Default	Description
advanced.delimited.eventDelimiter	Char	any	'\n'	This setting would define a delimiter for separating events. If enabled, it would likely be used to distinguish different events within a single file.

Example File with multiple rows separate by "@"

Given row:

```
1|123|John|'Doe'|25|OpenAI|Toronto|Canada|john.doe@example.com|1234567890@2|124|Jane|'
Smith'|35|Google|New York|USA|jane.smith@example.com|0987654321
```

🔄 Differences Between FileFormat 1/2 and 11/12

Feature	fileFormat: 1 & 2	fileFormat: 11 & 12
Validation	Minimal	Strict, configurable per column
Deprecation Status	Deprecated	Recommended
Header Handling	Yes	Yes + advanced mapping
Filtering	Basic	Advanced (typed, expression-based)
Column Trimming	Optional	Optional
Regex Validation	No	Yes
Column Specs	No	Yes
Quote Character	Yes	Yes
Event Filtering	Basic	Enhanced
Output Restriction	Yes (`columnFilter`)	Yes (`restrictToColumnSpecs`)
Event Delimiter	fileFormat: 2 only	fileFormat: 12 only

Column Specs Configuration Guide

This document explains how to configure the `columnSpecs` section used in `fileFormat: 5`, `fileFormat: 11` or `fileFormat: 12`.

Each item in `columnSpecs` defines parsing, validation, coercion, and output behavior for a specific column in your delimited file.

Configurations Explained

Setting	What It Does	Example	Mandatory	Default
`name`	Column name as it appears in the header row or `columnHeaderMap`. Used to match the column.	`"Name"`	MANDATORY	null
`length`	Optional. Maximum number of characters for this column. Used for fixed-length validation.	`10`	OPTIONAL	`0`
`type`	Declares the data type for parsing: - `STRING`: text - `INT`: whole numbers - `DOUBLE`: decimal numbers - `BOOLEAN`: true/false - `DATE`: date format	`STRING`	OPTIONAL	`STRING`
`inputDateFormat`	For `DATE` type. Format used in the incoming data file.	`"dd/MM/yyyy"` (example: 25/12/1990)	OPTIONAL	null
`outputDateFormat`	For `DATE` type. Format to be used in the output event.	`"dd-MM-yyyy"` (example: 25-12-1990)	OPTIONAL	null
`regex`	Regex pattern that the value must match. Can be used to enforce formats like alphabets only, numeric only, etc.	`"[A-Za-z]+"`	OPTIONAL	null

Setting	What It Does	Example	Mandatory	Default
`onRegexFail`	Action to take when `regex` validation fails: - `THROW`: raises an exception and fails processing - `SKIP_COLUMN`: sets value to null - `SKIP_ROW`: discards the row - `LOG`: logs the error and continues - `DEFAULT_VALUE`: uses the value from `defaultValue` - `COERCE`: attempts to fix the value (e.g., remove non-matching characters)	`"DEFAULT_VALUE"`	OPTIONAL	`THROW`
`defaultValue`	Value to use if validation fails and `onRegexFail` is set to `DEFAULT_VALUE`.	`"Unknown"`	OPTIONAL	null
`required`	Defines how the column is handled in the output: - `REQUIRED`: must appear in output and must not be null - `OPTIONAL`: may appear in output; null values allowed - `DROP`: column is processed for logic but excluded from output event	`"REQUIRED"`	OPTIONAL	`REQUIRED`

String Format (Simple)

Use this for basic setups. Each column is a single line with values separated by commas.

```
columnSpecs:
  - "Name, 10, STRING"
  - "Age, 3, INT"
  - "Balance, 10, DOUBLE"
  - "Active, 5, BOOLEAN"
  - "Birthday, 10, DATE, dd/MM/yyyy, dd-MM-yyyy"
```

What Each Part Means

1. **Name**: What to call the column (e.g., `"Name"`).
2. **Length**: How many characters the column takes up (e.g., `10`).
3. **Type**: The data type (`STRING`, `INT`, `DOUBLE`, `BOOLEAN`, `DATE`).
4. **Input Date Format** (optional, for `DATE` only): How dates are written in the file (e.g., `"dd/MM/yyyy"`).

5. **Output Date Format** (optional, for `DATE` only): How dates should look in the output (e.g., `"dd-MM-yyyy"`).

and so on.

Example File

John 25 123.45 true 25/12/1990

- `"Name, 10, STRING"`: Takes `John ` (10 characters).
- `"Age, 3, INT"`: Takes `25 ` (3 characters).
- Total row length: 38 (if `eventFixedLength: 38`).

Notes: - Use this format for simple files without complex rules. - All columns are required in the output by default, and errors stop processing if `strictValidation` is `true`.

Map Format (Detailed)

Use this for more control, like adding validation rules or excluding columns from the output. Each column is a block with named settings.

```
columnSpecs:
  - name: Name
    length: 10
    type: STRING
    regex: "[A-Za-z]+"
    onRegexFail: DEFAULT_VALUE
    defaultValue: "Unknown"
    required: REQUIRED
  - name: Age
    length: 3
    type: INT
    regex: "\\d+"
    onRegexFail: COERCE
    required: OPTIONAL
  - name: Balance
    length: 10
    type: DOUBLE
    regex: "[0-9]*\\.?[0-9]*(?:[eE][-+]?[0-9]+)?"
    onRegexFail: SKIP_ROW
    required: REQUIRED
  - name: Active
    length: 5
    type: BOOLEAN
    required: OPTIONAL
  - name: Birthday
    length: 10
    type: DATE
    inputDateFormat: "dd/MM/yyyy"
    outputDateFormat: "dd-MM-yyyy"
```

```
    required: DROP
```

Example File and Output

John 25 123.45 true 25/12/1990 Jane 2x 678.90 false 01/01/2000

- **Output (JSON)**:
- Row 1: `{"Name": "John", "Age": 25, "Balance": 123.45, "Active": true}` (`Birthday` dropped).
- Row 2: `{"Name": "Jane", "Age": 2, "Balance": 678.90, "Active": false}` (`COERCE` fixes `2x` to `2`).

Tips: - Use `regex` to enforce data quality (e.g., `"\\d+"` for digits only). - Set `required: DROP` for columns you need to process but not include in the output. - Combine `onRegexFail` and `defaultValue` for fallback values.

