Esper Adapter Reference Documentation

Version: 1.3.0

Table of Contents

Preface

Esper Adapters contains all input and output adapters for the Esper Java event stream processor.

If you are new to Esper, the Esper reference manual should be your first stop. If you are looking for

If you are looking for information on a specific adapter, you are at the right spot.

Chapter 1. Adapter Overview

Input and output adapters to Esper provide the means of accepting events from various sources, and for making available events to destinations.

The following input adapters are available. There are currently no output adapters available.

Table 1.1. Input and Output Adapters

Adapter	Description
CSV Input Adapter	The CSV input adapter can read one or more CSV files, transform the textual values into events, and play the events into the engine. The CSV adapter also makes it possible to run complete simulations of events arriving in time-order from different input streams.

1.1. Adapter Library Classes

1.1.1. Using AdapterInputSource

The `net.esper.adapter.AdapterInputSource` encapsulates information about an input source. Input adapters use the `AdapterInputSource` to determine how to read input. The class provides constructors for use with different input sources:

- `java.io.Reader` to read character streams
- `java.io.InputStream` to read byte streams
- URL
- Classpath resource
- `java.io.File`

Chapter 2. The CSV Input Adapter

This chapter discusses the CSV input adapter. CSV is an abbreviation for comma-separated values. CSV files are simple text files in which each line is a comma-separated list of values. CSV-formatted text can also be read from other input sources via `net.esper.adapter.AdapterInputSource`.

2.1. Introduction

In summary the CSV input adapter API can perform the following functions. Please consult the JavaDoc for additional information.

- Read events from a CSV file and send the events to an Esper engine instance
  - Read from different input sources
  - Use a timestamp column to schedule events being sent into the engine
  - Define a column ordering
  - Playback with options such as file looping, presence of a header row TODO
  - Use the Esper engine timer thread to read the CSV file
- Read multiple CSV files using a timestamp column to simulate events coming from different streams

2.2. Playback of CSV-formatted Events

The adapter reads events from a CSV input source and sends events to an engine using the class `net.esper.adapter.csv.CSVInputAdapter`.

```
AdapterInputSource source = new AdapterInputSource("simulation.csv");
(new CSVInputAdapter(epServiceProvider, source, "MyEventType")).start();
```

This example reads the file "simulation.csv" from classpath. The `AdapterInputSource` class can also take other input sources.

The event type `MyEventType` and it's property names and value types must be known to the engine:

- Configure the engine instance for a Map-based event type
- Place a header record in your CSV file that names each column as specified in the event type

The sample application code below shows how to configures via the configuration API a Map-based event type.

```
Map<String, Class> eventProperties = new HashMap<String, Class>();
eventProperties.put("symbol", String.class);
eventProperties.put("price", double.class);
eventProperties.put("volume", Integer.class);

Configuration configuration = new Configuration();
configuration.addEventTypeAlias("MyEventType", eventProperties);

epService = EPServiceProviderManager.getDefaultProvider(configuration);

EPStatement stmt = epService.getEPAdministrator().createEQL(
   "select symbol, price, volume from MyEventTypeA.win:length(100)");

(new CSVInputAdapter(epService, new AdapterInputSource(filename), "MyEventType")).start();
```

The contents of a sample CSV file is shown next.

```
symbol,price,volume
IBM,55.5,1000
```

The next code snippet outlines using a `java.io.Reader` an an alternative input source :

```
String myCSV = "symbol, price, volume" + NEW_LINE + "IBM, 10.2, 10000";
  StringReader reader = new StringReader(myCSV);
  (new CSVInputAdapter(epService, new AdapterInputSource(reader), "MyEventType")).start();
```

2.3. CSV Playback Options

Use the `CSVInputAdapterSpec` class to set playback options. The following options are available:

- Loop - Reads the CSV input source in a loop; When the end is reached, the input adapter rewinds to the beginning
- Events per second - Controls the number of events per second that the adapter sends to the engine
- Property order - Controls the order of event property values in the CSV input source, for use when the CSV input source does not have a header column
- Engine thread - Instructs the adapter to use the engine timer thread to read the CSV input source and send events to the engine
- Timestamp column name - Defines the name of the timestamp column in the CSV input source; The column must carry long-typed timestamp value relative to the current time; Use zero for the current time

The next code snippet shows the use of `CSVInputAdapterSpec` to set playback options.

```
CSVInputAdapterSpec spec = new CSVInputAdapterSpec(new AdapterInputSource(myURL), "MyEventType");
spec.setEventsPerSec(1000);
spec.setLooping(true);

InputAdapter inputAdapter = new CSVInputAdapter(epService, spec);
inputAdapter.start(); // method blocks unless engine thread option is set
```

2.4. Simulating Multiple Event Streams

The CSV input adapter can run simulations of events arriving in time-order from different input streams. Use the the `AdapterCoordinator` as a specialized input adapter for coordinating multiple CSV input sources by timestamp.

```
CSVInputAdapter firstFeed = new CSVInputAdapter(new AdapterInputSource(url1, "MyEventTwo"));
CSVInputAdapter secondFeed = new CSVInputAdapter(new AdapterInputSource(url2), "MyEventOne"));
AdapterCoordinator coordinator = new AdapterCoordinatorImpl(epService, true)

coordinator.coordinate(firstFeed);
coordinator.coordinate(secondFeed);
coordinator.start();
```