CEM MiNiFi C++ Agent 1.23.02

Collecting Audit Events

Date published: 2020-10-14
Date modified: 2023-03-09

https://docs.cloudera.com/

Legal Notice

Contents

Consume Windows event logs through MiNiFi

Learn how to capture Windows security events through MiNiFi and ship them to NiFi over Site to Site.

You can use EFM to create dataflows for MiNiFi and execute wherever it is located. All MiNiFi agents are assigned an agent class. When you turn on an agent, it contacts the EFM server for runtime instructions. The runtime instructions are set at the class level. That means, all agents within a class run the same instructions.

Setting up ConsumeWindowsEventLog

Learn how to set up the ConsumeWindowsEventLog processor to capture windows events through MiNiFi, and to send data from MiNiFi to NiFi.

Procedure

1. Download MiNiFi MSI and set the classname.

   In this example, the classname is test6. You can set this property at install time (MSI) or by going directly to the minifi.properties file. Also ensure that the nifi.c2.enable property is set to true. This informs MiNiFi that run time flow instructions will be received from EFM.
2. Start MiNiFi.

   MiNiFi can be configured to send data to multiple endpoints (for example, Kafka, NiFi, EventHub). In this example, data will be sent to NiFi over S2S.
3. Create an input port on NiFi.

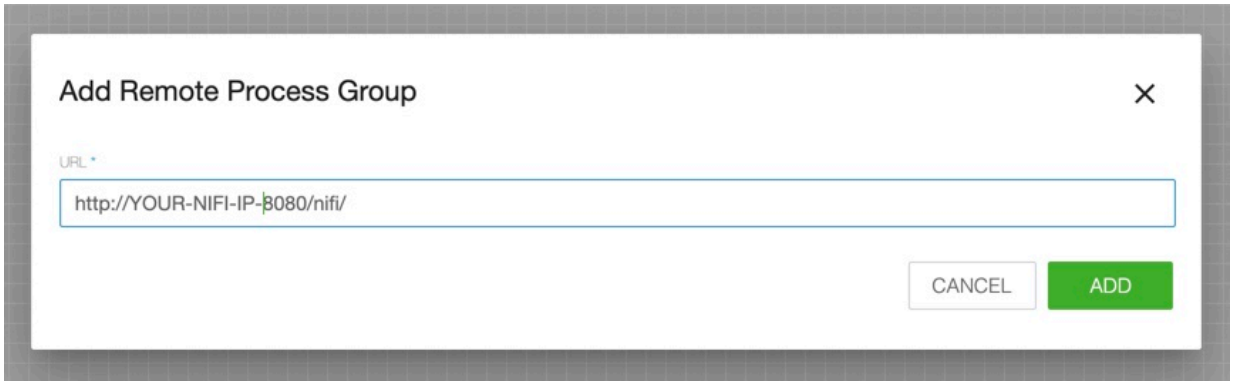
4. Capture the port ID.

   The port ID will be used in EFM later on.

5. On EFM, open the test6 class.

   This is where we design the flow for all agents with their class set to test6.
6. To capture windows events through MiNiFi, add ConsumeWindowsEventLog processor to the canvas.
7. Configure the processor to pull events.

   In this example, MiNiFi collects the windows security events.
8. To send data from MiNiFi to NiFi, add a Remote Process Group to the canvas and provide a NiFi endpoint.



9. Connect the ConsumeWindowsEventLog processor to the Remote Process Group and provide the NiFi Input Port ID captured earlier.
10. Click Publish.

    MiNiFi contacts EFM at a set interval (nifi.c2.agent.heartbeat.period). Once that occurs, MiNiFi receives new run time flow instructions. At that time data starts flowing into NiFi.

Apply Filters

Learn how you can query for events from a channel or a log file.

