# CSV

## CSV

The CSV Data Format uses Apache Commons CSV to handle CSV payloads (Comma Separated Values) such as those exported/imported by Excel.

As of Camel 2.15.0, it now uses the Apache Commons CSV 1.1 which is based on a completely different set of options.

### Available options until Camel 2.15

| Option | Type | Description |
|---|---|---|
| config | CSVConfig | Can be used to set a custom `CSVConfig` object. |
| strategy | CSVStrategy | Can be used to set a custom `CSVStrategy`; the default is `CSVStrategy.DEFAULT_STRATEGY`. |
| autogenColumns | boolean | Whether or not columns are auto-generated in the resulting CSV. The default value is `true`; subsequent messages use the previously created columns with new fields being added at the end of the line. |
| delimiter | String | **Camel 2.4:** The column delimiter to use; the default value is `,`. |
| skipFirstLine | boolean | **Camel 2.10:** Whether or not to skip the first line of CSV input when unmarshalling (e.g. if the content has headers on the first line); the default value is `false`. |
| lazyLoad | boolean | **Camel 2.12.2:** Whether or not to Sequential access CSV input through an iterator which could avoid OOM exception when processing huge CSV file; the default value is false |
| useMaps | boolean | **Camel 2.13:** Whether to use List<Map> when unmarshalling instead of List<List>. |

### Available options as of Camel 2.15

| Option | Type | Description |
|---|---|---|
| `format` | `CSVFormat` | The reference format to use, it will be updated with the other format options, the default value is `CSVFormat.DEFAULT` |
| `commentMarkerDisabled` | `boolean` | Disables the comment marker of the reference format.<br><br>This option is `false` by default. |
| `commentMarker` | `Character` | Overrides the comment marker of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `null` for `CSVFormat.DEFAULT`. |

| | | |
|---|---|---|
| `delimiter` | `Character` | Overrides the delimiter of the reference format.<br><br>This option is `null` by defaut. When `null` it keeps the value of the reference format which is `','` for `CSVFormat.DEFAULT`. |
| `escapeDisabled` | `boolean` | Disables the escape character of the reference format.<br><br>This option is `false` by default. |
| `escape` | `Character` | Overrides the escape character of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `null` for `CSVFormat.DEFAULT`. |
| `headerDisabled` | `boolean` | Disables the header of the reference format.<br><br>This option is `false` by default. |
| `header` | `String[]` | Overrides the header of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `null` for `CSVFormat.DEFAULT`.<br><br>In the XML DSL, this option is configured using children `<header>` tags:<br><br>`<csv >`<br><br>`<header>orderId </header>`<br><br>`<header>amount< /header>`<br>`</csv>` |
| `allowMissingColumnNames` | `Boolean` | Overrides the missing column names behavior of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `false` for `CSVFormat.DEFAULT`. |
| `ignoreEmptyLines` | `Boolean` | Overrides the empty line behavior of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `true` for `CSVFormat.DEFAULT`. |
| `ignoreSurroundingSpaces` | `Boolean` | Overrides the surrounding spaces behavior of the reference format.<br><br>This option is `null` by default. When `null` it keeps the value of the reference format which is `false` for `CSVFormat.DEFAULT`. |
| `nullStringDisabled` | `boolean` | Disables the null string representation of the reference format.<br><br>This option is `false` by default. |

