

# Fact-Tools Documentation

## Automatisch generierte Fact-Tools Dokumentation

Christian Bockermann

Kai Bruegge

September 19, 2013

### Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet...

## Contents

<b>1</b>	<b>Einleitung zu den Fact-Tools</b>	<b>1</b>
	<b>Class Hierarchy</b>	<b>1</b>
<b>2</b>	<b>Package fact.utils</b>	<b>3</b>
2.1	Interface FactEvent . . . . .	4
2.1.1	Declaration . . . . .	4
2.1.2	Fields . . . . .	4
2.2	Class CutSlices . . . . .	4
2.2.1	Declaration . . . . .	5
2.2.2	Constructors . . . . .	5
2.2.3	Methods . . . . .	5
2.3	Class CutValues . . . . .	6
2.3.1	Declaration . . . . .	6
2.3.2	Constructors . . . . .	6
2.3.3	Methods . . . . .	6
2.4	Class Diff . . . . .	6
2.4.1	Declaration . . . . .	6
2.4.2	Constructors . . . . .	7
2.4.3	Methods . . . . .	7
2.5	Class ExFit . . . . .	7
2.5.1	Declaration . . . . .	7

2.5.2	Constructors	7
2.5.3	Methods	8
2.6	Class Remapping	8
2.6.1	Declaration	8
2.6.2	Constructors	8
2.6.3	Methods	8
2.7	Class SelectIndecesFromArray	8
2.7.1	Declaration	8
2.7.2	Constructors	8
2.7.3	Methods	8
2.8	Class SimpleFactEventProcessor	9
2.8.1	Declaration	9
2.8.2	All known subclasses	9
2.8.3	Constructors	9
2.8.4	Methods	9
2.9	Class SimpleFactPixelProcessor	10
2.9.1	Declaration	10
2.9.2	All known subclasses	10
2.9.3	Constructors	10
2.9.4	Methods	10
2.10	Class SumKey	10
2.10.1	Declaration	11
2.10.2	Constructors	11
2.10.3	Methods	11
2.11	Class ThresholdEventCounter	11
2.11.1	Declaration	11
2.11.2	Constructors	11
2.11.3	Methods	11
2.12	Class ThresholdPixelCounter	11
2.12.1	Declaration	11
2.12.2	Constructors	11
2.12.3	Methods	12
<b>3</b>	<b>Package fact</b>	<b>12</b>
3.1	Class Constants	12
3.1.1	Declaration	12
3.1.2	Fields	12
3.1.3	Constructors	15
3.2	Class CreateDocs	15
3.2.1	Declaration	15
3.2.2	Constructors	15
3.2.3	Methods	15
3.3	Class FactViewer	15
3.3.1	Declaration	15
3.3.2	Methods	15

<b>4</b>	<b>Package fact.statistics</b>	<b>17</b>
4.1	Class ArrayMean . . . . .	17
4.1.1	Declaration . . . . .	18
4.1.2	Constructors . . . . .	18
4.1.3	Methods . . . . .	18
4.2	Class ArrayRMS . . . . .	18
4.2.1	Declaration . . . . .	18
4.2.2	Constructors . . . . .	18
4.2.3	Methods . . . . .	18
4.3	Class ArrayVariance . . . . .	19
4.3.1	Declaration . . . . .	19
4.3.2	Constructors . . . . .	19
4.3.3	Methods . . . . .	19
4.4	Class CreateHistogram . . . . .	19
4.4.1	Declaration . . . . .	19
4.4.2	Constructors . . . . .	19
4.4.3	Methods . . . . .	20
4.5	Class PixelAverage . . . . .	20
4.5.1	Declaration . . . . .	20
4.5.2	Constructors . . . . .	21
4.5.3	Methods . . . . .	21
4.6	Class PixelDistribution2D . . . . .	21
4.6.1	Declaration . . . . .	21
4.6.2	Constructors . . . . .	21
4.6.3	Methods . . . . .	21
4.7	Class Quantiles . . . . .	23
4.7.1	Declaration . . . . .	23
4.7.2	Constructors . . . . .	23
4.7.3	Methods . . . . .	23
4.8	Class StdDeviation . . . . .	23
4.8.1	Declaration . . . . .	24
4.8.2	Constructors . . . . .	24
4.8.3	Methods . . . . .	24
<b>5</b>	<b>Package fact.filter</b>	<b>24</b>
5.1	Class AverageJumpRemoval . . . . .	25
5.1.1	Declaration . . . . .	25
5.1.2	Constructors . . . . .	25
5.1.3	Methods . . . . .	25
5.2	Class DrsCalibration . . . . .	26
5.2.1	Declaration . . . . .	26
5.2.2	Constructors . . . . .	26
5.2.3	Methods . . . . .	26
5.3	Class ExponentialSmoothing . . . . .	27
5.3.1	Declaration . . . . .	27
5.3.2	Constructors . . . . .	27

5.3.3	Methods . . . . .	27
5.4	Class <code>FirFilter</code> . . . . .	27
5.4.1	Declaration . . . . .	27
5.4.2	Constructors . . . . .	27
5.4.3	Methods . . . . .	27
5.5	Class <code>InterpolateBadPixel</code> . . . . .	28
5.5.1	Declaration . . . . .	28
5.5.2	Constructors . . . . .	28
5.5.3	Methods . . . . .	28
5.6	Class <code>MotionDiff</code> . . . . .	28
5.6.1	Declaration . . . . .	28
5.6.2	Constructors . . . . .	28
5.6.3	Methods . . . . .	29
5.7	Class <code>MovingAverage</code> . . . . .	29
5.7.1	Declaration . . . . .	29
5.7.2	Constructors . . . . .	29
5.7.3	Methods . . . . .	29
5.8	Class <code>MultiplyValues</code> . . . . .	29
5.8.1	Declaration . . . . .	29
5.8.2	Constructors . . . . .	29
5.8.3	Methods . . . . .	30
5.9	Class <code>RemoveSpikesMars</code> . . . . .	30
5.9.1	Declaration . . . . .	30
5.9.2	Constructors . . . . .	30
5.9.3	Methods . . . . .	30
5.10	Class <code>SliceNormalization</code> . . . . .	30
5.10.1	Declaration . . . . .	30
5.10.2	Constructors . . . . .	30
5.10.3	Methods . . . . .	31
<b>6</b>	<b>Package <code>fact.features</code></b> . . . . .	<b>31</b>
6.1	Class <code>DistributionFromShower</code> . . . . .	32
6.1.1	Declaration . . . . .	32
6.1.2	Fields . . . . .	32
6.1.3	Constructors . . . . .	32
6.1.4	Methods . . . . .	32
6.2	Class <code>HillasAlpha</code> . . . . .	33
6.2.1	Declaration . . . . .	33
6.2.2	Constructors . . . . .	33
6.2.3	Methods . . . . .	33
6.3	Class <code>HillasConcentration</code> . . . . .	34
6.3.1	Declaration . . . . .	34
6.3.2	Constructors . . . . .	34
6.3.3	Methods . . . . .	34
6.4	Class <code>HillasConcentration2</code> . . . . .	34
6.4.1	Declaration . . . . .	34

