

# Apache NiFi Record Processing

**Bryan Bende / @bbende**

Staff Software Engineer  
September 8<sup>th</sup> 2017



# Background

- ◆ Flow File
  - Unit of work that moves through the data flow
  - Made up of attributes + content
- ◆ Attributes are a map of key/value pairs
  - Available in-memory as strings
  - Accessible from expression language
  - Useful for quick decision-making/routing
- ◆ Content is arbitrary bytes
  - Flow File is a pointer to the content in the content repository
  - Content is only accessed if the processor needs to operate on it
  - Could pass through many processors without every accessing the content

# The Problem

- ◆ Specialized processors to operate on different data types
  - SplitJson, EvaluateJsonPath, ConvertJsonToAvro
  - SplitAvro, ExtractAvroMetadata, ConvertAvroToJson
  - SplitText, ExtractText, RouteText
- ◆ Sometimes missing conversions
  - No ConvertCsvToJson, so ConvertCsvToAvro then ConvertAvroToJson
- ◆ Sometimes missing a specific function for a data type
  - No EvaluateAvroPath, so ConvertAvroToJson then EvaluateJsonPath
- ◆ Sometimes implemented with different libraries causing inconsistencies
  - Some Avro processors implemented with Kite, others with Apache Avro libraries
  - Each library may have different features/error-handling

# The Solution

- ◆ Introduce the concept of a "record"
- ◆ Centralize the logic for reading/writing records into controller services
- ◆ Provide standard processors that operate on records
- ◆ Can still handle arbitrary data, but process records when appropriate

# Record Readers & Writers

## ◆ Readers

- *AvroReader*
- *CsvReader*
- *GrokReader*
- *JsonPathReader*
- *JsonTreeReader*
- *ScriptedReader*

## ◆ Writers

- *AvroRecordSetWriter*
- *CsvRecordSetWriter*
- *JsonRecordSetWriter*
- *FreeFormTextRecordSetWriter*
- *ScriptedRecordSetWriter*

# But how is data turned into a record?

- ◆ A record has fields, and fields have information like a name and type
- ◆ Schemas define the fields of a record and give meaning to the data
- ◆ Apache Avro already utilizes schemas, widely used & supported by many tools
- ◆ We can use Avro schemas to define a schema for any type of data
- ◆ Each reader & writer needs a way to obtain a schema



# Schema Access Strategy

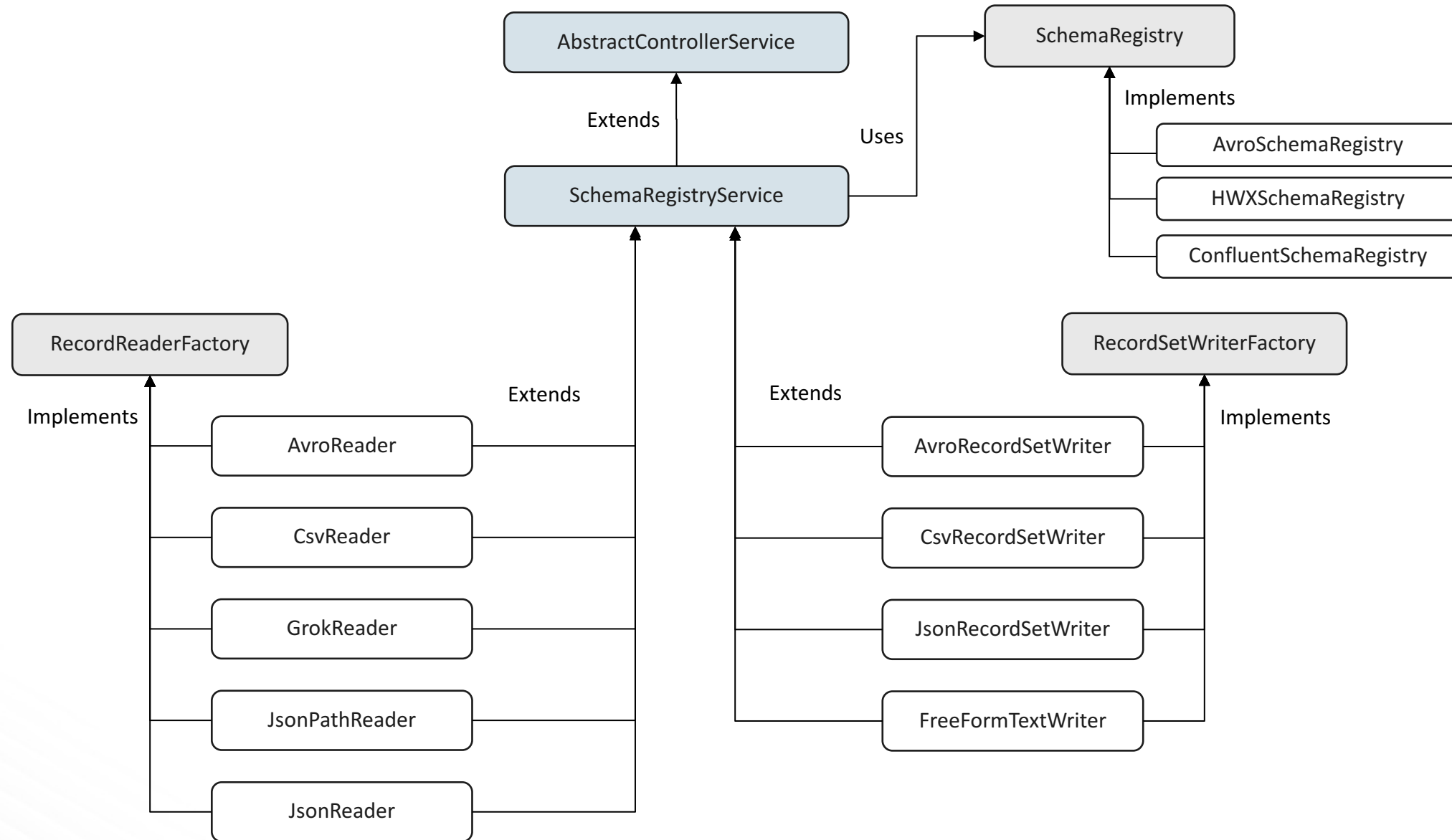
- ◆ *Schema Name*
  - Provide the name of a schema to look up in a Schema Registry, can use EL to obtain the name
- ◆ *Schema Text*
  - Provide the text of a schema in reader/writer, can use EL to obtain the text
- ◆ *HWX Content-Encoded Schema Reference*
  - Content of the Flow File contains special header referencing a schema in a Schema Registry
- ◆ *HWX Schema Reference Attributes*
  - Flow File contains three attributes that will be used to lookup a schema from the configured Schema Registry: *'schema.identifier'*, *'schema.version'*, and *'schema.protocol.version'*
- ◆ Readers & writers may have additional options specific to the data type
  - Ex: CsvReader can make a schema on the fly from the column names
  - Ex: AvroReader can use the schema embedded in the Avro data file