Fixed-Length File Processing Configuration Guide

This guide explains how to configure the application to process fixed-length files. Fixed-length files have data organized in rows where each column has a specific width, and this configuration tells the application how to read, validate, and format that data into output events (e.g., JSON, CSV). You'll use a YAML file to set up the structure, columns, and output preferences.

Overview

The configuration lives under the `file-events.source.advanced.fixedLengthFile` section of your YAML file. It controls: - How rows are split (fixed length or separator-based). - How columns are defined and validated. - How the output is formatted (e.g., JSON, XML).

You can define columns in two ways: a simple **string format** (comma-separated) or a detailed **map format** (key-value pairs). The map format offers more options, like validation rules and output control.

Sample Configuration

Here's a complete example to get you started:

```yaml
file-events:
  source:
    advanced:
      fixedLengthFile:
        isRowSeparatorPresent: false
        eventFixedLength: 51
        rowSeparator: "|"
        strictValidation: true
        columnSpecs:
          - "Name, 10, STRING"
          - "Age, 3, INT"
          - "Balance, 10, DOUBLE"
          - "Active, 5, BOOLEAN"
          - "Birthday, 10, DATE, dd/MM/yyyy, dd-MM-yyyy"
```

Let's break down each part.

File Structure Settings

These settings define how the file is split into rows.

Setting	What It Does	Example
`isRowSeparatorPresent`	Set to `true` if rows end with a separator (like `	`). Set to `false` if rows are a fixed length.
`false` (rows are 51 characters long)	`eventFixedLength`	Number of characters per row (excluding separator). Required if `isRowSeparatorPresent` is `false`.

Setting	What It Does	Example
51	rowSeparator	The character that separates rows (used only if isRowSeparatorPresent is true).
"|"	strictValidation	If true, stops processing on errors (e.g., bad data). If false, logs errors and continues with missing values.

Tips: - Use `isRowSeparatorPresent: false` for most fixed-length files and set `eventFixedLength` to match your row size. - Set `strictValidation: false` if you want to process files with occasional bad data without stopping.

Column Definitions (`columnSpecs`)

The `columnSpecs` section lists each column in your file—its name, width, type, and optional rules. You can use two formats:

Refer to Column Specifications Configuration for more details.

Filtering Rows (Optional)

Add an `eventFilterExpression` to skip rows based on conditions:

```
eventFilterExpression: "Age < 30 && Balance > 100"
```

- Example: Only processes rows where `Age` is under 30 and `Balance` exceeds 100.
- Use column names from `columnSpecs` and operators like `<`, `>`, `==`, `&&`.

Putting It All Together

Example File

John 25 123.45 true 25/12/1990 Jane 2x 678.90 false 01/01/2000

String Config

```
fixedLengthFile:
  eventFixedLength: 38
  strictValidation: true
  columnSpecs:
    - "Name, 10, STRING"
    - "Age, 3, INT"
    - "Balance, 10, DOUBLE"
```

- Row 1: `{"Name": "John", "Age": 25, "Balance": 123.45}`
- Row 2: Stops with an error (`"2x"` isn't an `INT`).

Map Config

```
file-events:
  source:
    advanced:
      fixedLengthFile:
        eventFixedLength: 38
        strictValidation: true
        columnSpecs:
          - name: Name
            length: 10
            type: STRING
            regex: "[A-Za-z]+"
            onRegexFail: DEFAULT_VALUE
            defaultValue: "Unknown"
            required: REQUIRED
          - name: Age
            length: 3
            type: INT
            regex: "\\d+"
            onRegexFail: COERCE
            required: OPTIONAL
```

- Row 1: `{"Name": "John", "Age": 25}`
- Row 2: `{"Name": "Jane", "Age": 2}` (`COERCE` fixes `"2x"`).

Troubleshooting

- **Row Length Errors**: Check `eventFixedLength` matches your file's row size (sum of column lengths).
- **Validation Failures**: Set `strictValidation: false` to log issues instead of stopping, or use `onRegexFail: COERCE`.
- **Missing Columns**: Ensure `required: REQUIRED` columns have valid data, or adjust `regex` rules.

File Format 6: Lines as Events

This mode allows the streaming of individual line as events from the configured source files

Additionally, a line separator (default: newline) can be configured if newline is not the event delimiter in the file

Configuration Properties

Field	Type	Allowed Values	Default	Description
advanced.line.eventDelimiter	Char	any	'\n'	This setting would define a delimiter for separating events. If configured, it would be used to distinguish different events within a single file.
`advanced.line.rootObjectName`	String	any	*(empty)*	If defined the event can be enclosed inside the rootObjectName, primarily used with output formats json or xml.

Example Configuration

```
file-events:
  source:
    advanced:
      fileFormat: 6
      line:
        eventDelimiter: "|"
        rootObjectName: "data"
```

Output Formatting (`output`)

This transformation options allows controlling how the output events will be streamed in case of the below file formats only -

fileFormat	Description
1	CSV Line by Line File Streaming (Version 1 - Deprecated)
2	CSV - Custom Delimited Events File Streaming (Version 1 - Deprecated)
5	Fixed Length File Streaming
11	CSV Line by Line File Streaming (Version 2 - Enhanced)
12	CSV - Custom Delimited Events File Streaming (Version 2 - Enhanced)

Controls how the processed data is formatted (e.g., JSON, CSV, XML, or text).

Setting	What It Does	Example	Default
`format`	Output format type. Valid values: `json`, `csv`, `xml`, `text`, `raw`	`"json"`	`"raw"`
`encloseInRootObjectName`	If the output needs to be enclosed in a new root object - Name of the new root element. Typically used in JSON or XML.	`"root"`	`"empty"`
`prettyPrintJson`	Adds line breaks and indentation for JSON output.	`true`	`false`
`ignoreNulls`	If `true`, omits fields with null values in JSON, XML, text.	`true`	`false`
`csvDelimiter`	Character used to separate fields in CSV.	`","`	`","`
`csvQuoteChar`	Character used to wrap field values in CSV.	`"\""`	`"\""`