6.4.2	Constructors	34
6.4.3	Methods	35
6.5	Class <code>HillasDistance</code>	35
6.5.1	Declaration	35
6.5.2	Constructors	35
6.5.3	Methods	35
6.6	Class <code>HillasLength</code>	36
6.6.1	Declaration	36
6.6.2	Constructors	36
6.6.3	Methods	36
6.7	Class <code>HillasParameter</code>	36
6.7.1	Declaration	37
6.7.2	Constructors	37
6.7.3	Methods	37
6.8	Class <code>HillasWidth</code>	38
6.8.1	Declaration	38
6.8.2	Constructors	38
6.8.3	Methods	38
6.9	Class <code>Leakage</code>	39
6.9.1	Declaration	39
6.9.2	Constructors	39
6.9.3	Methods	39
6.10	Class <code>MaxAmplitude</code>	39
6.10.1	Declaration	40
6.10.2	Constructors	40
6.10.3	Methods	40
6.11	Class <code>MaxAmplitudePosition</code>	40
6.11.1	Declaration	40
6.11.2	Constructors	40
6.11.3	Methods	40
6.12	Class <code>NumberOfIslands</code>	41
6.12.1	Declaration	41
6.12.2	Constructors	41
6.12.3	Methods	41
6.13	Class <code>NumberOfPixelInShower</code>	41
6.13.1	Declaration	41
6.13.2	Constructors	41
6.13.3	Methods	41
6.14	Class <code>PhotonCharge</code>	42
6.14.1	Declaration	42
6.14.2	Constructors	42
6.14.3	Methods	42
6.15	Class <code>RisingEdge</code>	43
6.15.1	Declaration	43
6.15.2	Constructors	43
6.15.3	Methods	43

6.16	Class ShowerEllipse	43
6.16.1	Declaration	43
6.16.2	Fields	44
6.16.3	Constructors	44
6.16.4	Methods	44
6.17	Class Size	45
6.17.1	Declaration	45
6.17.2	Constructors	45
6.17.3	Methods	45
6.18	Class SizeInInterval	45
6.18.1	Declaration	45
6.18.2	Constructors	45
6.18.3	Methods	46
6.19	Class SourcePosition	46
6.19.1	Declaration	46
6.19.2	Constructors	47
6.19.3	Methods	47
6.20	Class TimeDependentParameter	48
6.20.1	Declaration	48
6.20.2	Constructors	49
6.20.3	Methods	49
<b>7</b>	<b>Package fact.io</b>	<b>49</b>
7.1	Class BinaryFactWriter	50
7.1.1	Declaration	50
7.1.2	Constructors	50
7.1.3	Methods	50
7.2	Class ByteChunkStream	51
7.2.1	Declaration	51
7.2.2	All known subclasses	51
7.2.3	Fields	51
7.2.4	Constructors	51
7.2.5	Methods	52
7.3	Class CreateAnimatedGif	52
7.3.1	Declaration	52
7.3.2	Constructors	52
7.3.3	Methods	52
7.4	Class FitsEventSplitter	53
7.4.1	Declaration	53
7.4.2	Constructors	53
7.4.3	Methods	53
7.5	Class FitsStream	53
7.5.1	Declaration	53
7.5.2	Constructors	53
7.5.3	Methods	54
7.6	Class FitsStream.FitsHeader	54

7.6.1	Declaration	54
7.6.2	Constructors	54
7.6.3	Methods	54
7.7	Class ReadMCcsv	55
7.7.1	Declaration	55
7.7.2	Fields	55
7.7.3	Constructors	55
7.7.4	Methods	55
7.8	Class RootASCIIWriter	56
7.8.1	Declaration	56
7.8.2	Constructors	56
7.8.3	Methods	56
7.9	Class SerializedEventStream	57
7.9.1	Declaration	57
7.9.2	Constructors	57
7.9.3	Methods	57
7.10	Class WeatherStream	58
7.10.1	Declaration	58
7.10.2	Fields	58
7.10.3	Constructors	58
7.10.4	Methods	58
7.11	Class Weird8ByteChunkStream	59
7.11.1	Declaration	59
7.11.2	Constructors	59
7.11.3	Methods	59
7.12	Class WStream	59
7.12.1	Declaration	59
7.12.2	Fields	59
7.12.3	Constructors	59

## 1 Einleitung zu den Fact-Tools

Ganz viel intro text zu den *Fact-Tools*. Lecker!

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet...

## Class Hierarchy

### Classes

- java.lang.Object

- AbstractLineStream
  - fact.io.ReadMCcsv (in 7.7, page 55)
- AbstractStream
  - fact.io.ByteChunkStream (in 7.2, page 51)
    - fact.io.WStream (in 7.12, page 59)
    - fact.io.WeatherStream (in 7.10, page 58)
  - fact.io.FitsStream (in 7.5, page 53)
  - fact.io.SerializedEventStream (in 7.9, page 57)
- CsvWriter
  - fact.io.BinaryFactWriter (in 7.1, page 50)
  - fact.io.RootASCIIWriter (in 7.8, page 56)
- fact.Constants (in 3.1, page 12)
- fact.CreateDocs (in 3.2, page 15)
- fact.features.DistributionFromShower (in 6.1, page 32)
- fact.features.HillasAlpha (in 6.2, page 33)
- fact.features.HillasConcentration (in 6.3, page 34)
- fact.features.HillasConcentration2 (in 6.4, page 34)
- fact.features.HillasDistance (in 6.5, page 35)
- fact.features.HillasLength (in 6.6, page 36)
- fact.features.HillasParameter (in 6.7, page 36)
- fact.features.HillasWidth (in 6.8, page 38)
- fact.features.Leakage (in 6.9, page 39)
- fact.features.NumberOfPixelInShower (in 6.13, page 41)
- fact.features.PhotonCharge (in 6.14, page 42)
- fact.features.ShowerEllipse (in 6.16, page 43)
- fact.features.Size (in 6.17, page 45)
- fact.features.SizeInInterval (in 6.18, page 45)
- fact.features.SourcePosition (in 6.19, page 46)
- fact.features.TimeDependentParameter (in 6.20, page 48)
- fact.filter.AverageJumpRemoval (in 5.1, page 25)
- fact.filter.DrsCalibration (in 5.2, page 26)
- fact.io.CreateAnimatedGif (in 7.3, page 52)
- fact.io.FitsEventSplitter (in 7.4, page 53)
- fact.io.FitsStream.FitsHeader (in 7.6, page 54)
- fact.statistics.ArrayMean (in 4.1, page 17)
- fact.statistics.ArrayRMS (in 4.2, page 18)
- fact.statistics.ArrayVariance (in 4.3, page 19)
- fact.statistics.CreateHistogram (in 4.4, page 19)
- fact.statistics.PixelDistribution2D (in 4.6, page 21)
- fact.statistics.Quantiles (in 4.7, page 23)
- fact.utils.CutSlices (in 2.2, page 4)
- fact.utils.CutValues (in 2.3, page 6)
- fact.utils.Diff (in 2.4, page 6)
- fact.utils.SelectIndicesFromArray (in 2.7, page 8)
- fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
  - fact.features.MaxAmplitudePosition (in 6.11, page 40)