# Schema Registries

- ◆ Avro Schema Registry
  - Access schema by name
  - Only accessible with in NiFi
- ◆ Hortonworks Schema Registry
  - Access schema by name and/or version
  - Accessible across systems in the enterprise
  - <https://github.com/hortonworks/registry>
- ◆ Confluent Schema Registry
  - Access schema by name and/or version
  - Accessible across systems in the enterprise
  - <https://github.com/confluentinc/schema-registry>
  - *Not in an official Apache NiFi release yet, available in master branch (1.4.0-snapshot)*



# Full Picture



# Record Path

- ◆ Domain specific language (DSL) for specifying/accessing fields of a record
- ◆ Similar to JSON Path or XPath
- ◆ Examples:
  - Child: `/details/address/zip`
  - Descendant: `//zip`
  - Arrays: `/addresses[1]`
  - Maps: `/details/address['zip']`
  - Predicates: `/*[./state != 'NY']`
- ◆ More info...
  - <https://nifi.apache.org/docs/nifi-docs/html/record-path-guide.html>

# Record Processors

- ◆ Many processors for operating on records
  - *ConvertRecord*
  - *LookupRecord*
  - *PartitionRecord*
  - *QueryRecord*
  - *SplitRecord*
  - *UpdateRecord*
  - *ConsumeKafkaRecord\_0\_10*
  - *PublishKafkaRecord\_0\_10*
- ◆ Goal is to keep many records per flow file and avoid splitting if possible
- ◆ Check latest docs usage details and other record processors
  - <https://nifi.apache.org/docs.html>

# Example – CSV to JSON w/Local Schema Registry

# Example - CSV to JSON

- ◆ Incoming CSV that looks like:

```
first_name, last_name
```

```
John, Smith
```

```
Mike, Jones
```

- ◆ Want JSON that looks like:

```
[
```

```
{ "first_name" : "John", "last_name" : "Smith" },
```

```
{ "first_name" : "Mike", "last_name" : "Jones" }
```

```
]
```

# Step 1 – Define an Avro Schema

```
{  
  "name": "person",  
  "namespace": "nifi",  
  "type": "record",  
  "fields": [  
    { "name": "first_name", "type": "string" },  
    { "name": "last_name", "type": "string" }  
  ]  
}
```