| | | |
|---|---|---|
| nullString | String | Overrides the null string representation of the reference format.<br><br>This option is null by default. When null it keeps the value of the reference format which is null for CSVFormat.DEFAULT. |
| quoteDisabled | boolean | Disables the quote of the reference format.<br><br>This option is false by default. |
| quote | Character | Overrides the quote symbol of the reference format.<br><br>This option is null by default. When null it keeps the value of the reference format which is '"' (double quote) for CSVFormat.DEFAULT. |
| quoteMode | QuoteMode | Overrides the quote mode of the reference format.<br><br>This option is null by default. When null it keeps the value of the reference format which is null for CSVFormat.DEFAULT. |
| recordSeparatorDisabled | boolean | Disables the record separator of the reference format.<br><br>This option is false by default. |
| recordSeparator | String | Overrides the record separator of the reference format.<br><br>This option is null by default. When null it keeps the value of the reference format which is \r\n (CRLF) for CSVFormat.DEFAULT. |
| skipHeaderRecord | Boolean | Overrides the header record behavior of the reference format.<br><br>This option is null by default. When null it keeps the value of the reference format which is false for CSVFormat.DEFAULT. |
| lazyLoad | boolean | Whether the unmarshalling should produce an iterator that reads the lines on the fly or if all the lines must be read at one.<br><br>This option is false by default. |
| useMaps | boolean | Whether the unmarshalling should produce maps for the lines values instead of lists. It requires to have header (either defined or collected).<br><br>This options is false by default. |
| recordConverter | CsvRecordConverter | Sets the record converter to use. If defines the useMaps options is disabled.<br><br>This option is null by default. |

## Marshalling a Map to CSV

The component allows you to marshal a Java Map (or any other message type that can be converted in a Map) into a CSV payload.

| | |
|---|---|
| Considering the following body | `Map<String, Object> body = new LinkedHashMap<>(); body.put("foo", "abc"); body.put("bar", 123);` |
| and this Java route definition | `from("direct:start") .marshal().csv() .to("mock:result");` |
| or this XML route definition | `<route> <from uri="direct:start" /> <marshal> <csv /> </marshal> <to uri="mock:result" /> </route>` |
| then it will produce | `abc,123` |

## Unmarshalling a CSV message into a Java List

Unmarshalling will transform a CSV messsage into a Java List with CSV file lines (containing another List with all the field values).

An example: we have a CSV file with names of persons, their IQ and their current activity.

```
Jack Dalton, 115, mad at Averell
Joe Dalton, 105, calming Joe
William Dalton, 105, keeping Joe from killing Averell
Averell Dalton, 80, playing with Rantanplan
Lucky Luke, 120, capturing the Daltons
```

We can now use the CSV component to unmarshal this file:

```
from("file:src/test/resources/?fileName=daltons.csv&noop=true")
    .unmarshal().csv()
    .to("mock:daltons");
```

The resulting message will contain a `List<List<String>>` like...

```
List<List<String>> data = (List<List<String>>) exchange.getIn().getBody();
for (List<String> line : data) {
    LOG.debug(String.format("%s has an IQ of %s and is currently %s", line.
get(0), line.get(1), line.get(2)));
}
```

## Marshalling a List<Map> to CSV

**Available as of Camel 2.1**

If you have multiple rows of data you want to be marshalled into CSV format you can now store the message payload as a `List<Map<String, Object>>` object where the list contains a Map for each row.

## File Poller of CSV, then unmarshaling

Given a bean which can handle the incoming data...

**MyCsvHandler.java**

```
// Some comments here
public void doHandleCsvData(List<List<String>> csvData)
{
    // do magic here
}
```

... your route then looks as follows

```
<route>
        <!-- poll every 10 seconds -->
        <from uri="file:///some/path/to/pickup/csvfiles?delete=true&amp;
consumer.delay=10000" />
        <unmarshal><csv /></unmarshal>
        <to uri="bean:myCsvHandler?method=doHandleCsvData" />
</route>
```

## Marshaling with a pipe as delimiter

| | |
|---|---|
| Considering the following body | `Map<String, Object> body = new LinkedHashMap<>(); body.put("foo", "abc"); body.put("bar", 123);` |

| | |
|---|---|
| and this Java route definition | ```<br>// Camel version < 2.15<br>CsvDataFormat oldCSV = new<br>CsvDataFormat();<br>oldCSV.setDelimiter("\|");<br>from("direct:start")<br>    .marshal(oldCSV)<br>    .to("mock:result")<br><br>// Camel version >= 2.15<br>from("direct:start")<br>    .marshal(new<br>CsvDataFormat().setDelimiter<br>('\|'))<br>    .to("mock:result")<br>``` |
| or this XML route definition | ```<br><route><br>  <from uri="direct:start" /><br>  <marshal><br>    <csv delimiter="\|" /><br>  </marshal><br>  <to uri="mock:result" /><br></route><br>``` |
| then it will produce | ```<br>abc\|123<br>``` |