- `fact.features.NumberOfIslands` (in 6.12, page 41)
- `fact.features.RisingEdge` (in 6.15, page 43)
- `fact.filter.ExponentialSmoothing` (in 5.3, page 27)
- `fact.filter.FirFilter` (in 5.4, page 27)
- `fact.filter.InterpolateBadPixel` (in 5.5, page 28)
- `fact.filter.MotionDiff` (in 5.6, page 28)
- `fact.filter.MovingAverage` (in 5.7, page 29)
- `fact.filter.MultiplyValues` (in 5.8, page 29)
- `fact.filter.RemoveSpikesMars` (in 5.9, page 30)
- `fact.filter.SliceNormalization` (in 5.10, page 30)
- `fact.statistics.StdDeviation` (in 4.8, page 23)
- `fact.utils.ExFit` (in 2.5, page 7)
- `fact.utils.Remapping` (in 2.6, page 8)
- `fact.utils.SimpleFactPixelProcessor` (in 2.9, page 10)
  - `fact.features.MaxAmplitude` (in 6.10, page 39)
  - `fact.statistics.PixelAverage` (in 4.5, page 20)
- `fact.utils.SumKey` (in 2.10, page 10)
- `fact.utils.ThresholdEventCounter` (in 2.11, page 11)
- `fact.utils.ThresholdPixelCounter` (in 2.12, page 11)
- `java.awt.Component`
  - `java.awt.Container`
    - `java.awt.Window`
      - `java.awt.Frame`
        - `javax.swing.JFrame`
      - `fact.FactViewer` (in 3.3, page 15)
- `java.io.InputStream`
  - `fact.io.Weird8ByteChunkStream` (in 7.11, page 59)

## Interfaces

- `fact.utils.FactEvent` (in 2.1, page 4)

## 2 Package `fact.utils`

### *Package Contents*

*Page*

### Interfaces

<b>FactEvent</b> .....	4
This is an implementation of the Data item interface that provides easy access to all pixels of an event by their SoftID.	

### Classes

<b>CutSlices</b> .....	4
This is a processor to cut slices of a Fact-Event in each Pixel.	
<b>CutValues</b> .....	6
This operator simply cuts all values below and above the min and maxValue.	

<b>Diff</b> .....	6
This operator calculates the difference of all the slices in each Pixel between two arrays given by the keys keyA and keyB and stores the result as a float array named outputKey.	
<b>ExFit</b> .....	7
This operator does a very simple fit of an exp-function to the data in each pixel.	
<b>Remapping</b> .....	8
This processors changes the order of the pixels in the data from SoftId to Chid	
<b>SelectIndecesFromArray</b> .....	8
This processors takes an array and an array of indices.	
<b>SimpleFactEventProcessor</b> .....	9
<b>SimpleFactPixelProcessor</b> .....	10
This class provides a simple Interface for someone who wants to build a processor that operates on a single pixel and returns a single value for each one.	
<b>SumKey</b> .....	10
This operator simply sums up all values with the given key.	
<b>ThresholdEventCounter</b> .....	11
<b>ThresholdPixelCounter</b> .....	11
This processor counts the number of Pixels in each event that have a value >maxValue.	

This package contains processors for common tasks such as Array modifications or Counters.

## 2.1 Interface FactEvent

This is an implementation of the Data item interface that provides easy access to all pixels of an event by their SoftID.

### 2.1.1 Declaration

```
public interface FactEvent
```

### 2.1.2 Fields

- java.lang.String **DATA\_KEY**
- java.lang.String **EVENT\_ID\_KEY**
- java.lang.String **TRIGGER\_NUM\_KEY**
- java.lang.String **TRIGGER\_TYPE\_KEY**
- int **NUM\_OF\_PIXELS**
- fact.viewer.ui.DefaultPixelMapping **PIXEL\_MAPPING**

## 2.2 Class CutSlices

This is a processor to cut slices of a Fact-Event in each Pixel. It takes the rawdata from a fact-event and cuts of all “textit{slice}”

### 2.2.1 Declaration

```
public class CutSlices
extends java.lang.Object
```

### 2.2.2 Constructors

- **CutSlices**  
public CutSlices()

### 2.2.3 Methods

- **getEnd**  
public java.lang.Integer **getEnd**()
  - **Returns** – the end
- **getKeys**  
public java.lang.String[] **getKeys**()
  - **Returns** – the keys
- **getStart**  
public java.lang.Integer **getStart**()
  - **Returns** – the start
- **process**  
public Data **process**(Data data)
  - **See also**
    - \* stream.DataProcessor#process(stream.Data)
- **setEnd**  
public void **setEnd**(java.lang.Integer end)
  - **Parameters**
    - \* end – the end to set
- **setKeys**  
public void **setKeys**(java.lang.String[] keys)
  - **Parameters**
    - \* keys – the keys to set

- **setStart**  
`public void setStart(java.lang.Integer start)`
  - **Parameters**
    - \* `start` – the start to set

## 2.3 Class CutValues

This operator simply cuts all values below and above the min and maxValue.

### 2.3.1 Declaration

```
public class CutValues
extends java.lang.Object
```

### 2.3.2 Constructors

- **CutValues**  
`public CutValues()`

### 2.3.3 Methods

- **getKeys**  
`public java.lang.String[] getKeys()`
- **getMaxValue**  
`public java.lang.Float getMaxValue()`
- **getMinValue**  
`public java.lang.Float getMinValue()`
- **process**  
`public Data process(Data event)`
  - **See also**
    - \* `stream.DataProcessor#process(stream.Data)`
- **setKeys**  
`public void setKeys(java.lang.String[] keys)`
- **setMaxValue**  
`public void setMaxValue(java.lang.Float maxValue)`
- **setMinValue**  
`public void setMinValue(java.lang.Float minValue)`

## 2.4 Class Diff

This operator calculates the difference of all the slices in each Pixel between two arrays given by the keys keyA and keyB and stores the result as a float array named outputKey.