## Step 2 - Create a Local Schema Registry & Add Schema

**NiFi Flow Configuration**

GENERAL CONTROLLER SERVICES

Name ▲

AvroSchemaRegistry

**Configure Controller Service**

SETTINGS PROPERTIES COMMENTS

Required field +

Property

person

```
1 {  
2   "name": "person",  
3   "namespace": "nifi",  
4   "type": "record",  
5   "fields": [  
6     { "name": "first_name", "type": "string" },  
7     { "name": "last_name", "type": "string" }  
8   ]  
9 }
```

☐ Set empty string

CANCEL OK

CANCEL APPLY



# Step 3 - Create a CsvReader

NiFi Flow Configuration

GENERAL

CONTROLLER SERVICES

Name ▲

AvroSchemaRegistry

CSVReader

Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

+

Property	Value	
Schema Access Strategy	Use 'Schema Name' Property	
Schema Registry	AvroSchemaRegistry	→
Schema Name	\${schema.name}	
Schema Text	\${avro.schema}	
Date Format	No value set	
Time Format	No value set	
Timestamp Format	No value set	
CSV Format	Custom Format	
Value Separator	,	
Skip Header Line	true	
Quote Character	"	
Escape Character	\	
Comment Marker	No value set	
Null String	No value set	

CANCEL

APPLY

# Step 4 – Create a JsonRecordSetWriter

NiFi Flow Configuration

GENERAL

CONTROLLER SERVICES

Name ▲

AvroSchemaRegistry

CSVReader

JsonRecordSetWriter

Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

Property	Value	
Schema Write Strategy	Set 'schema.name' Attribute	
Schema Access Strategy	Use 'Schema Name' Property	
Schema Registry	AvroSchemaRegistry	→
Schema Name	\${schema.name}	
Schema Text	\${avro.schema}	
Date Format	No value set	
Time Format	No value set	
Timestamp Format	No value set	
Pretty Print JSON	false	

CANCEL

APPLY

## Step 5 – GenerateFlowFile Processor

- Set Run Schedule to something like 10 seconds
- Put example CSV data in Custom Text property
- The reader & writer had their 'Schema Name' set to `${schema.name}`
- Add an property called 'schema.name' with the value of 'person' since this is the name in the schema registry

Configure Processor

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field

Property

File Size ?

Batch Size ?

Data Format ?

Unique FlowFiles ?

Custom Text ?

schema.name ? person

1 first\_name,last\_name  
2 John, Smith  
3 Mike, Jones

☐ Set empty string

CANCEL OK

CANCEL APPLY

## Step 6 – Convert Record Processor

- Select the appropriate reader and writer

### Configure Processor

SETTINGSSCHEDULINGPROPERTIESCOMMENTS

Required field +

Property		Value	
Record Reader	?	CSVReader	→
Record Writer	?	JsonRecordSetWriter	→