The channel or log file can exist on the local computer or in a remote computer. To specify the events that you want to get from the channel or log file, you can use an XPath query or a structured XML query. Windows Event Log supports a subset of XPath 1.0. The following example shows simple XPath expressions:

```
// The following query selects all events from the channel or log file
XPath Query: *

// The following query selects all the LowOnMemory events from the channel
 or log file
```

```
XPath Query: *[UserData/LowOnMemory]

// The following query selects all events with a severity level of 1 (Critic
al) from the channel or log file
XPath Query: *[System/Level=1]
// The following query shows a compound expression that selects all events
from the channel or log file
// where the printer's name is MyPrinter and severity level is 1.
XPath Query: *[UserData/*/PrinterName="MyPrinter" and System/Level=1]

// The following query selects all events from the channel or log file where
 the severity level is
// less than or equal to 3 and the event occurred in the last 24 hour perio
d.
XPath Query: *[System[(Level <= 3) and TimeCreated[timediff(@SystemTime) <=
 86400000]]]
```

You can use the XPath expressions directly when calling the EvtQuery or EvtSubscribe function or you can use a structured XML query that contains the XPath expression. You can use an XPath expression in simple queries that query events from a single source. If the XPath expression is a compound expression that contains more than 20 expressions or you are querying for events from multiple sources, then you must use a structured XML query.

For more information on how to query events and which API to use, see *https://docs.microsoft.com/en-gb/windows/win32/wes/querying-for-events*.

Using DefragmentText processor

Learn about the DefragmentText processor, its properties, its relationships, and its limitations. Also learn about how to use the DefragmentText processor.

DefragmentText processor buffers the incoming flow files until their contents create a cohesive message, based on the start or end line pattern.

Properties

The following list describes the properties of the DefragmentText processor:

- Pattern

  A regular expression to match at the start or end of messages.
- Pattern Location

  Whether the pattern is located at the start or at the end of the messages.
- Max Buffer Age

  The maximum age of the buffer after which it is transferred to success when matching Start of Message patterns or to failure when matching End of Message patterns.

  Expected format is <duration> <time unit>.
- Max Buffer Size

  The maximum buffer size. If the buffer exceeds this, it is transferred to failure.

  Expected format is <size> <size unit>.

The following image shows the DefragmentText processor properties:

DefragmentText (Processor)

Configuration

Settings

Processor Name*

DefragmentText

Penalty Duration*

30000 ms

Yield Duration*

1000 ms

Automatically Terminated Relationships

failure success

Scheduling

Scheduling Strategy*

Timer Driven

Concurrent Tasks*

1

Run Schedule*

1000 ms

Run Duration*

0ms 25ms 50ms 100ms 250ms 500ms 1s 2s

Properties

Property	Value
Max Buffer Age	5 min
Max Buffer Size	*No value set*
Pattern	**<[0-9]+>**
Pattern Location	Start of Message

Apply

Relationships

The two relationships of the DefragmentText processor are as follows:

- Success

  The part of the incoming flow files that form cohesive messages.
- Failure

  Flowfiles that failed the defragmentation process. This can happen if the buffer size is reached, or if the incoming files originate from different sources.

How to use

Simply connect to a source processor which generates a consecutive stream of text based data (for example, TailFile), and configure the Pattern and Pattern Location properties so that the DefragmentText processor can resegment the data with regex matching.

With the Maximum Buffer Size you can limit how large these flow files can grow (if the pattern matching fails), and with Maximum Buffer Age you can ensure that the messages are sent out even if there is no more incoming data.

The Failure relationship indicates that the data routed to this relationship might not be defragmented (Buffer size limit reached, incoming data is from different sources etc), but no data is lost.

Limitations

Limitations of the DefragmentText processor are as follows:

- It is a single threaded processor (multi threaded operations are disabled).
- Since this processor can only buffer one flow file at a time, this processor should only be used with a single source.
- When used with TailFile, TailFile should be in Single File mode.