### 2.4.1 Declaration

```
public class Diff
extends java.lang.Object
```

### 2.4.2 Constructors

- **Diff**  
public Diff()

### 2.4.3 Methods

- **getKeyA**  
public java.lang.String getKeyA()
- **getKeyB**  
public java.lang.String getKeyB()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **process**  
public Data process(Data input)
  - See also  
\* stream.DataProcessor#process(stream.Data)
- **setKeyA**  
public void setKeyA(java.lang.String keyA)
- **setKeyB**  
public void setKeyB(java.lang.String keyB)
- **setOutputKey**  
public void setOutputKey(java.lang.String output)

## 2.5 Class ExFit

This operator does a very simple fit of an exp-function to the data in each pixel. The function is simple section-wise defined curve based on the load and unload cycles of a traditional capacity. The peak position and amplitude will be set according to the values the MaxAmplitude Processor. This is not supposed to generate a good fit. Its intention is to identify showerpixel via the StdClean Processor.

### 2.5.1 Declaration

```
public class ExFit
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 2.5.2 Constructors

- **ExFit**  
public **ExFit()**

### 2.5.3 Methods

- **processSeries**  
public float[] **processSeries**(float[] value)

## 2.6 Class Remapping

This processors changes the order of the pixels in the data from SoftId to Chid

### 2.6.1 Declaration

```
public class Remapping
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 2.6.2 Constructors

- **Remapping**  
public **Remapping()**

### 2.6.3 Methods

- **processSeries**  
public short[] **processSeries**(short[] data)

## 2.7 Class SelectIndecesFromArray

This processors takes an array and an array of indices. It puts all values with the given indeces into a new array. That means the new Array is of the same length as the indices array.

### 2.7.1 Declaration

```
public class SelectIndecesFromArray
extends java.lang.Object
```

### 2.7.2 Constructors

- **SelectIndecesFromArray**  
public **SelectIndecesFromArray()**

### 2.7.3 Methods

- **getIndices**  
public java.lang.String **getIndices()**
- **getKey**  
public java.lang.String **getKey()**
- **getOutputKey**  
public java.lang.String **getOutputKey()**
- **process**  
public Data **process**(Data input)
- **setIndices**  
public void **setIndices**(java.lang.String indices)
- **setKey**  
public void **setKey**(java.lang.String key)
- **setOutputKey**  
public void **setOutputKey**(java.lang.String outputKey)

## 2.8 Class SimpleFactEventProcessor

### 2.8.1 Declaration

```
public abstract class SimpleFactEventProcessor
extends java.lang.Object
```

### 2.8.2 All known subclasses

ThresholdPixelCounter (in 2.12, page 11), ThresholdEventCounter (in 2.11, page 11), SumKey (in 2.10, page 10), SimpleFactPixelProcessor (in 2.9, page 10), Remapping (in 2.6, page 8), ExFit (in 2.5, page 7), StdDeviation (in 4.8, page 23), PixelAverage (in 4.5, page 20), SliceNormalization (in 5.10, page 30), RemoveSpikesMars (in 5.9, page 30), MultiplyValues (in 5.8, page 29), MovingAverage (in 5.7, page 29), MotionDiff (in 5.6, page 28), InterpolateBadPixel (in 5.5, page 28), FirFilter (in 5.4, page 27), ExponentialSmoothing (in 5.3, page 27), RisingEdge (in 6.15, page 43), NumberOfIslands (in 6.12, page 41), MaxAmplitudePosition (in 6.11, page 40), MaxAmplitude (in 6.10, page 39)

### 2.8.3 Constructors

- **SimpleFactEventProcessor**  
public **SimpleFactEventProcessor()**

### 2.8.4 Methods

- **finish**  
public void **finish()**

- **getColor**  
`public java.lang.String getColor()`
- **getKey**  
`public java.lang.String getKey()`
- **getOutputKey**  
`public java.lang.String getOutputKey()`
- **init**  
`public void init(ProcessContext context)`
- **process**  
`public Data process(Data input)`
- **processSeries**  
`public abstract java.io.Serializable processSeries(java.io.Serializable data)`
- **resetState**  
`public void resetState()`
- **setColor**  
`public void setColor(java.lang.String color)`
- **setKey**  
`public void setKey(java.lang.String key)`
- **setOutputKey**  
`public void setOutputKey(java.lang.String outputKey)`

## 2.9 Class SimpleFactPixelProcessor

This class provides a simple Interface for someone who wants to build a processor that operates on a single pixel and returns a single value for each one.

### 2.9.1 Declaration

```
public abstract class SimpleFactPixelProcessor
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 2.9.2 All known subclasses

PixelAverage (in 4.5, page 20), MaxAmplitude (in 6.10, page 39)

### 2.9.3 Constructors

- **SimpleFactPixelProcessor**  
`public SimpleFactPixelProcessor()`



### 2.9.4 Methods

- **processPixel**  
public abstract float **processPixel**(float[] **pixelData**)
- **processSeries**  
public float[] **processSeries**(float[] **data**)

## 2.10 Class SumKey

This operator simply sums up all values with the given key.

### 2.10.1 Declaration

```
public class SumKey
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 2.10.2 Constructors

- **SumKey**  
public **SumKey**()

### 2.10.3 Methods

- **processSeries**  
public java.lang.Double **processSeries**(float[] **data**)

## 2.11 Class ThresholdEventCounter

### 2.11.1 Declaration

```
public class ThresholdEventCounter
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 2.11.2 Constructors

- **ThresholdEventCounter**  
public **ThresholdEventCounter**()

### 2.11.3 Methods

- **processSeries**  
public java.lang.Long **processSeries**(float[] **data**)

## 2.12 Class ThresholdPixelCounter

This processor counts the number of Pixels in each event that have a value >maxValue.

### 2.12.1 Declaration

public class ThresholdPixelCounter  
**extends** fact.utils.SimpleFactEventProcessor (in 2.8, page 9)

### 2.12.2 Constructors

- **ThresholdPixelCounter**  
 public **ThresholdPixelCounter**()

### 2.12.3 Methods

- **getMaxValue**  
 public float **getMaxValue**()
- **processSeries**  
 public java.lang.Long **processSeries**(float[] **data**)
- **setMaxValue**  
 public void **setMaxValue**(float **maxValue**)

## 3 Package fact

<i>Package Contents</i>	<i>Page</i>
<b>Classes</b>	
<b>Constants</b> .....	<a href="#">12</a>
<b>CreateDocs</b> .....	<a href="#">15</a>
<b>FactViewer</b> .....	<a href="#">15</a>

The **Fact-Tools** are supposed to be a modular Analysis Framework for the **FACT Telescope**. It is build upon the **streams**-framework which allows to define the control- and dataflow of the program via .xml files. To quote the official [Website](#)

The streams framework is a Java implementation of a simple stream processing environment. It aims at providing a clean and easy-to-use Java-based platform to process streaming data. The core module of the streams library is a thin API layer of interfaces and classes that reflect a high-level view of streaming processes. This API serves as a basis for implementing custom processors and providing services with the streams library.

$$F(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

### 3.1 Class Constants

#### 3.1.1 Declaration