CANCELAPPLY

# Step 7 - LogAttribute

- Set Log Payload to true

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

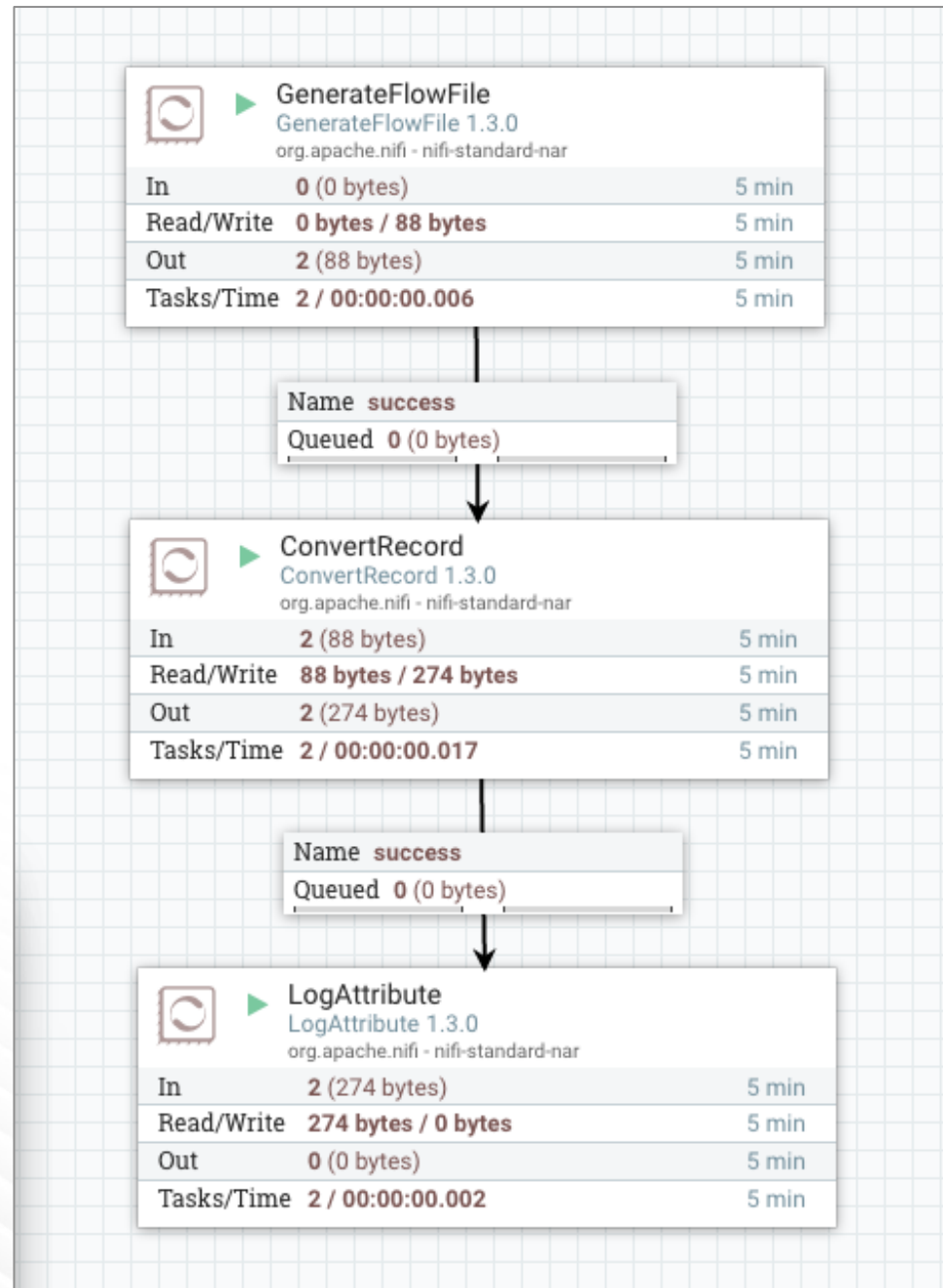
Required field

Property		Value	
Log Level	?	info	
Log Payload	?	true	
Attributes to Log	?	No value set	
Attributes to Ignore	?	No value set	
Log prefix	?	No value set	

CANCEL

APPLY

## Step 8 – Connect Processors & Run Flow



## Step 9 – Check nifi-app.log for JSON

-----Standard FlowFile Attributes

Key: 'entryDate ' Value: 'Thu Aug 31 13:28:02 EDT 2017'

Key: 'lineageStartDate' Value: 'Thu Aug 31 13:28:02 EDT 2017'

Key: 'fileSize' Value: '137'

FlowFile Attribute Map Content

Key: 'filename' Value: '326844487150210'

Key: 'mime.type' Value: 'application/json'

Key: 'path'Value: './'

Key: 'record.count' Value: '2'

Key: 'schema.name' Value: 'person'

Key: 'uuid'Value: 'e9198166-0cff-400b-a39d-9c8c9c565f85'

-----  
[  
{"first\_name":"John","last\_name":"Smith"},  
{"first\_name":"Mike","last\_name":"Jones"}  
]



# Example – CSV to JSON w/Hortonworks Schema Registry

# Step 1 – Run the Hortonworks Schema Registry

## ◆ Download the latest release

- <https://github.com/hortonworks/registry/releases/download/v0.2.1/hortonworks-registry-0.2.1.tar.gz>

## ◆ Extract the tar and run the application

- `tar xzvf hortonworks-registry-0.2.1.tar.gz`
- `cd hortonworks-registry-0.2.1`
- `./bin/registry-server-start.sh conf/registry-dev.yaml`

## ◆ Navigate to registry UI in your browser

- <http://localhost:9090>

## Step 2 – Add Schema

Add New Schema

NAME \*

person

DESCRIPTION \*

Person Schema

TYPE \*

Avro schema provider

SCHEMA GROUP \*

NiFi

COMPATIBILITY

BACKWARD

☒ EVOLVE

SCHEMA TEXT \*

1 {

2 "name": "person",

3 "namespace": "nifi",

4 "type": "record",

5 "fields": [

6 { "name": "first\_name", "type": "string"


7 { "name": "last\_name", "type": "string"

8 ]

9 }

CANCEL

SAVE

 **person**  
BACKWARD









TYPE  
avro

GROUP  
NiFi

VERSION  
1

SERIALIZER & DESERIALIZER  
0

## Step 3 – Create HortonworksSchemaRegistry Service


Name ▲	Type
  AvroSchemaRegistry	AvroSchemaRegistry 1.3.0
  CSVReader	
  <b>HortonworksSchemaRegistry</b>	
  JsonRecordSetWriter	




### Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field 

Property		Value
Schema Registry URL		http://localhost:9090/api/v1
Cache Size		1000
Cache Expiration		1 hour