Setting	What It Does	Example	Default
csvQuoteMode	Quote Mode used to wrap field values in CSV. Values allowed are as follows - ALL : Quotes all fields, regardless of content. - ALL_NON_NULL : Quotes all non-null fields. - MINIMAL : Only quotes fields when necessary, i.e., if they contain: The delimiter (e.g., comma) A quote character A newline character - NON_NUMERIC : Quotes only non-numeric fields (e.g., Strings), leaving numbers unquoted. - NONE : No fields are quoted, even if they contain special characters.	ALL	MINIMAL
csvRecordSeparator	Record Separator String Value. Default is "" which uses System.lineSeparator	"@"	""
xmlRootElement	Name of the root element in XML.	"customer"	"root"
prettyPrintXml	If true, indents XML output for readability.	true	false
includeXmlDeclaration	Adds XML declaration header (<?xml ⋯ ?>) if true.	true	true
xmlVersion	XML version to declare.	"1.0"	"1.0"
xmlEncoding	Encoding to declare in XML.	"UTF-8"	"UTF-8"
xmlStandalone	Sets standalone declaration in XML (yes or no).	"no"	"no"
xmlMethod	Specifies XML output method (xml, html, text), if applicable.	"xml"	null
textValuesOnly	If true, only outputs values (no field names) in text format.	false	false
textValueSeparator	Separator between field names and values in text format.	": "	":"
textLineSeparator	Line separator for text format.	"\n"	"\n"

Setting	What It Does	Example	Default
`fieldOrder`	Ordered list of fields to output. Unlisted fields come after, unless `strictFieldOrder` is `true`.	`["Name", "Balance", "Birthday"]`	`[]`
`strictFieldOrder`	If `true`, only fields listed in `fieldOrder` are included.	`false`	`false`

Example YAML

```yaml
file-events:
  source:
    advanced:
      output:
        outputFormat: "json"
        prettyPrintJson: false
        ignoreNulls: true
        csvDelimiter: ","
        csvQuoteChar: "\""
        csvQuoteMode: MINIMAL
        csvRecordSeparator: ""
        xmlRootElement: "customer"
        prettyPrintXml: true
        includeXmlDeclaration: true
        xmlVersion: "1.0"
        xmlEncoding: "UTF-8"
        xmlStandalone: "no"
        xmlMethod: "xml"
        textValuesOnly: false
        textValueSeparator: ": "
        textLineSeparator: "\n"
        fieldOrder:
          - "Name"
          - "Balance"
          - "Birthday"
        strictFieldOrder: false
```

Example Configurations and Outputs

Below are formatted examples showing how the output would look for different `outputFormat` settings using the following sample input data:

```
Name: Alice
Balance: 1234.56
Birthday: 1990-12-25
Notes: null
```

JSON Output

YAML Configuration:

```
outputFormat: "json"
prettyPrintJson: true
ignoreNulls: false
```

Output:

```
{
  "Name": "Alice",
  "Balance": 1234.56,
  "Birthday": "1990-12-25",
  "Notes": null
}
```

With `ignoreNulls: true`:

```
{
  "Name": "Alice",
  "Balance": 1234.56,
  "Birthday": "1990-12-25"
}
```

CSV Output

YAML Configuration:

```
outputFormat: "csv"
csvDelimiter: ","
csvQuoteChar: "\""
csvQuoteMode: ALL
fieldOrder:
  - "Name"
  - "Balance"
  - "Birthday"
  - "Notes"
```

Output:

```
"Name","Balance","Birthday","Notes"
"Alice","1234.56","1990-12-25",""
```

XML Output

YAML Configuration:

```
outputFormat: "xml"
xmlRootElement: "customer"
prettyPrintXml: true
includeXmlDeclaration: true
```

Output:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<customer>
  <Name>Alice</Name>
  <Balance>1234.56</Balance>
  <Birthday>1990-12-25</Birthday>
  <Notes/>
</customer>
```

Text Output

YAML Configuration:

```
outputFormat: "text"
textValueSeparator: ": "
textLineSeparator: "\n"
textValuesOnly: false
```

Output:

```
Name: Alice
Balance: 1234.56
Birthday: 1990-12-25
Notes: null
```

With `textValuesOnly: true`:

```
Alice
1234.56
1990-12-25
null
```

Raw Output (Default)

If `outputFormat` is not specified or explicitly set to "raw", the application may output the original file

or record as-is, without formatting.

Extended Example Configurations and Outputs

1. JSON Output Example

YAML Configuration:

```yaml
file-events:
  source:
    advanced:
      output:
        outputFormat: "json"
        prettyPrintJson: true
        ignoreNulls: true
        fieldOrder:
          - "Name"
          - "Balance"
          - "Birthday"
        strictFieldOrder: false
```

Expected JSON Output:

```
{
  "Name": "Alice",
  "Balance": 1234.56,
  "Birthday": "1990-12-25"
}
```

- `prettyPrintJson: true` adds indentation and line breaks.
- `ignoreNulls: true` removes fields with `null` values.

2. CSV Output Example

YAML Configuration:

```yaml
file-events:
  source:
    advanced:
      output:
        outputFormat: "csv"
        csvDelimiter: ","
        csvQuoteChar: "\""
        csvQuoteMode: ALL
        fieldOrder:
          - "Name"
          - "Balance"
          - "Birthday"
```

```
        strictFieldOrder: true
```

Expected CSV Output:

```
"Name","Balance","Birthday"
"Alice","1234.56","1990-12-25"
```

- `csvDelimiter: ","` separates values with commas.
- `csvQuoteChar: "\""` wraps values in double quotes.

3. XML Output Example

YAML Configuration:

```
file-events:
  source:
    advanced:
      output:
        outputFormat: "xml"
        xmlRootElement: "customer"
        prettyPrintXml: true
        includeXmlDeclaration: true
        xmlVersion: "1.0"
        xmlEncoding: "UTF-8"
        xmlStandalone: "no"
```

Expected XML Output:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<customer>
  <Name>Alice</Name>
  <Balance>1234.56</Balance>
  <Birthday>1990-12-25</Birthday>
</customer>
```

- `xmlRootElement: "customer"` sets the root element to `<customer>`.
- `prettyPrintXml: true` adds indentation for readability.

4. Text Output Example

YAML Configuration:

```
file-events:
  source:
    advanced:
      output:
        outputFormat: "text"
        textValuesOnly: false
```

```
        textValueSeparator: ": "
        textLineSeparator: "\n"
```

Expected Text Output:

```
Name: Alice
Balance: 1234.56
Birthday: 1990-12-25
```

- `textValuesOnly: false` means field names are included with their values.
- `textValueSeparator: ": "` uses `: ` to separate field names from values.
- `textLineSeparator: "\n"` adds a newline after each field.

