1 Module Configuration

Streamline supports different attributes while specifying graph, node and link details in graphML format. A summary of these attributes is following:

Sl	Graph attributes	Description
1.	topicCfg	Topic configuration for the event streaming system
2.	faultCfg	Fault configuration (e.g., link down) for reliability tests

Sl	Node attributes	Description
1.	prodType	Data source type (used for data ingestion)
2.	prodCfg	Data source configuration
3.	consType	Data consumption type (used for data consumption)
4.	consCfg	Data sink configuration
5.	streamProcType	Stream processing engine type (e.g., Spark, Flink, KStream)
6.	streamProcCfg	Stream processing engine configuration
7.	storeType	Data store type (e.g., MySQL, MongoDB, RocksDB)
8.	storeCfg	Data store configuration
9.	brokerType	Message broker type (e.g., Kafka, RabbitMQ)
10	brokerCfg	Message broker configuration
11	cpuPercentage	Cap on overall system CPU usage

Sl	Link attributes	Description
1.	lat	Link latency (in milliseconds)
2.	bw	Link bandwidth (in Mbps)
3.	loss	Link loss (%)
4.	st	Source port
5.	dt	Destination port

Table 1: Streamline attributes

1.1. Brokers

To use event streaming platform as part of testing stream processing application, user requires to specify the event streaming platform (key='brokerType'). Type can be 'kafka'/ 'rMQ' based on the user application (currently supports Apache Kafka 2.8.0 and RabbitMQ 3.9.13). Default is kafka. For RabbitMQ, streamline supports three types of queue: classic, quorum and stream.

Users can specify a broker configuration YAML file path as a graph attribute in their input graph (key="brokerCfg"). The YAML configuration file may contain the following parameters:

brokerID : **no need to specify.** Streamline will implicitly take the host ID as the default brokerID (e.g., provided that brokerCfg specified in 'node id *h2'*, brokerID becomes 2.)

replicaMaxWait : Max wait time for each fetcher request issued by follower replicas.

replicaMinBytes : Minimum bytes expected for each fetch response.

** Specifying empty broker configuration YAML is considered as initiating a broker with default setup.

Table 2 shows default setup for broker configuration parameters:

Configuration Parameter	Value Format	Default Value	Restriction
replicaMaxWait	Integer	500	replicaMaxWait must be less than the value of REPLICA_LAG_TIME_MAX_MS (default is 30000ms)
replicaMinBytes	Integer	1	

Table 2: Default parameter setting for broker configuration parameters

1.2. Faults

Users can specify a fault configuration YAML file path as a graph attribute in their input graph (key="faultCfg"). The YAML configuration file may contain the following parameters:

duration : Duration of the link disconnection (in seconds).

links : Links to be disconnected for the specified duration. Users can choose single/ multiple links to be disconnected (e.g., s3-h2 / s1-h1, s1-h2).

1.3. Topics

Streamline requires an event streaming system (the default one is Apache Kafka) to forward messages among different components in the data processing pipeline. Because of that, the user needs to provide the desired configuration for public/subscribe topics (e.g., how many partitions a topic will have and its replication factor).

Users can specify a topic configuration YAML file path as a graph attribute in their input graph (key="topicCfg"). The YAML configuration file may contain the following parameters:

topicName : Name of the topic to be used for publishing/subscribing for messages.

topicBroker : Data broker in which the topic will be created.

topicPartition : How many partitions the topic will have.

topicReplica : How many replicas the topic will have.

1.4. Data producers

Users must indicate **a type and a configuration file for each data producer**. Both are **node attributes** in streamline's input graph (key="*prodType*" and key="*prodCfg*", respectively).

1.4.1. Producer type

Producers can be of five different types: SFST, MFST, ELTT, STANDARD, and CUSTOM.

SFST means "Single File to Single Topic". SFST producers will send the whole content of a file as a single message (stream data unit) to a given topic. They repeat this process until a certain number of messages (which is specified as a parameter) is reached.