CANCEL

APPLY

# Step 4 – Reconfigure CsvReader

Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

+

Property		Value	
Schema Access Strategy	?	Use 'Schema Name' Property	
Schema Registry	?	HortonworksSchemaRegistry	→
Schema Name	?	\${schema.name}	
Schema Text	?	\${avro.schema}	
Date Format	?	No value set	
Time Format	?	No value set	
Timestamp Format	?	No value set	
CSV Format	?	Custom Format	
Value Separator	?	,	
Skip Header Line	?	true	
Quote Character	?	"	
Escape Character	?	\	
Comment Marker	?	No value set	
Null String	?	No value set	

CANCEL

APPLY

# Step 5 – Reconfigure JsonRecordSetWriter

### Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

+

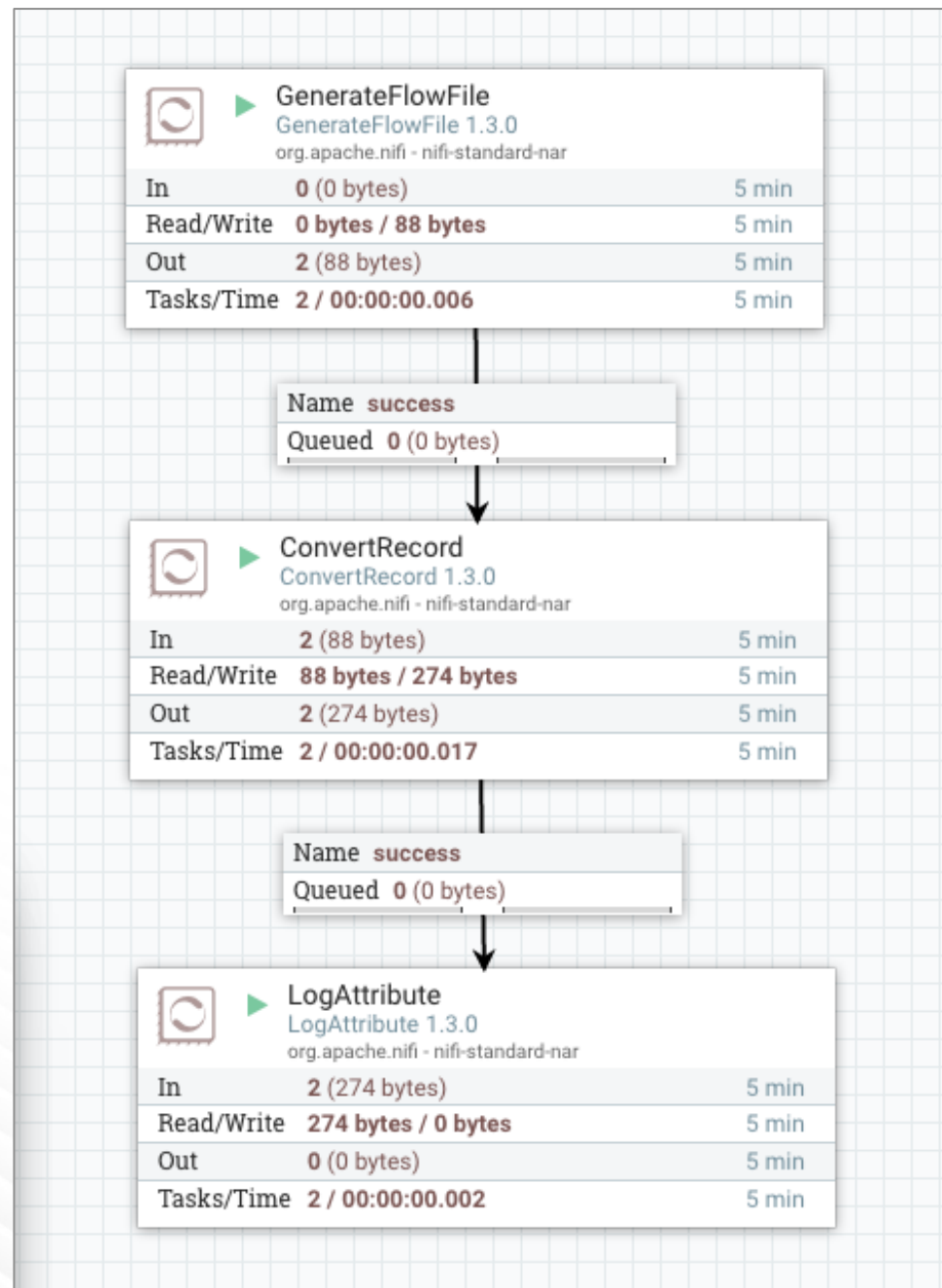
Property		Value	
Schema Write Strategy	?	Set 'schema.name' Attribute	
Schema Access Strategy	?	Use 'Schema Name' Property	
Schema Registry	?	HortonworksSchemaRegistry	→
Schema Name	?	\${schema.name}	
Schema Text	?	\${avro.schema}	
Date Format	?	No value set	
Time Format	?	No value set	
Timestamp Format	?	No value set	
Pretty Print JSON	?	false	