## Using autogenColumns, configRef and strategyRef attributes inside XML DSL

**Available as of Camel 2.9.2 / 2.10 and deleted for Camel 2.15**

You can customize the CSV Data Format to make use of your own `CSVConfig` and/or `CSVStrategy`. Also note that the default value of the `autogenColumns` option is true. The following example should illustrate this customization.

```
<route>
  <from uri="direct:start" />
  <marshal>
    <!-- make use of a strategy other than the default one which is 'org.
apache.commons.csv.CSVStrategy.DEFAULT_STRATEGY' -->
    <csv autogenColumns="false" delimiter="|" configRef="csvConfig"
strategyRef="excelStrategy" />
  </marshal>
  <convertBodyTo type="java.lang.String" />
  <to uri="mock:result" />
</route>

<bean id="csvConfig" class="org.apache.commons.csv.writer.CSVConfig">
  <property name="fields">
    <list>
      <bean class="org.apache.commons.csv.writer.CSVField">
        <property name="name" value="orderId" />
      </bean>
      <bean class="org.apache.commons.csv.writer.CSVField">
        <property name="name" value="amount" />
      </bean>
    </list>
  </property>
</bean>

<bean id="excelStrategy" class="org.springframework.beans.factory.config.
FieldRetrievingFactoryBean">
  <property name="staticField" value="org.apache.commons.csv.CSVStrategy.
EXCEL_STRATEGY" />
</bean>
```

## Using skipFirstLine option while unmarshaling

**Available as of Camel 2.10 and deleted for Camel 2.15**

You can instruct the CSV Data Format to skip the first line which contains the CSV headers. Using the Spring/XML DSL:

```
<route>
  <from uri="direct:start" />
  <unmarshal>
    <csv skipFirstLine="true" />
  </unmarshal>
  <to uri="bean:myCsvHandler?method=doHandleCsv" />
</route>
```

Or the Java DSL:

```
CsvDataFormat csv = new CsvDataFormat();
csv.setSkipFirstLine(true);

from("direct:start")
  .unmarshal(csv)
.to("bean:myCsvHandler?method=doHandleCsv");
```

### Unmarshaling with a pipe as delimiter

Using the Spring/XML DSL:

```
<route>
  <from uri="direct:start" />
  <unmarshal>
    <csv delimiter="|" />
  </unmarshal>
  <to uri="bean:myCsvHandler?method=doHandleCsv" />
</route>
```

Or the Java DSL:

```
CsvDataFormat csv = new CsvDataFormat();
CSVStrategy strategy = CSVStrategy.DEFAULT_STRATEGY;
strategy.setDelimiter('|');
csv.setStrategy(strategy);

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

```
CsvDataFormat csv = new CsvDataFormat();
csv.setDelimiter("|");

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

```
CsvDataFormat csv = new CsvDataFormat();
CSVConfig csvConfig = new CSVConfig();
csvConfig.setDelimiter(";");
csv.setConfig(csvConfig);

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

**Issue in CSVConfig**

It looks like that

```
CSVConfig csvConfig = new CSVConfig();
csvConfig.setDelimiter(';');
```

doesn't work. You have to set the delimiter as a String!

## Dependencies

To use CSV in your Camel routes you need to add a dependency on **camel-csv**, which implements this data format.

If you use Maven you can just add the following to your pom.xml, substituting the version number for the latest and greatest release (see the download page for the latest versions).

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-csv</artifactId>
  <version>x.x.x</version>
</dependency>
```