5. Raw Output Example

YAML Configuration:

```
file-events:
  source:
    advanced:
      output:
        outputFormat: "raw"
```

Expected Raw Output:

```
Name: Alice
Balance: 1234.56
Birthday: 1990-12-25
Notes: null
```

- The raw output is just the original input data with no formatting.

File Events Sink (Events to File)

Configuring Output Destination

The configuration property `spring.cloud.stream.bindings.output-0.destination` defines the output destination path for the sink micro integration (MI) instance.

This property supports dynamic file naming using the Dynamic Expressions resolver, allowing the output path and filename to be constructed at runtime based on context such as dates, topic components, message headers, and more.

For more details on dynamic expressions, refer to the Dynamic Expressions section.

Example Usage Dynamic Expressions

Set the property to a path template that includes dynamic expressions enclosed in ${...}. At runtime, the expressions are resolved and substituted into the path, creating unique output files.

Common expressions useful for output destinations include:

Expression Category	Description
`DATE`	Inserts a formatted timestamp, with optional offsets and formatting. Example: ${DATE(%1$tY%1$tm%1$td)} resolves to 20251015.
`TOPIC`	Extracts components from the topic hierarchy by index. Example: ${TOPIC(2)} for the third segment (0-based indexing); ${TOPIC(-2)} for the second-to-last segment.
`HEADER`	References values from message headers. Example: ${HEADER(header)} resolves to the value of the specified header.
`FILE`	Extracts metadata from the source file, such as name or path. Example: ${FILE(FBASE)} for the base filename without extension.
`UMAP`	References user-defined properties. Example: ${UMAP(region)} for a property named "region".
`AUTO_INCREMENT`	Replaces the dynamic variable expression with the auto increment integer value associated with the file count or the event count depending on the field it is used inside. This count is maintained by the application in the application state.
`Level 2 Operators`	Post-processes the result of a Level 1 expression, e.g., #UPPER(${TOPIC(2)}) to uppercase a topic segment.

The resolved path must be a valid absolute or relative file path, ending with a filename (optionally dynamic).

For more details on dynamic expressions, refer to the Dynamic Expressions section.

Examples

Basic Date-Based Naming

```
spring.cloud.stream.bindings.output-0.destination = /output/daily-report-
${DATE(%1$tY%1$tm%1$td)}.txt
```

This generates a file like /output/daily-report-20251015.txt based on the current date.

Topic and Header Integration

```
spring.cloud.stream.bindings.output-0.destination = /output/logs/${TOPIC(1)}-
${HEADER(source-id)}-${DATE(%1$tH%1$tM)}.log
```

For a topic like acme/sales/order/created and header source-id: app1, this resolves to /output/logs/acme-app1-1430.log.

Advanced with File Metadata and Increment

```
spring.cloud.stream.bindings.output-0.destination = /output/files/file-
${DATE(%1$tY%1$tm%1$td)}-${TOPIC(2)}-${TOPIC(-2)}-${HEADER(header)}-
${AUTO_INCREMENT}.txt
```

This creates a file like /output/files/file-20251015-sales-created-user123-1.txt, where ${AUTO_INCREMENT} appends a sequential number for uniqueness (e.g., 1, 2). Note: ${AUTO_INCREMENT} is a special expression for auto-incrementing counters per run or batch.

For more complex scenarios, combine expressions with Level 2 operators, such as #UPPER(${TOPIC(2)}) to enforce case conventions in filenames.

For more details on dynamic expressions, refer to the Dynamic Expressions section.

Dynamic Variable expressions Quick guide

These expressions can be used inside the following configuration values when configuring sink MI.

1. Output binding destination. (Destination File Path) `spring.cloud.stream.bindings.output-<>.destination`
2. Advanced options `advanced.emptyEventsSubstitute` and `advanced.uniqueEventIdentifyingHeader`
3. File Pre Processors defined under configuration pre_process.<> - `prependToFile`, `prependToFirstEvent`, `prependToEvents`
4. File Post Processors defined under configuration post_process.<> - `appendToEvents`

For more details on dynamic expressions, refer to the Dynamic Expressions section.

Dynamic Variable	Expression	Description
UMAP	${UMAP(<key_name>)}	Replaces the dynamic variable expression with the User Property map Value associated with the key_name defined in the expression. For example ${UMAP(UNIQUE_FILE_ID))} will replace this expression with the value of the User Property Map key UNIQUE_FILE_ID
HEADER	${HEADER(<key_name>)}	Replaces the dynamic variable expression with the Header Value associated with the key_name defined in the expression For example ${HEADER(UNIQUE_FILE_ID))} will replace this expression with the value of the Header key UNIQUE_FILE_ID
TOPIC	${TOPIC(<level>)}	Replaces the dynamic variable expression with the Source topic level associated with the level defined in the expression For example if source topic is solace/acme/orders/books/canada/init ${TOPIC(1)} will replace the expression with value solace ${TOPIC(3)} will replace the expression with value orders ${TOPIC(-2)} will replace the expression with second level value from the end i.e. canada ${TOPIC(0)} will replace the expression with entire source topic value solace/acme/orders/books/canada/init
DATE	${DATE(<specifier>)}	Replaces the dynamic variable expression with the Date specifier value associated with the specifier defined in the expression For example, ${DATE%1$tY%1$tm%1$td} will replace the expression with value "YYYYMMDD" i.e. "20240101"

Dynamic Variable	Expression	Description
AUTO_INCREMENT	${AUTO_INCREMENT}	Replaces the dynamic variable expression with the auto increment integer value associated with the file count or the event count depending on the field it is used inside. This count is maintained by the application in the application state.

File Events Sink (Events to File) Configuration Options

These configuration options are all prefixed by `file-events.sink.`:

General Configuration

Config Option	Type	Valid Values	Default Value	Description
`general.miID`	`string`	`any`	`empty`	Unique ID for the connector instance
`general.storeBackupStateRemotely`	`boolean`	`true,false`	`false`	When set to true, Sink connector will create the backup state file on the remote Channel (GCS, SFTP, FTP, FTPs) configured
`general.state_backup_path`	`string`	`any`	`empty`	Absolute file location to store Sink Connector's checkpoint information. The file should end in .cfg format(Ex:<path>/sink_connector_state_backup.cfg. The file will be created by connector if it doesn't exist.