CANCEL

APPLY



## Step 6 – Run the same flow with same results



-----Standard FlowFile Attributes

Key: 'entryDate ' Value: 'Thu Aug 31 13:28:02 EDT 2017'

Key: 'lineageStartDate' Value: 'Thu Aug 31 13:28:02 EDT 2017'

Key: 'fileSize' Value: '137'

FlowFile Attribute Map Content

Key: 'filename' Value: '326844487150210'

Key: 'mime.type' Value: 'application/json'

Key: 'path' Value: './'

Key: 'record.count' Value: '2'

Key: 'schema.name' Value: 'person'

Key: 'uuid' Value: 'e9198166-0cff-400b-a39d-9c8c9c565f85'

-----  
[  
{"first\_name":"John","last\_name":"Smith"},  
{"first\_name":"Mike","last\_name":"Jones"}  
]

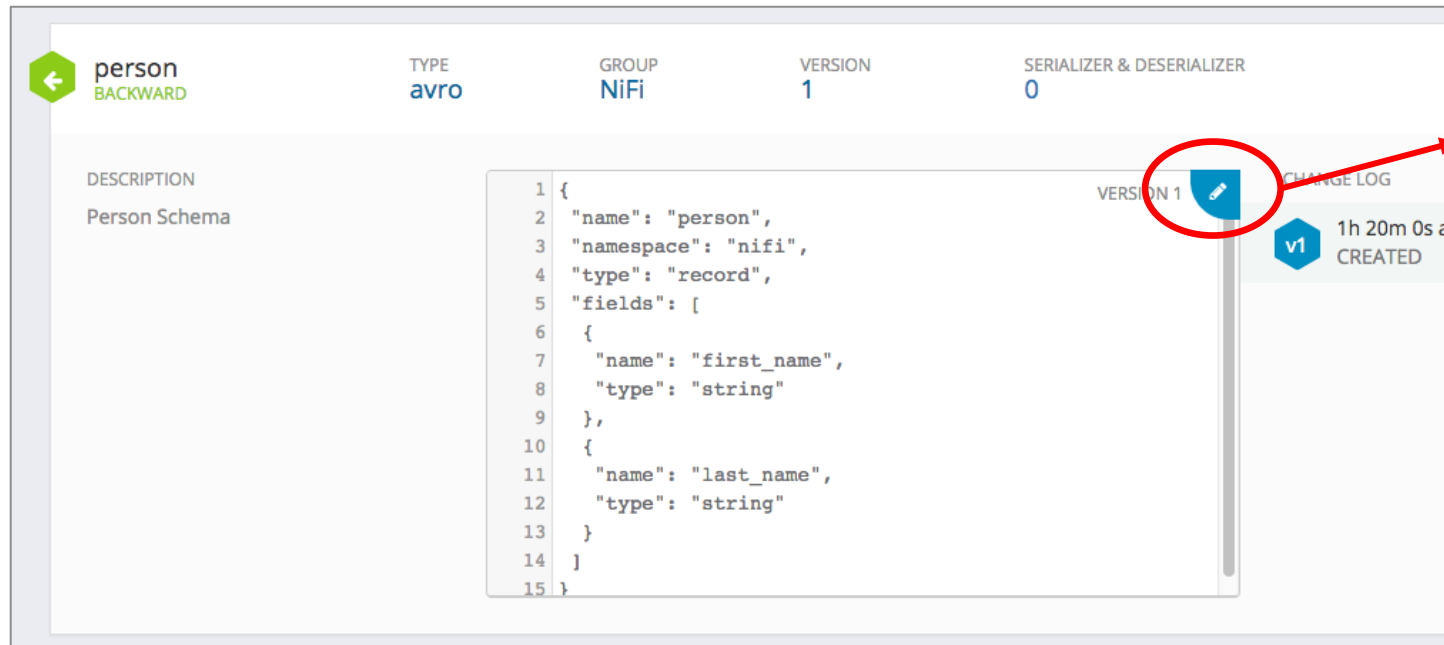


# Example – Use Specific Schema from HWX Schema Registry

# Specifying a Schema Version

- Previous example used “Schema Name” for “Schema Access Strategy”
  - NiFi retrieved latest version of schema for name
  - Cached schema based on configuration in controller service
- We can also use “HWX Schema Reference Attributes” to be more specific
  - *schema.identifier*
  - *schema.version*
  - *schema.protocol.version*