Real world example: Tailing a Java application logfile

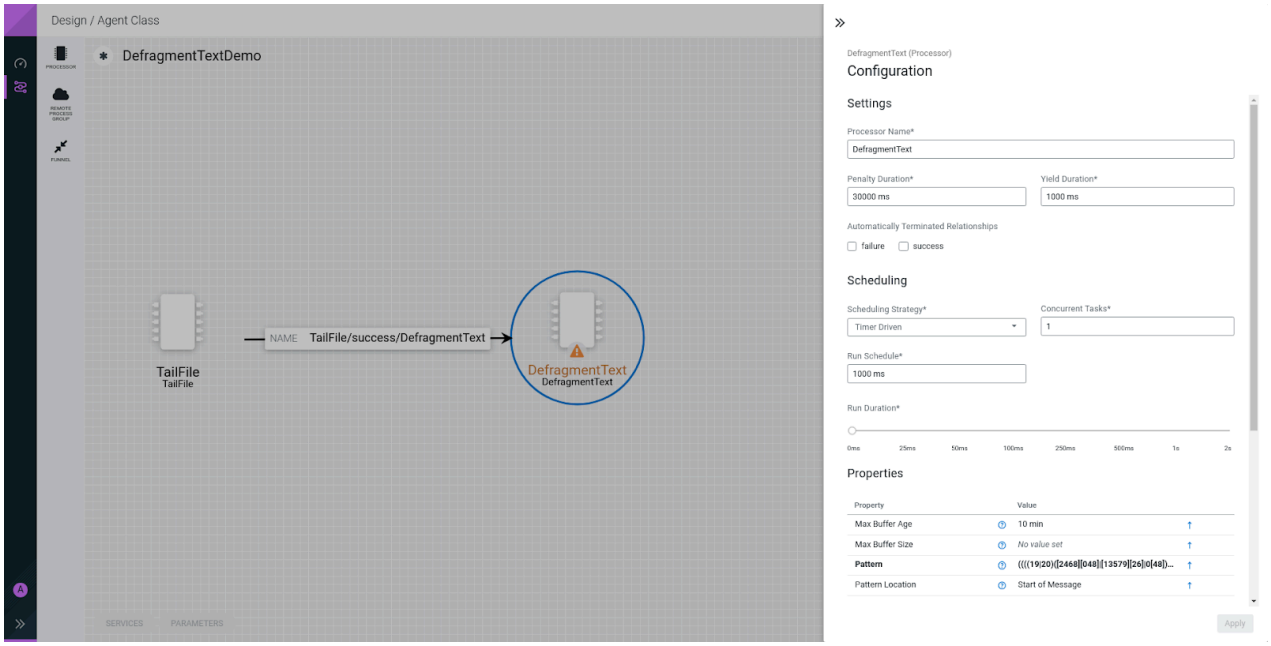
TailFile is set up to tail the log file of a Java application. The output of the TailFile is connected to the DefragmentText processor.

The DefragmentText Pattern property is set to a regular expression to match the timestamp at the start of each log message. For example,

```
(((19|20)([2468][048]|[13579][26]|0[48])|2000)-02-29|((19|20)[0-9]{2}-(0[467
8]|1[02])-(0[1-9]|[12][0-9]|30)|(19|20)[0-9]{2}-(0[1359]|11)-(0[1-9]|[12][0-
9]|3[01])|(19|20)[0-9]{2}-02-(0[1-9]|1[0-9]|2[0-8])))\s([01][0-9]|2[0-3]):([
012345][0-9]):([012345][0-9]))
```

The DefragmentText Pattern Location property is set to the Start of Message.

This setup ensures that when a java exception happens, the contents of that exception are not split among 10-20 flowfiles; instead they are in a single flowfile.


Design / Agent Class
DefragmentTextDemo
TailFile
TailFile
NAME
TailFile/success/DefragmentText
DefragmentText
DefragmentText
SERVICES
PARAMETERS
DefragmentText (Processor)
Configuration
Settings
Processor Name*
DefragmentText
Penalty Duration*
30000 ms
Yield Duration*
1000 ms
Automatically Terminated Relationships
failure
success
Scheduling
Scheduling Strategy*
Timer Driven
Concurrent Tasks*
1
Run Schedule*
1000 ms
Run Duration*
0ms
25ms
50ms
100ms
250ms
500ms
1s
2s
Properties
Property
Value
Max Buffer Age
10 min
Max Buffer Size
No value set
Pattern
((((19|20)([2468][048]|[13579][26]|0[48])...
Pattern Location
Start of Message
Apply