Advanced Configuration

Config Option	Type	Valid Values	Default Value	Description
`advanced.maxEventsPerFile`	`int`	`0, 1, >1`	`0`	Default value 0 (unlimited). Maximum events allowed to be written in a file. To respect this, destination file name must include dynamic fields such as ${AUTO_INCREMENT} or any header parameter for successful rollover to a new file. If not configured properly, same file will be appended as a result.

Config Option	Type	Valid Values	Default Value	Description
advanced.maxFileSize	int	0, >0	0	Default value 0 (unlimited). Maximum file size in bytes allowed. To respect this, destination file name must include dynamic fields such as ${AUTO_INCREMENT} or any header parameter for successful rollover to a new file. If not configured properly, same file will be appended as a result. Irrespective of the maxFileSize defined, atleast 1 event will always be written to a file.
advanced.checkDynamicFileNameChangesEverytime	boolean	true, false	false	Default value false. If set to true, on every new event will check if file name changes are required. For example if a date is present in the file name. Or a Header which is expected to change often. ${AUTO_INCREMENT} will only take affect if there are other dynamic variables in the file name which are expected to change. ${AUTO_INCREMENT} will not change the name of the file in case checkDynamicFileNameChangesEverytime is set to true

Config Option	Type	Valid Values	Default Value	Description
advanced.escapeCharsRegex	string	any	empty	Regex to escape characters present inside the dynamic variable. Works if variable categories are used as ${UMAP_E(<>)}, ${TOPIC_E(<>)}, ${DATE_E(<>)}, ${HEADER_E(<>)}
advanced.escapeCharsReplacement	string	any	"@"	Default Value "@". Only works when escapeCharsRegex is configured
advanced.ignoreEmptyEvents	boolean	true, false	true	Default value true. If set to false, need to provide value for emptyEventsSubstitute. If no value for emptyEventsSubstitute is defined, message will be acked as a failure.
advanced.emptyEventsSubstitute	string	any	empty	String value to be replaced in case of empty events. Doesn't work if ignoreEmptyEvents is set to true OR if empty event is a file close trigger event Dynamic Variable expression can be used with this option. You can define multiple expressions inside the string value of this configuration For example "${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT}" The ${AUTO_INCREMENT} would give the event count associated with the current destination file.

Config Option	Type	Valid Values	Default Value	Description
advanced.preventDuplicateEvents	boolean	true, false	false	Default value false. If set to true duplicate events will be ignored. Max events in Duplicate check cache 260.
advanced.preventDuplicateOnlyForRedeliveredEvents	boolean	true, false	true	Default value true. Used only when preventDuplicateEvents set to true. If set to true duplicate key will only be checked for Redelivered events. If false, will be checked for all events.

Config Option	Type	Valid Values	Default Value	Description
advanced.uniqueEventIdentifyingHeader	string	any	empty	Used only when preventDuplicateEvents set to true. If value is set, will be treated as duplicate check key. If not set then complete event payload will be set as Duplicate key. You can either add the header name directly like uniqueEventIdentifyingHeader: "PROPERTY1". OR Can also add a combination of headers and merge the values using the Dynamic Variable Replacement like "${HEADER(PROPERTY1)}-${HEADER(PROPERTY2)}" You can add more variables to make the identifier value unique. Chose from ${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT} The ${AUTO_INCREMENT} would give the event count associated with the current destination file.
advanced.closeFileTriggerHeaderName	string	any	empty	If value is set, File will be closed if this header field exists in the event. Works in combination with closeFileTriggerHeaderValue

Config Option	Type	Valid Values	Default Value	Description
advanced.closeFileTriggerHeaderValue	string	any	empty	Default value "". Works if value of closeFileTriggerHeaderName is set. File will be closed if set to "*" i.e. all values OR specific value configured matches the exact string value with the event header
advanced.ignorePayloadOfCloseTriggerEvent	boolean	true, false	false	Default Value false. If set to true any payload present in the file closing trigger event will not be written to file. Works in combination with closeFileTriggerHeaderName and closeFileTriggerHeaderValue
advanced.ignoreWritingPayloadOfAllEvents	boolean	true, false	false	Default Value false. If set to true any payload present in the event will not be written to file. Will make sense to use if only headers or dynamic parameters need to be written to the destination file using prependToEvents or appendToEvents options.
advanced.alwaysVerifyIfFileExists	boolean	true, false	false	Default Value false. If set to true, the MI will check if the previously opened destination file exists before writing any new event to the file. If the file doesn't exist, it will either throw an exception or if recreateFileIfNotExists is set to true it will recreate the file.

Config Option	Type	Valid Values	Default Value	Description
advanced.recreateFileIfNotExists	boolean	true, false	false	Default Value false. If set to true, MI will recreate the destination file if it was removed or moved and doesn't exist. Only works if alwaysVerifyIfFileExists is set to true.
advanced.alwaysAddLineSeparator	boolean	true, false	true	Default Value true. Sink MI will add a line separator at the end of each event by default

Event Pre Processor Configuration

Config Option	Type	Valid Values	Default Value	Description
pre_process.prependToFile	string	any	empty	If value is set, Content of this configuration will be added to the beginning of the file. Dynamic Variable expression can be used with this option. You can define multiple expressions inside the string value of this configuration For example "${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT}" The ${AUTO_INCREMENT} would give the file count associated with the current destination file.

Config Option	Type	Valid Values	Default Value	Description
pre_process.prependToFirstEvent	string	any	empty	If value is set, the content of this configuration only gets added before the first event. Dynamic Variable expression can be used with this option. You can define multiple expressions inside the string value of this configuration For example “${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT}” The ${AUTO_INCREMENT} would give the event count associated with the current destination file. Helps to manage creation of specific file formats such as JSON.

Config Option	Type	Valid Values	Default Value	Description
pre_process.prependToEvents	string	any	empty	If value is set, the content of this configuration gets added before all event. Would be ignored for first event if prependToFirstEvent parameter is configured. Dynamic Variable expression can be used with this option. You can define multiple expressions inside the string value of this configuration For example “${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT}” The ${AUTO_INCREMENT} would give the event count associated with the current destination file.

Event Post Processor Configuration

Config Option	Type	Valid Values	Default Value	Description
post_process.appendToFile	string	any	empty	If value is set, Content of this configuration will be added to the end of the file.