# Add New Version of Schema



person  
BACKWARD

TYPE  
avro

GROUP  
NiFi

VERSION  
1

SERIALIZER & DESERIALIZER  
0

DESCRIPTION  
Person Schema

```
1 {  
2   "name": "person",  
3   "namespace": "nifi",  
4   "type": "record",  
5   "fields": [  
6     {  
7       "name": "first_name",  
8       "type": "string"  
9     },  
10    {  
11      "name": "last_name",  
12      "type": "string"  
13    }  
14  ]  
15 }
```

VERSION 1

CHANGE LOG  
v1 1h 20m 0s ago  
CREATED



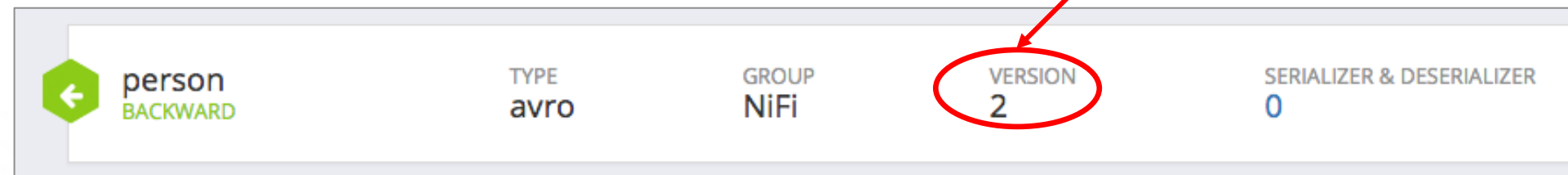
Edit Version

DESCRIPTION \*

Removing Last Name

SCHEMA TEXT \* CLEAR

```
1 {  
2   "name": "person",  
3   "namespace": "nifi",  
4   "type": "record",  
5   "fields": [  
6     { "name": "first_name", "type": "string" }  
7   ]  
8 }
```



person  
BACKWARD

TYPE  
avro

GROUP  
NiFi

VERSION  
2

SERIALIZER & DESERIALIZER  
0

# Obtaining Identifier, Version, Protocol

- We can get these values from the schema registry REST API
  - <http://localhost:9090/api/v1/schemaregistry/schemas/person>
  - <http://localhost:9090/api/v1/schemaregistry/schemas/person/versions>
  - Protocol Version is always '1' for now

# Update Flow to Specify Attributes

- Remove schema.name and add additional attributes in GenerateFlowFile

### Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field 

+

Property		Value	
File Size	?	0B	
Batch Size	?	1	
Data Format	?	Text	
Unique FlowFiles	?	false	
Custom Text	?	first_name,last_name John, Smith Mike, Jones	
schema.identifier	?	1	
schema.version	?	2	
schema.protocol.version	?	1	

CANCEL

APPLY

# Update CsvReader with new Schema Access Strategy

### Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

+

Property		Value	
Schema Access Strategy	?	HWX Schema Reference Attributes	
Schema Registry	?	HortonworksSchemaRegistry	→
Schema Name	?	\${schema.name}	
Schema Text	?	\${avro.schema}	
Date Format	?	No value set	
Time Format	?	No value set	
Timestamp Format	?	No value set	
CSV Format	?	Custom Format	
Value Separator	?	,	
Skip Header Line	?	true	
Quote Character	?	"	
Escape Character	?	\	
Comment Marker	?	No value set	
Null String	?	No value set	

CANCEL

APPLY



# Update JsonRecordSetWriter with new Schema Access Strategy

### Configure Controller Service

SETTINGS

PROPERTIES

COMMENTS

Required field

+

Property		Value	
Schema Write Strategy	?	Set 'schema.name' Attribute	
Schema Access Strategy	?	HWX Schema Reference Attributes	
Schema Registry	?	HortonworksSchemaRegistry	→
Schema Name	?	\${schema.name}	
Schema Text	?	\${avro.schema}	
Date Format	?	No value set	
Time Format	?	No value set	
Timestamp Format	?	No value set	
Pretty Print JSON	?	false	

CANCEL

APPLY



# Run the Flow Again

- Using v2 of the schema we should only see first\_name:

Key: 'schema.identifier' Value: '1'

Key: 'schema.name' Value: 'person'

Key: 'schema.protocol.version' Value: '1'

Key: 'schema.version' Value: '2'

Key: 'uuid' Value: '34407f4e-3bf1-46d5-a6d4-6da5ba197eb8'

