Creating Streams applications

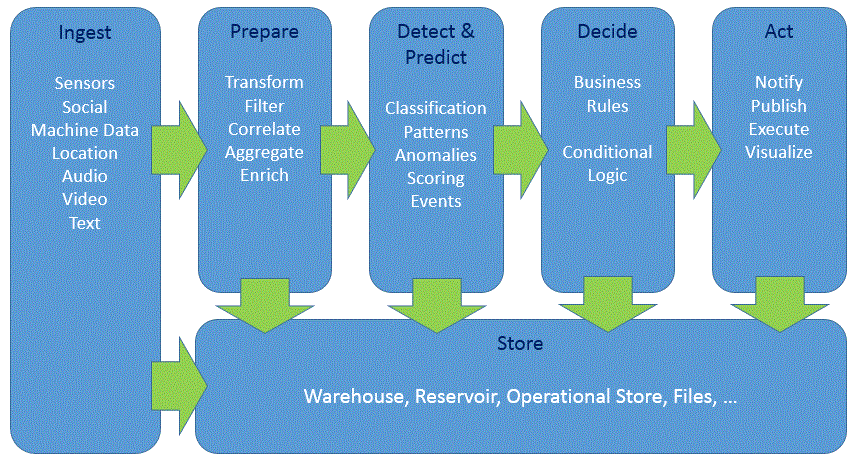
* Streams application pattern
* Best practice
* Acquiring Input Data
  + Ingest data using a source operator
    - Find the right source operator
    - Define the incoming data schema
    - (Optional) Convert JSON, XML and Binary data to tuples
    - Validate the incoming data
  + Generating data using a Beacon or Custom operator
    - Geospatial data generator
* Where to find examples
* Writing your own Custom operators

**Streams application pattern**

You have an idea of what your application should do, e.g. “Detect when a moving bus passes the known points of interest and send any alerts to the bus”.

How should you design your application?

First, recall the Streams application pattern:



All Streams applications follow this basic pattern: ingest the data, filter/discard any uninteresting/erroneous data, analyze it and act on the results.

Each stage in the diagram above is handled by one or more operators.

For example, in the BusAlerts application:



**INGEST data from NextBus and the POIs from a file**

**DETECT when a bus is near the POI**

**Print the ALERT**

The bus data is *ingested*, buses near POIs are *detected*, and then the *alert* is sent.

**Best practice**

To follow this pattern, you should build your application incrementally.

Instead of adding all the operators at once, **the best way to create your application is to do so in stages, starting small and progressively adding complexity.**

* Ingest the data and validate that the data you are working with is correct.
* Then, you can add operator(s) to further refine the data, with validation again as the last step.
* Add more complex analytics and logic
* Reporting, alerts and visualization.

So let’s start with the first step in your streaming application, which is acquiring data for processing.

**Acquiring input data**

Since all Streams applications start with a data ingestion step, this is the first stage of your application.

If you do not yet have data to ingest, you can skip to the “generating sample data” section for a few tips on generating data for your application.

Ingesting data using a source operator

The first step is to find the right source operator for your data.

The table below lists common data sources and the corresponding Streams operators.

View the full list of supported toolkits in the cloud and in a local install.

Supported toolkits on the IBM Cloud Streaming Analytics service [LINK]

<https://cloud.ibm.com/docs/apps/tutorials/tutorial_scratch.html#tutorial>

<https://cloud.ibm.com/docs/services/StreamingAnalytics/r_integrating_cloudant_rest.html#tutorials>

Supported toolkits on-prem [LINK] <https://www.ibm.com/support/knowledgecenter/SSCRJU_4.3.0/com.ibm.streams.toolkits.doc/spldoc/dita/toolkits/toolkits.html>

Where is my data coming from? File, HDFS, HBase, IoT devices, e.t.c.

|  |  |  |
| --- | --- | --- |
| Data source | Operator | Toolkit |
| Event Streams (formerly Message Hub) | MessageHubConsumer | streamsx.messagehub |
| MQTT | MQTTSource | streamsx.mqtt |
| Kafka | KafkaConsumer | streamsx.kafka |
| HDFS | HDFS2FileSource  HDFS2DirectoryScan | streamsx.hdfs |
| HBase | HBaseScan/HBaseGet | streamsx.hbase |
| Any JDBC compliant RDBMS | JDBCRun | streamsx.jdbc |
| JMS/XMS/ | JMSSource  XMSSource |  |

1B) Define the incoming data schema and use it with the source operator

Define the output schema that describes each incoming tuple:

e.g.

type XMLFromNextBus = xml xmlString;

type JsonFromKafka = rstring jsonString;

type DBRow = rstring id, rstring name, int32 id, timestamp last\_seen;

1c) If your data is in a different format, such as JSON or XML string, or a binary blob, it will need to be converted to Streams tuples.

For example, if you have JSON data, use the JSONToTuple operator to convert it to SPL tuples. The XMLParse operator is used to convert XML data to tuples.

Step 1d) Verify the data is correct. Create a small application that ingests the data and then prints it to console or to a file:

Example 1: no parsing needed

//change this to match the tuples you expect

type RawDataType = int32 id, rstring name, rstring timestamp;

composite MyApp {

graph

stream<RawDataType> DataFromXYZ = XYZSource() {

}

**() as DataPrinter = Custom(DataFromXYZ as port0) {**

**logic**

**onTuple port0: {**

**printStringLn(“New Tuple : + (rstring)port0.id + port0.name”);**

**}**

**}**

The DataPrinter operator will almost always be more or less the same as shown above.

If you are not using the Streaming Analytics service and you have access to the local filesystem, you could also write the incoming data to a file using a FileSink and verify the ouput file’s contents.

Example 2

Adding a parsing step and using a FileSink

Generating data

* Use a Beacon to generate data: <https://github.com/IBMStreams/samples/blob/master/Examples-for-beginners/003_sink_at_work/sample/sink_at_work.spl#L16>
* More complex samples can be generated using a Custom operator:

<https://github.com/IBMStreams/samples/blob/master/Geospatial/MapViewerSample/com.ibm.streamsx.mapviewer/Main.spl#L27>

Helper functions defined here: https://github.com/IBMStreams/samples/blob/master/Geospatial/MapViewerSample/com.ibm.streamsx.mapviewer.gen/GeospatialGen.spl

**Where to find examples**

* Samples for most toolkits are included in the toolkit repository in the samples folder.
* You can also search the Streams Samples catalog for examples. Click download zip to download the sample that you can import into Streams Studio, Atom or VSCode.
* Streamsdev also has articles and tutorials, search there.