```
public class Constants
extends java.lang.Object
```

#### 3.1.2 Fields

- public static final java.lang.String **DEFAULT\_KEY\_MC**
- public static final java.lang.String **DEFAULT\_KEY\_CALIBRATED**
- public static final java.lang.String **DEFAULT\_KEY\_MC\_CALIBRATED**
- public static final java.lang.String **DEFAULT\_KEY**
- public static final java.lang.String **KEY\_EXFIT**
- public static final java.lang.String **KEY\_EVENT\_NUM**
- public static final java.lang.String **KEY\_TRIGGER\_TYPE**
- public static final java.lang.String **KEY\_MAX\_AMPLITUDE\_POSITIONS**
- public static final java.lang.String **KEY\_MAX\_AMPLITUDES**
- public static final java.lang.String **KEY\_PHOTONCHARGE**
- public static final java.lang.String **KEY\_SIMPLE\_CLEAN\_COREPIXEL**
- public static final java.lang.String **KEY\_CORENEIGHBOURCLEAN**
- public static final java.lang.String **KEY\_STD**
- public static final java.lang.String **KEY\_FIR\_RESULT**
- public static final java.lang.String **RISINGEDGEPOSITION**
- public static final java.lang.String **KEY\_AVERAGES**
- public static final java.lang.String **KEY\_SPIKES\_REMOVED**
- public static final java.lang.String **KEY\_STD\_SHOWER**
- public static final java.lang.String **KEY\_EXPONENTIALY\_SMOOTHED**
- public static final java.lang.String **KEY\_TIME\_MEDIAN\_CLEAN**
- public static final java.lang.String **PIXELSET**
- public static final java.lang.String **REMOVE\_SPIKES\_MARS**
- public static final java.lang.String **KEY\_PLOT\_COLORMAP**

- public static final java.lang.String **KEY\_EVENT\_TIME**
- public static final java.lang.String **KEY\_NORMALIZED\_SLICES**
- public static final java.lang.String **KEY\_SOURCE\_POSITION\_OVERLAY**
- public static final java.lang.String **KEY\_NUMBER\_ISLANDS**
- public static final java.lang.String **KEY\_SHOWER\_PHOTONCHARGE**
- public static final java.lang.String **KEY\_SHOWER\_ARRIVALTIME\_DEV**
- public static final java.lang.String **KEY\_COLOR**
- public static final java.lang.String **KEY\_INTERPOLATED\_DATA**
- public static final java.lang.String **ELLIPSE\_DELTA**
- public static final java.lang.String **ELLIPSE\_ALPHA**
- public static final java.lang.String **ELLIPSE\_ALPHA\_1**
- public static final java.lang.String **ELLIPSE\_ALPHA\_2**
- public static final java.lang.String **ELLIPSE\_ALPHA\_3**
- public static final java.lang.String **ELLIPSE\_LENGTH**
- public static final java.lang.String **ELLIPSE\_AREA**
- public static final java.lang.String **ELLIPSE\_WIDTH**
- public static final java.lang.String **ELLIPSE\_SIZE**
- public static final java.lang.String **ELLIPSE\_DISTANCE**
- public static final java.lang.String **HILLAS\_LEAKAGE\_BORDER**
- public static final java.lang.String **HILLAS\_LEAKAGE\_SECONDBORDER**
- public static final java.lang.String **HILLAS\_CONCENTRATION1**
- public static final java.lang.String **HILLAS\_CONCENTRATION2**
- public static final java.lang.String **ELLIPSE\_OVERLAY**
- public static final java.lang.String **HILLAS\_NUMBER\_ISLANDS**
- public static final float **PIXELGAIN**
- public static final int **NUMBEROFTHREADS**
- public static final int **NUMBEROFPIXEL**
- public static final double **COEFFICENTS\_CFD**

- public static final double **COEFFICIENTS\_N5**
- public static final double **COEFFICIENTS\_REMOVE\_SIGNAL**
- public static final double **COEFFICIENTS\_N3**
- public static final double **PIXEL\_SIZE**
- public static final java.lang.String **SOURCE\_POS\_X**
- public static final java.lang.String **SOURCE\_POS\_Y**
- public static final java.lang.String **ERROR\_WRONG\_KEY**
- public static final java.lang.String **EXPECT\_ARRAY\_F**
- public static final java.lang.String **PLOT\_AREAVSSIZE**
- public static final java.lang.String **PLOT\_ANGLE\_HISTOGRAM**
- public static final java.lang.String **PLOT\_PER\_PIXEL**
- public static final java.lang.String **PLOT\_FILE\_SEPARATOR**
- public static final java.lang.String **EXPECT\_ARRAY**
- public static final java.lang.String **KEY\_DIFF**

### 3.1.3 Constructors

- **Constants**  
public Constants()

## 3.2 Class CreateDocs

### 3.2.1 Declaration

```
public class CreateDocs
extends java.lang.Object
```

### 3.2.2 Constructors

- **CreateDocs**  
public CreateDocs()

### 3.2.3 Methods

- **main**  
public static void main(java.lang.String[] args)  
– **Parameters**  
\* args –

### 3.3 Class FactViewer

#### 3.3.1 Declaration

```
public class FactViewer  
extends javax.swing.JFrame
```

#### 3.3.2 Methods

- **getCamMap**  
public viewer.ui.CameraPixelMap getCamMap()
- **getCamWindowList**  
public java.util.ArrayList getCamWindowList()  
  
– Returns – the overlays
- **getChartWindowList**  
public java.util.ArrayList getChartWindowList()
- **getCurrentKey**  
public java.lang.String getCurrentKey()
- **getEvent**  
public Data getEvent()
- **getEvWList**  
public java.util.ArrayList getEvWList()
- **getInstance**  
public static FactViewer getInstance()
- **getNextButton**  
public javax.swing.JButton getNextButton()  
  
– Returns – the map
- **getOverlayPanel**  
public viewer.ui.OverlayPanel getOverlayPanel()
- **getPrevButton**  
public javax.swing.JButton getPrevButton()
- **loadFitsFile**  
public void loadFitsFile(java.io.File file) throws java.lang.Exception
- **loadNextEvent**  
public void loadNextEvent()
- **main**  
public static void main(java.lang.String[] args) throws  
java.lang.Exception

– Parameters

\* args –

- **selectSlice**  
public void selectSlice(int i)
- **setCamMap**  
public void setCamMap(viewer.ui.CameraPixelMap camMap)
- **setCamWindowList**  
public void setCamWindowList(java.util.ArrayList camWindowList)
- **setChartWindowList**  
public void setChartWindowList(java.util.ArrayList chartWindowList)
- **setCurrentKey**  
public void setCurrentKey(java.lang.String currentKey)
- **setEvent**  
public void setEvent(Data event)

– Description

This will be called whenever a new Event is supposed to be displayed.

– Parameters

\* event – The Event to be displayed

- **setEvWList**  
public void setEvWList(java.util.ArrayList evWList)
- **setOverPanel**  
public void setOverPanel(viewer.ui.OverlayPanel over)

## 4 Package fact.statistics

*Package Contents*

*Page*

### Classes

<b>ArrayMean</b> .....	17
This operator calculates the mean value of hte values in of the array specified by the key	
<b>ArrayRMS</b> .....	18
This operator calculates the rms of the array specified by the key	
<b>ArrayVariance</b> .....	19
This operator calculates the rms of the array specified by the key	
<b>CreateHistogram</b> .....	19
Takes a float[] and returns an int[]	
<b>PixelAverage</b> .....	20
This operator calculates the average of all the slices in each Pixel and stores the result as a double array.	

<b>PixelDistribution2D</b> .....	21
<b>Quantiles</b> .....	23
<b>StdDeviation</b> .....	23
This Processor calculates the Standarddeviation of the slices in each pixel.	

This package is supposed to be a collection of convinience processors to help calculate some statistical values. For example the ArrayRMS, ArrayMean, ArrayVariance etc. processors take an array as input and put the values back into the map.

## 4.1 Class ArrayMean

This operator calculates the mean value of hte values in of the array specified by the key

### 4.1.1 Declaration