Config Option	Type	Valid Values	Default Value	Description
post_process.appendToEvents	string	any	empty	If value is set, the content of this configuration gets added to the end of all events. Dynamic Variable expression can be used with this option. You can define multiple expressions inside the string value of this configuration For example “${UMAP(<key_name>)}, ${HEADER(<key_name>)}, ${TOPIC(<level>)}, ${DATE(<specifier>)}, ${AUTO_INCREMENT}” The ${AUTO_INCREMENT} would give the event count associated with the current destination file.

Example Configurations for Quick Setup

Always configure destination to unique names using ${AUTO_INCREMENT} or some other unique value depending on the business requirement. Existing files will be appended.

Single Event per File

```
spring:
  application:
    name: File Events (Sink)
  cloud:
    stream:
      bindings:
        input-0:
          destination: q.file.events.mesh.queue
          binder: solace
        output-0:
          destination: /output/files/file-${DATE(%1$tY%1$tm%1$td)}-
${AUTO_INCREMENT}.txt
          binder: file-events
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 1
```

No Limits for Event count per file

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 0
```

Max Size per file

Maximum size in bytes

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
```

```
    advanced:
      maxFileSize: 2048
```

No Size limits per file

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxFileSize: 0
```

Event Count and File Size limits

New events will be appended to file until one of the conditions is reached.

Note - Output destination file names should resolve to unique files

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 10
      maxFileSize: 4096
```

Event Pre Processors

The `pre_process` sections allow you to add prefixes to files and events during processing. Useful when generating json, xml or log files

These support dynamic expressions; refer to the Dynamic Expressions section for details.

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 0
      maxFileSize: 0
    pre_process:
      prependToFile: "START" #Adds a String to the start of file
      prependToFirstEvent: "${AUTO_INCREMENT},${DATE(%1$tc)}," #Adds a String before
writing the first event. Supports Dynamic Expressions
```

```
      prependToEvents: "${AUTO_INCREMENT},${DATE(%1$tc)}," #Adds a String before
writing the event. Supports Dynamic Expressions
```

Event Post Processors

The `post_process` sections allow you to add suffixes to files and events during processing. Useful when generating json, xml or log files

These support dynamic expressions; refer to the Dynamic Expressions section for details.

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 0
      maxFileSize: 0
    post_process:
      appendToFile: "This is file Close" #Appends to the end of each file
      appendToEvents: ",END" #Appends to every event. Supports Dynamic Expressions
```

Generating XML Output File

The `pre_process` and `post_process` sections allow you to add prefixes and suffixes to files and events during processing.

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 0
      maxFileSize: 0
    pre_process:
      prependToFile: "<xml>"
      prependToEvents: "<event>"
    post_process:
      appendToEvents: "</event>"
      appendToFile: "</xml>"
```

Generating JSON Output File

The `pre_process` and `post_process` sections allow you to add prefixes and suffixes to files and events during processing.

```
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: backup_state.cfg
    advanced:
      maxEventsPerFile: 0
      maxFileSize: 0
    pre_process:
      prependToFile: "{["
      prependToFirstEvent: "{"
      prependToEvents: ",{"
    post_process:
      appendToEvents: "}"
      appendToFile: "]}"
```

Sink configuration for Source MI Events