MFST ("Multiple Files to Single Topic") producers are a variation of SFST where each message contains the content of a **different file**. All files produced by an MFST producer must be in the same directory.

ELTT stands for "Each Line To a Topic". In this case, each line of a file is produced to a given topic as a separate message. Lines from the **same file** will be repeatedly sent until a certain number of messages is reached.

STANDARD producer type is mainly for testing purposes. It will generate a string of 'a's and produce it to a topic at a certain frequency till the end of the simulation.

CUSTOM indicates the user will bring his/her own data producer. In this case, a producer path must be indicated. The user is responsible for ensuring the producer is producing data to the right topic as well as that it is producing enough data.

1.4.2. Producer configuration

In addition to the producer type, users must specify a YAML file path describing the producer configuration. Each producer **must** have its own configuration file, though the same template can be reused among producers. Producer configuration files may contain the following parameters:

producerPath : User provided producer script path.

filePath : Location of the file to be produced. N.B. for MFST producer type, user needs to assign the directory path in *filePath* attribute (ended with '/'). MFST producer will produce each file in that directory as a single message in the topic.

totalMessages : How many messages to be produced. For SFST and MFST, each file data is considered as single message, while for ELTT, each single line will be a new message. [Shouldn't it be "totalMessages"? what if we are producing lines rather than files?]

topicName : Name of the topic to be used for producing messages.

producerInstances : How many producer instances will be initiated on this node.

Table 3 shows a summary of which parameters are required for each producer type:

Configuration Parameter	Producer Type				
	SFST	**MFST**	**ELTT**	**STANDARD**	**CUSTOM**
producerPath	×	×	×	×	✓
topicName	✓	✓	✓	✓	×
filePath	✓	✓	✓	×	×
totalMessages	✓		✓	×	×
producerInstances	✓	✓	✓	✓	✓

Table 3. Required parameters for each producer type. ✓ : mandatory parameter. × : not needed.

[we should also move any producer-related parameter (e.g., message rate) we are currently reading as part of terminal input to the YAML files. Please check the code and see which other parameters should be added here]

Moreover, user can specify the following parameters for all producer types (excluding CUSTOM producers). If no value assigned for these parameters, streamline will work with default parameter setup (check Table 4 for default value setup).

messageRate : Message rate in msgs/second.

acks : Controls how many replicas must receive the record before producer considers successful write.

compression : Compression algorithm used to compress data sent to brokers.

batchSize : When multiple records are sent to the same partition, the producer will batch them together (bytes).

linger : Controls the amount of time (in ms) to wait for additional messages before sending the current batch.

requestTimeout : Controls how long producer waits for a reply from server when sending data.

bufferMemory : The total bytes of memory the producer can use to buffer records waiting to be sent to the server.

Table 4 shows default setup for some of the producer configuration parameters:

Configuration Parameter	Value Format	Default Value	Restriction
messageRate	String	None (As quick as possible)	User can specify any positive message rate less than 100 msg/second. If nothing specified, messages will be produced as quick as possible (Kafka default setup).
acks	Integer	1	Acks should be 0, 1 or 2. 0 : no replica acknowledgement 1 : only leader replica acknowledgement 2: all replica acknowledgement
compression	String	None	Currently supports gzip, snappy, lz4
batchSize	Integer	16384	
linger	Integer	0	
requestTimeout	Integer	30000	
bufferMemory	Integer	33554432	

Table 4: Default parameter setting for producer configuration parameters

1.5. Data consumers

Similarly, to producers, users must also indicate **a type and a configuration file for each data consumer**. Both are **node attributes** in streamline's input graph (key="consType" and key="consCfg", respectively).

1.5.1. Consumer type

Streamline currently supports two types of consumers: STANDARD and CUSTOM.

STANDARD consumers consume messages from a topic iteratively and log them. Each consumer instance will be in a different consumer group and consume messages from the beginning of the topic specified. [does the consumer connect to the broker only once or is the connection intermittent?] Consumer connects to the broker in an intermittent fashion.