```
public class ArrayMean
extends java.lang.Object
```

### 4.1.2 Constructors

- **ArrayMean**  
public ArrayMean()

### 4.1.3 Methods

- **getKey**  
public java.lang.String getKey()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **process**  
public Data process(Data input)
- **setKey**  
public void setKey(java.lang.String key)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)

## 4.2 Class ArrayRMS

This operator calculates the rms of the array specified by the key



#### 4.2.1 Declaration

```
public class ArrayRMS  
extends java.lang.Object
```

#### 4.2.2 Constructors

- **ArrayRMS**  
    **public** **ArrayRMS**()

#### 4.2.3 Methods

- **getKey**  
    **public** java.lang.String **getKey**()
- **getOutputKey**  
    **public** java.lang.String **getOutputKey**()
- **process**  
    **public** Data **process**(Data input)
- **setKey**  
    **public** void **setKey**(java.lang.String key)
- **setOutputKey**  
    **public** void **setOutputKey**(java.lang.String outputKey)

### 4.3 Class ArrayVariance

This operator calculates the rms of the array specified by the key

#### 4.3.1 Declaration

```
public class ArrayVariance  
extends java.lang.Object
```

#### 4.3.2 Constructors

- **ArrayVariance**  
    **public** **ArrayVariance**()

#### 4.3.3 Methods

- **getKey**  
    **public** java.lang.String **getKey**()
- **getOutputKey**  
    **public** java.lang.String **getOutputKey**()
- **process**  
    **public** Data **process**(Data input)

- **setKey**  
public void setKey(java.lang.String key)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)

#### 4.4 Class CreateHistogram

Takes a float[] and returns an int[]

##### 4.4.1 Declaration

```
public class CreateHistogram  
extends java.lang.Object
```

##### 4.4.2 Constructors

- **CreateHistogram**  
public CreateHistogram()

##### 4.4.3 Methods

- **finish**  
public void finish() throws java.lang.Exception
- **getKey**  
public java.lang.String getKey()
- **getMax**  
public float getMax()
- **getMin**  
public float getMin()
- **getNumberOfBins**  
public int getNumberOfBins()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **init**  
public void init(ProcessContext context)
- **process**  
public Data process(Data input)
- **resetState**  
public void resetState() throws java.lang.Exception
- **setKey**  
public void setKey(java.lang.String key)

- **setMax**  
public void setMax(float maxBin)
- **setMin**  
public void setMin(float minbin)
- **setNumberOfBins**  
public void setNumberOfBins(int numberOfBins)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)

## 4.5 Class PixelAverage

This operator calculates the average of all the slices in each Pixel and stores the result as a double array.

### 4.5.1 Declaration

```
public class PixelAverage
extends fact.utils.SimpleFactPixelProcessor (in 2.9, page 10)
```

### 4.5.2 Constructors

- **PixelAverage**  
public PixelAverage()

### 4.5.3 Methods

- **processPixel**  
public float processPixel(float[] pixelData)  
– See also  
\* stream.DataProcessor#process(stream.Data)

## 4.6 Class PixelDistribution2D

### 4.6.1 Declaration

```
public class PixelDistribution2D
extends java.lang.Object
implements java.io.Serializable
```

### 4.6.2 Constructors

- **PixelDistribution2D**  
public PixelDistribution2D(double varianceX, double varianceY, double covariance, double centerX, double centerY, double eigenVarianceX, double eigenVarianceY, double angle, double sumOfWeights)

- **PixelDistribution2D**

```
public PixelDistribution2D(double varianceX, double varianceY, double
covariance, double centerX, double centerY, double eigenVarianceX,
double eigenVarianceY, double eigenSkewnessX, double eigenSkewnessY,
double eigenKurtosisX, double eigenKurtosisY, double angle, double
sumOfWeights)
```

#### 4.6.3 Methods

- **getAngle**  
public double getAngle()
- **getCenterX**  
public double getCenterX()
- **getCenterY**  
public double getCenterY()
- **getCovariance**  
public double getCovariance()
- **getEigenDeviationX**  
public double getEigenDeviationX()
- **getEigenDeviationY**  
public double getEigenDeviationY()
- **getEigenKurtosisX**  
public double getEigenKurtosisX()
- **getEigenKurtosisY**  
public double getEigenKurtosisY()
- **getEigenSkewnessX**  
public double getEigenSkewnessX()
- **getEigenSkewnessY**  
public double getEigenSkewnessY()
- **getEigenVarianceX**  
public double getEigenVarianceX()
- **getEigenVarianceY**  
public double getEigenVarianceY()
- **getLength**  
public double getLength()
- **getSize**  
public double getSize()

- **getVarianceX**  
public double getVarianceX()
- **getVarianceY**  
public double getVarianceY()
- **getWidth**  
public double getWidth()
- **setAngle**  
public void setAngle(double angle)
- **setCenterX**  
public void setCenterX(double centerX)
- **setCenterY**  
public void setCenterY(double centerY)
- **setCovariance**  
public void setCovariance(double covariance)
- **setEigenKurtosisX**  
public void setEigenKurtosisX(double eigenKurtosisX)
- **setEigenKurtosisY**  
public void setEigenKurtosisY(double eigenKurtosisY)
- **setEigenSkewnessX**  
public void setEigenSkewnessX(double eigenSkewnessX)
- **setEigenSkewnessY**  
public void setEigenSkewnessY(double eigenSkewnessY)
- **setEigenVarianceX**  
public void setEigenVarianceX(double eigenVarianceX)
- **setEigenVarianceY**  
public void setEigenVarianceY(double eigenVarianceY)
- **setSize**  
public void setSize(double size)
- **setVarianceX**  
public void setVarianceX(double varianceX)
- **setVarianceY**  
public void setVarianceY(double varianceY)

## 4.7 Class Quantiles

### 4.7.1 Declaration