```
spring:
  application:
    name: File Events (Sink)
  cloud:
    stream:
      bindings:
        input-0:
          destination: q.file.events.mesh.queue
          binder: solace
        output-0:
          destination: ${header(DEST_FILE_NAME)}
          binder: file-events
file-events:
  sink:
    general:
      miID: SINK_ID1
      state_backup_path: ./backup_state.cfg
    advanced:
      maxEventsPerFile: 0
      maxFileSize: 0
      ignoreEmptyEvents: true
      preventDuplicateEvents: true
      preventDuplicateOnlyForRedeliveredEvents: false
      uniqueEventIdentifyingHeader: "${header(DEST_FILE_NAME)}-${HEADER(EVENT_NO)}"
      closeFileTriggerHeaderName: "EVENT_TYPE"
      closeFileTriggerHeaderValue: "FILE_COMPLETE"
      ignorePayloadOfCloseTriggerEvent: true
```

Remote Protocol Configuration

By default, Micro-Integration will automatically read and write files on the local file system unless the supported remote protocols are enabled in the configuration.

The following remote protocols are supported

1. Google Cloud Storage
2. SSH File Transfer Protocol or Secure File Transfer Protocol (SFTP)
3. File Transfer Protocol (FTP/FTPs)

Google Cloud Storage

Google Cloud Storage is a service for storing objects in Google Cloud.

An object is an immutable piece of data consisting of a file of any format. You store objects in containers called buckets.

All buckets are associated with a project, and you can group your projects under an organization.

These configuration options are all prefixed by `file-events.(source|sink).`

Config Option	Type	Valid Values	Default Value	Description
`gcs.enabled`	`boolean`	`false, true`	`false`	Set this value to false to disable and true to enable Google Cloud Storage as source location. Default false, the file will be searched for on the local source machine and if set to true, the files/blobs will be pulled from a remote location on the Google Cloud Storage If enabled, advanced.maxEventsPerFile property will automatically set to 1.
`gcs.projectId`	`string`	`any`	`empty`	ProjectId value from Google Cloud Storage.

Config Option	Type	Valid Values	Default Value	Description
gcs.credentialsFilePath	string	any	empty	Credentials file path in JSON format to access Google Cloud Storage.
gcs.enableBulkComposeStrategy	boolean	true,false	true	When set to true, the sink connector will request to compose multipart blobs in bulk at the end of receiving the last Multipart. If set to false, sequential compose strategy will be applied to compose all multiparts to create the final file
gcs.retrySettings.maxAttempts	int	Integer Value	10	Maximum Retry Attempts in case of connection failure
gcs.retrySettings.retryDelayMultiplier	double	Double Value	3.0	Retry Delay Multiplier in case of connection failure
gcs.retrySettings.maxDurationMinutes	int	Integer Value	5	Maximum Duration of retries in minutes in case of connection failure

Secure File Transfer Protocol (SFTP)

Secure File Transfer Protocol (SFTP) is a network protocol that provides a secure way to access, transfer, and manage large files and sensitive data over a network.

SFTP leverages SSH for secure file transfers and mandates client authentication by the server to ensure only authorized access.

All commands and data are encrypted end-to-end, safeguarding passwords and other sensitive information from exposure in plain text during transit.

Authentication methods supported include `Password` or `Private Key` (optionally protected by a passphrase for added security).

These configuration options are all prefixed by `file-events.(source|sink).`

Config Option	Type	Valid Values	Default Value	Description
`sftp.enabled`	`boolean`	`false, true`	`false`	Enable or disable SFTP support. When set to false, file operations are performed on the local machine; when true, files are retrieved from or sent to a remote SFTP server, enabling secure remote file handling.
`sftp.ip`	`string`	`any`	`empty`	The IP address or hostname of the target SFTP server to connect to for remote file access.
`sftp.port`	`int`	`any`	`22`	The port number used by the SFTP server. The default (22) is the standard port for SSH/SFTP services; adjust if your server uses a non-standard port.
`sftp.user`	`string`	`any`	`empty`	The username associated with the SFTP account for authentication purposes.
`sftp.password`	`string`	`any`	`empty`	The password corresponding to the SFTP username. For enhanced security, consider using private key authentication instead of passwords.

Config Option	Type	Valid Values	Default Value	Description
sftp.strictHostKeyChecking	string	yes, no	yes	Determines whether to perform strict host key verification during connection establishment. Set to 'yes' (default) for secure validation against known hosts; 'no' disables it, which may be useful for testing but reduces security.
sftp.privateKeyPath	string	any	empty	The absolute or relative file path to the private key file used for key-based authentication, as an alternative to password-based login. Ensure the file permissions are restricted for security.
sftp.privateKeyPassphrase	string	any	empty	The passphrase used to decrypt an encrypted private key file. Omit if the private key is unencrypted.
sftp.connectTimeout	int	any	5000	The maximum duration in milliseconds to attempt establishing a connection to the SFTP server before declaring a timeout and retrying or failing.
sftp.keepAliveInterval	int	any	60000	The frequency in milliseconds for sending keep-alive packets to the SFTP server, helping to prevent connection drops due to perceived inactivity.

SFTP Proxy Configuration

Proxy settings allow routing SFTP connections through an intermediary HTTP or SOCKS proxy for scenarios like network restrictions or enhanced privacy. These options are prefixed by file-events.(source|sink).sftp.proxy.. The proxy is disabled by default and only activated when at least the proxy type, host, and port are configured.

Config Option	Type	Valid Values	Default Value	Description
sftp.proxy.proxyType	string	HTTP, SOCKS4, SOCKS5	empty	Specifies the proxy protocol: 'HTTP' for HTTP proxies, 'SOCKS4' for SOCKS version 4, or 'SOCKS5' for the more feature-rich SOCKS version 5 (supports authentication and UDP).
sftp.proxy.proxyHost	string	any	empty	The hostname or IP address of the proxy server that will forward SFTP traffic.
sftp.proxy.proxyPort	int	any	0	The listening port on the proxy server for incoming connections. Common defaults are 8080 for HTTP and 1080 for SOCKS.
sftp.proxy.proxyUsername	string	any	empty	The username for proxy server authentication, required for proxies that enforce user credentials (common in SOCKS5 and some HTTP setups).
sftp.proxy.proxyPassword	string	any	empty	The password paired with the proxy username for secure proxy authentication.

File Transfer Protocol (FTP/FTPs)

The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server. FTP users may authenticate themselves with a plain-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) Both FTP or FTP secure are supported '" These configuration options are all prefixed by file-events.(sink|source).

Config Option	Type	Valid Values	Default Value	Description
`ftp.enabled`	`boolean`	`false, true`	`false`	Set this value to false to disable and true to enable FTP/FTPs source location. Default false, the file will be searched for on the local source machine and if set to true, the files will be pulled from a remote location on the ftp server. Enabling this shifts the connector from local file system operations to remote FTP/FTPs interactions, allowing integration with external servers for file polling, listing, and transfer.
`ftp.traceEnabled`	`boolean`	`true,false`	`false`	Enable detailed FTP command/response tracing. When true, logs all FTP protocol exchanges (e.g., commands sent, server replies) to the TRACE level via SLF4J for debugging; disable in production to reduce log volume.
`ftp.ip`	`string`	`any`	`empty`	FTP Server IP. Specify the hostname or IP address of the remote FTP/FTPs server to connect to. This is required when enabled is true; empty value defaults to local file system if not set.

Config Option	Type	Valid Values	Default Value	Description
ftp.port	int	any	21	FTP server port number. The default port is 21 for plain FTP; use 990 for implicit FTPS. Ensure the port matches the server's configuration to establish successful connections.
ftp.user	string	any	empty	FTP user username. The credentials for authenticating with the FTP server. Anonymous access may use 'anonymous' if supported by the server, but a valid username is typically required.