CUSTOM indicates the user will bring his/her own data consumer. In this case, a consumer path must be indicated. The user is responsible for ensuring the consumer is consuming data from the right topic and logging any necessary information.

1.5.2. Consumer configuration

Consumer configuration file may contain the following parameters:

consumerPath : User provided consumer script path.

topicName : Name of the topic to be used for consuming messages.

consumerInstances : How many consumer instances will be initiated on this node.

Assigning configuration parameters vary by the consumer types. Here is a summary of the consumer configuration parameter user need to assign for different consumer types:

Configuration Parameter	Consumer Type	
	STANDARD	**CUSTOM**
consumerPath	×	✓
topicName	✓	×
consumerInstances	✓	✓

Table 5. Required parameters for each producer type. ✓ : mandatory parameter. × : not needed.

Moreover, user can specify the following parameters for STANDARD consumers. If no value assigned for these parameters, streamline will work with default parameter setup (check Table 6 for default value setup).

fetchMinBytes : Minimum amount of data consumer needs to receive from the broker when fetching records (bytes).

fetchMaxWait : How long the broker will wait (in ms) before sending data to consumer.

sessionTimeout : Time (in ms) a consumer can be out of contact with brokers while still considered alive.

Table 6 shows default setup for some of the consumer configuration parameters:

Configuration Parameter	Value Format	Default Value	Restriction
fetchMinBytes	Integer	1	
fetchMaxWait	Integer	500	
sessionTimeout	Integer	10000	Session timeout must be in the allowable range as configured in the broker configuration by group.min.session.timeout.ms value of 6000 and group.max.session.timeout.ms value of 1800000.

Table 6: Default parameter setting for consumer configuration parameters

1.6. Stream processing engines

To run stream processing applications, user requires to specify the stream processing engine type (key='streamProcType') and its configuration (key='streamProcCfg'). Type can be 'Spark'/ 'Flink'/ 'KStream' based on the user application (currently supports Spark and Flink).

1.6.1. Apache Spark Structured Streaming

Provided that streamProcType is chosen as Spark, user needs to specify a spark configuration YAML file path as a graph attribute in their input graph (key="streamProcCfg"). The YAML configuration file may contain the following parameters:

app : Path of the stream processing application to be tested.

To support Spark standalone cluster, parameter are:

cluster (True/False): This parameter indicates whether to run Spark in cluster mode. If set to True, Spark will run on a cluster of workers set up in the same machine. If nothing is set, Spark will run in local mode.

nSPEWorkerInstances : This parameter specifies the number of Spark worker instances to run on the machine. The value should be an integer.

nWorkerCores : This parameter specifies the number of cores to use on each worker machine. The value should be an integer. The number of cores can affect the performance of Spark tasks, with more cores generally leading to faster execution times. Ensure that the cumulative value of *nWorkerCores* does not surpass the total number of available cores on the machine. Over-allocating cores can lead to performance issues or errors.

workerMemory : This parameter specifies the amount of memory to allocate to each worker instance. The value should be a string with a unit of measurement, such as "512m" for 512 megabytes or "2g" for 2 gigabytes. The amount of memory can affect the performance of Spark tasks, with more memory generally leading to faster execution times. Ensure that the cumulative value of *workerMemory* does not surpass the total available memory on the machine.

1.6.2. Apache Flink

Provided that streamProcType is chosen as Flink, user needs to specify a Flink configuration YAML file path as a graph attribute in their input graph (key="streamProcCfg"). The YAML configuration file may contain the following parameters:

app : Path of the stream processing application to be tested.

P.S. Cluster support for Apache Flink is yet not tested.

1.7. Data store

To operate data stores as a consumer, user must specify both the type of the data store (using the storeType key) and its configuration file path (using the storeCfg key). The storeType can be set to 'MySQL', 'MongoDB', or 'RocksDB', depending on application requirements (please note, only MySQL has been thoroughly tested). It's user's responsibility to ensure the appropriate connector is installed and the database connection is properly configured for successful operation.

P.S. currently extending support for store configuration in YAML.