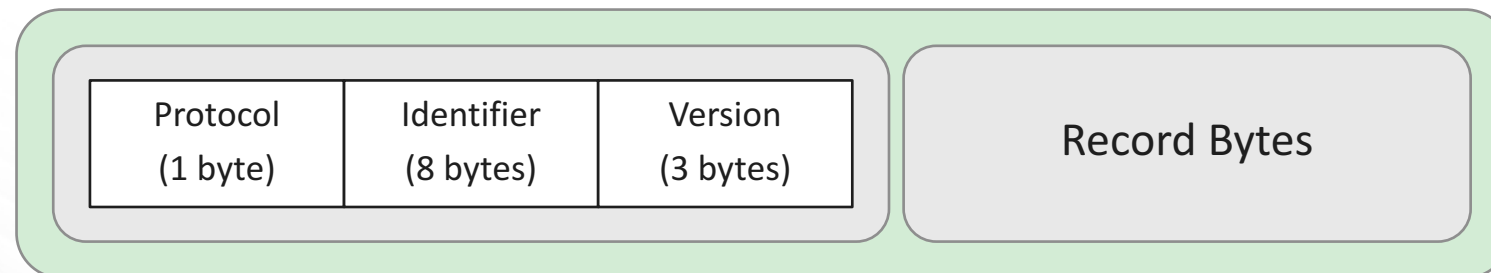
-----

[ { "**first\_name**": "John" }, { "**first\_name**": "Mike" } ]

# Apache NiFi + Apache Kafka + HWX Schema Registry

# Publishing

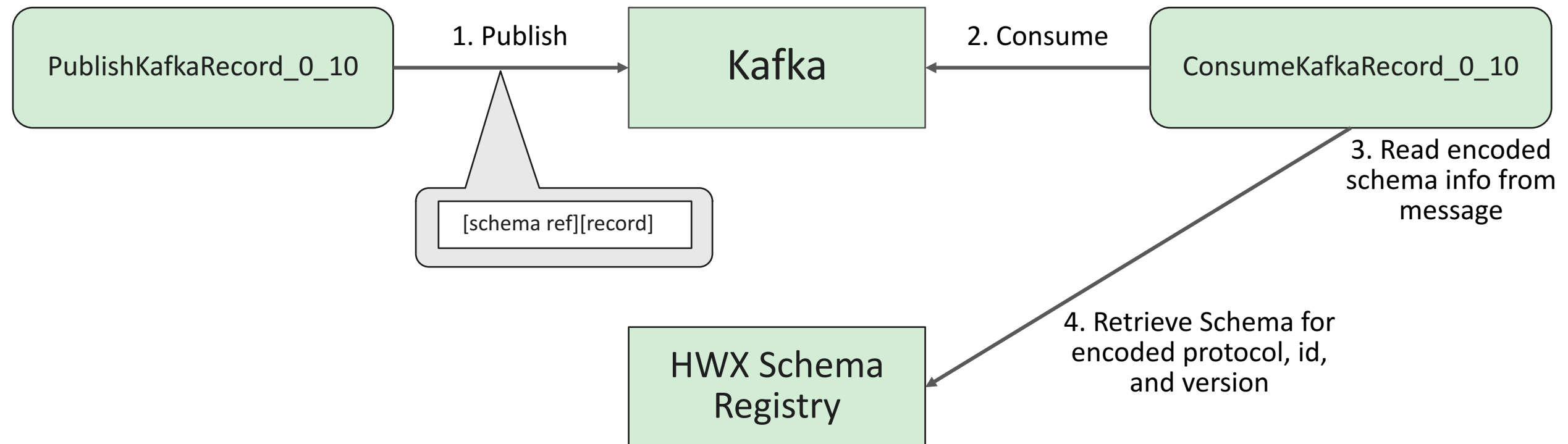
- ◆ PublishKafkaRecord\_0\_10
  - Streams incoming flow file as records using configured RecordReader
  - Serializes each record to bytes using configured RecordSetWriter
- ◆ Generally don't want to publish schema on every message
  - “*Schema Write Strategy*” of RecordSetWriter controls where schema ends up
  - “*HWX Content-Encoded Schema Reference*” encodes schema info at beginning of content
  - Single record published as encoded schema reference + bytes of a record



# Consuming

- ◆ ConsumeKafkaRecord\_0\_10
  - Reads messages from Kafka into records using configured RecordReader
  - Writes records to a flow file using configured RecordSetWriter
- ◆ If publisher used “*HWX Content-Encoded Schema Reference*” as the **Schema Writer Strategy** then consumer needs to use ““*HWX Content-Encoded Schema Reference*” as the **Schema Access Strategy**

# Publish & Consume



# Additional Resources

- ◆ <https://blogs.apache.org/nifi/entry/record-oriented-data-with-nifi>
- ◆ <https://blogs.apache.org/nifi/entry/real-time-sql-on-event>
- ◆ <https://community.hortonworks.com/content/kbentry/119766/installing-a-local-hortonworks-registry-to-use-wit.html>
- ◆ <https://community.hortonworks.com/articles/131320/using-partitionrecord-grokreaderjsonwriter-to-pars.html>
- ◆ <https://community.hortonworks.com/articles/115311/convert-csv-to-json-avro-xml-using-convertrecord-p.html>