ftp.password	string	any	empty	FTP user password. The password corresponding to the username for server authentication. For security, store this in a secure vault or environment variable rather than plaintext configuration.
ftp.useFTPS	boolean	true,false	false	For connecting over FTPs - TLS/secured. When true, enables FTPS for encrypted control and data channels, protecting against eavesdropping and man-in-the-middle attacks. Requires proper SSL/TLS configuration.
ftp.implicit	boolean	true,false	false	Use implicit FTPS mode (typically for port 990). In implicit mode, the entire session is encrypted from the start without needing an AUTH command; set to true only for servers requiring this legacy behavior.

Config Option	Type	Valid Values	Default Value	Description
ftp.strictHostKeyChecking	string	yes,no	yes	Strict host key checking for FTPS (yes/no). When 'yes', verifies the server's hostname against its certificate to prevent spoofing; set to 'no' for self-signed or untrusted certificates, but use cautiously.
ftp.keyStorePath	string	any	empty	Path to key store for client certificate authentication. Specifies the location of the JKS keystore file containing the client's private key and certificate for mutual TLS authentication in FTPS.
ftp.keyStorePassword	string	any	empty	Password for the key store. The passphrase to unlock the keystore file; required if keyStorePath is provided. Handle securely to avoid exposure.
ftp.trustStorePath	string	any	empty	Path to trust store for server certificate validation. Points to the JKS truststore containing trusted CA certificates to validate the FTPS server's identity.
ftp.trustStorePassword	string	any	empty	Password for the trust store. The passphrase for the truststore file; necessary when trustStorePath is set for custom certificate validation.
ftp.enabledProtocols	string	any	TLSv1.2,TLSv1.3	Comma-separated list of enabled TLS protocols. Restricts the TLS versions used in FTPS connections for security (e.g., 'TLSv1.3' only); defaults to modern secure protocols.

Config Option	Type	Valid Values	Default Value	Description
ftp.passiveMode	boolean	true,false	false	Sets the data connection mode to PASSIVE_LOCAL_DATA_CONNECTION_MODE. Enables passive mode where the server opens a port for data transfer, ideal for clients behind firewalls/NAT; active mode may be better for server-side restrictions.
ftp.activeMinPort	int	1-65535	0	Minimum local port for active mode data connections (0 for system-assigned). Defines the lower bound of ephemeral ports used in active mode to comply with firewall rules; set range for predictability.
ftp.activeMaxPort	int	1-65535	0	Maximum local port for active mode data connections (0 for system-assigned). The upper bound for active mode ports; must be >= activeMinPort if both are set.
ftp.listHiddenFiles	boolean	true,false	false	Include hidden files in directory listings. When true, uses server-specific commands (e.g., -a in UNIX) to list dot-files or hidden attributes, useful for complete file discovery.
ftp.useEPSVwithIPv4	boolean	true,false	false	Enable EPSV command in passive mode for IPv4 networks. Uses extended passive mode (EPSV) for better NAT/firewall compatibility by avoiding IP embedding in responses; fallback to PASV if unsupported.
ftp.remoteVerificationEnabled	boolean	true,false	false	Verify remote hostname/IP in passive mode responses. When true, cross-checks the IP/port returned by PASV/EPSV against the control connection host to detect proxy/misrouting issues.

Config Option	Type	Valid Values	Default Value	Description
ftp.serverSystemKey	string	UNIX,VMS,NETWARE,SYST_L8,etc.	UNIX	Server system type for parsing directory listings (e.g., UNIX, VMS, NETWARE). Configures the FTP client to interpret server-specific listing formats and timestamps accurately, preventing errors in file metadata extraction.
ftp.defaultFileTransferType	enum	ASCII,EBCDIC,BINARY,EBCDIC_IMAGE,LOCAL_EBCDIC	BINARY	Default file transfer mode (ASCII for text with line-ending translation; BINARY for raw bytes to avoid corruption). Set via enum value; applied globally unless overridden per operation. Use BINARY for mixed content safety.
ftp.serverTimezone	string	IANA Time Zone Database identifiers (e.g., "UTC", "America/New_York", "Europe/London")	System default (JVM□s local timezone)	The timezone of the FTP server used for parsing timestamps in directory listings and MDTM commands. If not specified, falls back to the JVM's default timezone for local alignment in file tracking.
ftp.connectTimeout	int	any (>0)	5000	Connection timeout in milliseconds. The maximum time to wait for the initial TCP connection to the FTP server; adjust higher for slow networks or lower for faster failure detection.
ftp.keepAliveInterval	int	any (>0)	60000	Control keep-alive interval in milliseconds. Sends periodic NOOP commands to maintain the control connection during idle periods, preventing timeouts on long-polling scenarios.

File Mesh Manager Configuration

All the file replications can be monitored on File Mesh Manager dashboard. The connector publishes the health, replication state to File Mesh Manager which analyses and represents the data in graphical representation.

File Mesh Manager is a separate package that need to be deployed as separate instance, which is built on ReactJS, NodeJS and uses postgres or oracle as databases.

File Mesh Manager can connect to the queue on configured Solace Instance to read and process file event state.

Once processed the information is represented on Dashboard.

Configuration Options

These configuration options are all prefixed by `file-events.<sink/source>.file_mesh_manager`:

Config Option	Type	Valid Values	Default Value	Description
`file_mesh_manager.enabled`	`boolean`	`true, false`	`false`	Set this value to false to disable and true to enable sending state events to file mesh manager

Config Option	Type	Valid Values	Default Value	Description
file_mesh_manager.solace_base_publish_topic	string	any	empty	Base publish topic for file mesh manager event. Connector will append adapter_id in first row and event state(start, complete, error, warning) to the topic when publishing data.Ex(<file_mesh_manager.solace_base_publish_topic>/<adapter_id>/<event_state>). This following topic subscription need to be added to File Mesh Manager queue 1. <file_mesh_manager.solace_base_publish_topic>/*/start 2. <file_mesh_manager.solace_base_publish_topic>/*/complete 3. <file_mesh_manager.solace_base_publish_topic>/*/error 4. <file_mesh_manager.solace_base_publish_topic>/*/warning
file_mesh_manager.solace_messaging_mode	string	DIRECT, PERSISTENT, NON-PERSISTENT	PERSISTENT	Set the message mode (DIRECT, PERSISTENT, NON-PERSISTENT)
file_mesh_manager.heartbeat_enabled	boolean	false, true	false	Set this to true to enable sending heartbeats to File Mesh Manager. (File Mesh Manager option need to be enabled as well). Setting it to false will disable sending heart beat events.

Config Option	Type	Valid Values	Default Value	Description
file_mesh_manager.heartbeat_interval	int	any	30	Integer value representing number of seconds to wait before sending heartbeat. This will work only if heartbeat is enabled.
file_mesh_manager.solace_base_publish_topic_heartbeat	string	any	empty	Base publish topic for heartbeat event. Connector will append adapter_id in first row and event state(heartbeat) to the topic when publishing data.Ex(<file_mesh_manager.solace_base_publish_topic_heartbeat>/<adapter_id>/<event_state>). This following topic subscription need to be added to heartbeat queue 1. <file_mesh_manager.solace_base_publish_topic_heartbeat>/*/heartbeat
file_mesh_manager.events.fileStart	boolean	true, false	false	true → send file start events for every file to File Mesh Manager false → No file start event
file_mesh_manager.events.fileComplete	boolean	true, false	false	true → send file complete events for every file to File Mesh Manager false → No file complete event
file_mesh_manager.events.connectorInit	boolean	true, false	true	true → send connector init event to File Mesh Manager false → No connector init event
file_mesh_manager.events.connectorStart	boolean	true, false	true	true → send connector start event with all configurations to File Mesh Manager false → No connector start event

Config Option	Type	Valid Values	Default Value	Description
`file_mesh_manager.events.connectorComplete`	`boolean`	`true`, `false`	`true`	true → send connector complete event after all files have been processed to File Mesh Manager false → No connector complete event

License

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

Support

Support is offered best effort via our Solace Developer Community.

Premium support options are available, please Contact Solace.