```
public class Quantiles
extends java.lang.Object
```

#### 4.7.2 Constructors

- **Quantiles**  
public **Quantiles**(int slice, float[] image)

#### 4.7.3 Methods

- **getQuantile**  
public float **getQuantile**(double phi)
- **print**  
public void **print**(java.lang.Double[] phis)

### 4.8 Class StdDeviation

This Processor calculates the Standarddeviation of the slices in each pixel. It uses the Average Processor to calculate the average value ina pixel.

#### 4.8.1 Declaration

public abstract class StdDeviation  
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)

#### 4.8.2 Constructors

- **StdDeviation**  
public **StdDeviation**()

#### 4.8.3 Methods

- **processSeries**  
public float[] **processSeries**(float[] data)

## 5 Package fact.filter

*Package Contents*

*Page*

#### Classes

<b>AverageJumpRemoval</b> .....	<a href="#">25</a>
<b>DrsCalibration</b> .....	<a href="#">26</a>
This processor handles the DRS calibration.	
<b>ExponentialSmoothing</b> .....	<a href="#">27</a>
Calculates first Order exponential Smoothing Let y be the original Series and s be the smoothed one.	
<b>FirFilter</b> .....	<a href="#">27</a>
This class implements a simple Fir-Filter.	
<b>InterpolateBadPixel</b> .....	<a href="#">28</a>

This Processor interpolates all values for a broken Pixel by the average values of its neighboring Pixels.

**MotionDiff** .....28

This operator calculates between data[i] and data[i+offset] for each pixel in each event and stores the result as a float array named outputKey.

**MovingAverage** .....29

**MultiplyValues** .....29

This operator simply multiplies all values by the given factor.

**RemoveSpikesMars** .....30

Supposedly removes all spikes in the data.

**SliceNormalization** .....30

Normalizes all values in a pixel.

The processors in this package are called filters cause they take the raw fact data as input and put an array of the same length. They usually iterate over the array and smooth it or try to remove artifacts and similar things. These processors usually extend the `SimpleFactEventProcessor` class for more readable code.

## 5.1 Class AverageJumpRemoval

### 5.1.1 Declaration

```
public class AverageJumpRemoval
extends java.lang.Object
```

### 5.1.2 Constructors

- **AverageJumpRemoval**  
public **AverageJumpRemoval**()

### 5.1.3 Methods

- **getColor**  
public java.lang.String **getColor**()
- **getKey**  
public java.lang.String **getKey**()
- **getLimit**  
public int **getLimit**()
- **getOutputKey**  
public java.lang.String **getOutputKey**()
- **getThreshold**  
public double **getThreshold**()

- **process**

```
public Data process(Data input)
```

- **Description**

Each event contains the StartCellData array which contains the current startcell for each pixel. We save the previous 50 events in the previousStartCells previousStartCells. Which is a linked list containing 50 startcelldata arrays

- **setColor**

```
public void setColor(java.lang.String color)
```

- **setKey**

```
public void setKey(java.lang.String key)
```

- **setLimit**

```
public void setLimit(int limit)
```

- **setOutputKey**

```
public void setOutputKey(java.lang.String outputKey)
```

- **setThreshold**

```
public void setThreshold(double threshold)
```

## 5.2 Class DrsCalibration

This processor handles the DRS calibration. It requires a DRS data source either as File or URL and will read the DRS data from that. This data is then applied to all FactEvents processed by this class.

### 5.2.1 Declaration

```
public class DrsCalibration
extends java.lang.Object
```

### 5.2.2 Constructors

- **DrsCalibration**

```
public DrsCalibration()
```

### 5.2.3 Methods

- **applyDrsCalibration**

```
public float[] applyDrsCalibration(float[] data, float[] destination,
short[] startCellVector)
```

- **getColor**

```
public java.lang.String getColor()
```

- **getOutputKey**

```
public java.lang.String getOutputKey()
```



- **getPathToAuxfiles**  
`public java.lang.String getPathToAuxfiles()`
- **process**  
`public Data process(Data data)`
  - See also  
`* fact.data.FactProcessor#process(stream.Data)`
- **setColor**  
`public void setColor(java.lang.String color)`
- **setOutputKey**  
`public void setOutputKey(java.lang.String outputKey)`
- **setPathToAuxfiles**  
`public void setPathToAuxfiles(java.lang.String pathToAuxfiles)`
- **setUrl**  
`public void setUrl(java.lang.String urlString)`
- **setUrl**  
`public void setUrl(java.net.URL url)`

### 5.3 Class ExponentialSmoothing

Calculates first Order exponential Smoothing Let  $y$  be the original Series and  $s$  be the smoothed one.  $s_0 = y_0$   $s_i = \alpha * y_i + (1-\alpha) * s_{(i-1)}$  see [http://en.wikipedia.org/wiki/Exponential\\_smoothing](http://en.wikipedia.org/wiki/Exponential_smoothing)

#### 5.3.1 Declaration

```
public class ExponentialSmoothing
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

#### 5.3.2 Constructors

- **ExponentialSmoothing**  
`public ExponentialSmoothing()`

#### 5.3.3 Methods

- **getAlpha**  
`public float getAlpha()`
- **processSeries**  
`public float[] processSeries(float[] data)`
- **setAlpha**  
`public void setAlpha(float alpha)`

## 5.4 Class FirFilter

This class implements a simple Fir-Filter. See [http://en.wikipedia.org/wiki/Fir\\_filter](http://en.wikipedia.org/wiki/Fir_filter) for Details. The coefficients of the are stored in an array  $\{n, n-1, n-2, ..\}$ . Values outside of the data domain are treated as zeroes.

### 5.4.1 Declaration

```
public class FirFilter
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.4.2 Constructors

- **FirFilter**  
public **FirFilter()**

### 5.4.3 Methods

- **getCoefficients**  
public double[] **getCoefficients()**
- **getTemplate**  
public java.lang.String **getTemplate()**
- **processSeries**  
public float[] **processSeries(float[] data)**
- **setCoefficients**  
public void **setCoefficients(double[] coefficients)**
- **setTemplate**  
public void **setTemplate(java.lang.String templateString)**

## 5.5 Class InterpolateBadPixel

This Processor interpolates all values for a broken Pixel by the average values of its neighboring Pixels.

### 5.5.1 Declaration

```
public class InterpolateBadPixel
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.5.2 Constructors

- **InterpolateBadPixel**  
public **InterpolateBadPixel()**

### 5.5.3 Methods

- **getBadChidIds**  
`public int[] getBadChidIds()`
- **processSeries**  
`public float[] processSeries(float[] series)`
- **setBadChidIds**  
`public void setBadChidIds(java.lang.String[] badChIdStrings)`

## 5.6 Class MotionDiff

This operator calculates between `data[i]` and `data[i+offset]` for each pixel in each event and stores the result as a float array named `outputKey`. “br>if `i+offset` is greater or smaller the current window the first respectively the last value will be continued.

### 5.6.1 Declaration

```
public class MotionDiff
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.6.2 Constructors

- **MotionDiff**  
`public MotionDiff()`

### 5.6.3 Methods

- **getOffset**  
`public int getOffset()`
- **processSeries**  
`public float[] processSeries(float[] data)`
- **setOffset**  
`public void setOffset(int offset)`

## 5.7 Class MovingAverage

### 5.7.1 Declaration

```
public class MovingAverage
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.7.2 Constructors

- **MovingAverage**  
`public MovingAverage()`

### 5.7.3 Methods

- **getLength**  
`public int getLength()`
- **processSeries**  
`public float[] processSeries(float[] data)`
- **setLength**  
`public void setLength(int length)`

## 5.8 Class MultiplyValues

This operator simply multiplies all values by the given factor.

### 5.8.1 Declaration

```
public class MultiplyValues
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.8.2 Constructors

- **MultiplyValues**  
`public MultiplyValues()`

### 5.8.3 Methods

- **getFactor**  
`public float getFactor()`
- **processSeries**  
`public float[] processSeries(float[] data)`
- **setFactor**  
`public void setFactor(float threshold)`

## 5.9 Class RemoveSpikesMars

Supposedly removes all spikes in the data. Original algorithm by F.Temme. Takes a float array and creates a float array as output

### 5.9.1 Declaration

```
public class RemoveSpikesMars
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.9.2 Constructors

- **RemoveSpikesMars**  
`public RemoveSpikesMars()`

### 5.9.3 Methods

- **getTopSlope**  
public float **getTopSlope()**
- **processSeries**  
public float[] **processSeries(float[] data)**
- **setTopSlope**  
public void **setTopSlope(float topSlope)**

## 5.10 Class SliceNormalization

Normalizes all values in a pixel. That means only 0 value 1 are should be output.

### 5.10.1 Declaration

```
public class SliceNormalization
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 5.10.2 Constructors

- **SliceNormalization**  
public **SliceNormalization()**

### 5.10.3 Methods

- **processSeries**  
public float[] **processSeries(float[] value)**

## 6 Package fact.features

<i>Package Contents</i>	<i>Page</i>
<b>Classes</b>	
<b>DistributionFromShower</b> .....	<a href="#">32</a>
<b>HillasAlpha</b> .....	<a href="#">33</a>
<b>HillasConcentration</b> .....	<a href="#">34</a>
<b>HillasConcentration2</b> .....	<a href="#">34</a>
<b>HillasDistance</b> .....	<a href="#">35</a>
<b>HillasLength</b> .....	<a href="#">36</a>
<b>HillasParameter</b> .....	<a href="#">36</a>

Calculates the Hillas Parameter from the Ellipse.	
<b>HillasWidth</b> .....	38
blavsdflsfdslfs fsdfsdfsd	
Hallo ich bin markdown <b>Fett</b>	
$F(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$	
<b>Leakage</b> .....	39
<b>MaxAmplitude</b> .....	39
This processor simply calculates the maximum value for all time slices in each Pixel.	
<b>MaxAmplitudePosition</b> .....	40
This processor simply calculates the position of the maximum value for all time slices in each Pixel.	
<b>NumberOfIslands</b> .....	41
If key refers to an int[] of showerpixel. this will calculate the number of islands	
<b>NumberOfPixelInShower</b> .....	41
<b>PhotonCharge</b> .....	42
This processor Calculates PhotonCharge by doing the following: 1.	
<b>RisingEdge</b> .....	43
<b>ShowerEllipse</b> .....	43
<b>Size</b> .....	45
<b>SizeInInterval</b> .....	45
Sum up all the weights for pixel between the max and min values.	
<b>SourcePosition</b> .....	46
This is supposed to calculate the position of the source in the camera.	
<b>TimeDependentParameter</b> .....	48
This class calculates time dependent parameters: The symbol '*' means implemented, '?'	

## 6.1 Class DistributionFromShower

### 6.1.1 Declaration

```
public class DistributionFromShower
extends java.lang.Object
```

### 6.1.2 Fields

- public float **mCenterOfGravityX**

- public float **mCenterOfGravityY**

### 6.1.3 Constructors

- **DistributionFromShower**  
public **DistributionFromShower()**

### 6.1.4 Methods

- **finish**  
public void **finish()** throws java.lang.Exception
- **getKey**  
public java.lang.String **getKey()**
- **getOutputKey**  
public java.lang.String **getOutputKey()**
- **getPixel**  
public java.lang.String **getPixel()**
- **getWeights**  
public java.lang.String **getWeights()**
- **init**  
public void **init**(ProcessContext context) throws java.lang.Exception
- **process**  
public Data **process**(Data input)
- **resetState**  
public void **resetState()** throws java.lang.Exception
- **setKey**  
public void **setKey**(java.lang.String key)
- **setOutputKey**  
public void **setOutputKey**(java.lang.String outputKey)
- **setPixel**  
public void **setPixel**(java.lang.String pixel)
- **setWeights**  
public void **setWeights**(java.lang.String wheights)

## 6.2 Class HillasAlpha

### 6.2.1 Declaration

```
public class HillasAlpha  
extends java.lang.Object
```

### 6.2.2 Constructors

- **HillasAlpha**  
public HillasAlpha()

### 6.2.3 Methods

- **getDistribution**  
public java.lang.String getDistribution()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **getSourcePosition**  
public java.lang.String getSourcePosition()
- **process**  
public Data process(Data input)
- **setDistribution**  
public void setDistribution(java.lang.String distribution)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setSourcePosition**  
public void setSourcePosition(java.lang.String sourcePosition)

## 6.3 Class HillasConcentration

### 6.3.1 Declaration

```
public class HillasConcentration  
extends java.lang.Object
```

### 6.3.2 Constructors

- **HillasConcentration**  
public HillasConcentration()

### 6.3.3 Methods

- **getOutputKey**  
public java.lang.String getOutputKey()
- **getShower**  
public java.lang.String getShower()
- **getWeights**  
public java.lang.String getWeights()



- **process**  
public Data process(Data input)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setShower**  
public void setShower(java.lang.String shower)
- **setWeights**  
public void setWeights(java.lang.String weights)

## 6.4 Class HillasConcentration2

### 6.4.1 Declaration

```
public class HillasConcentration2  
extends java.lang.Object
```

### 6.4.2 Constructors

- **HillasConcentration2**  
public HillasConcentration2()

### 6.4.3 Methods

- **getOutputKey**  
public java.lang.String getOutputKey()
- **getShower**  
public java.lang.String getShower()
- **getSize**  
public java.lang.String getSize()
- **getWeights**  
public java.lang.String getWeights()
- **process**  
public Data process(Data input)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setShower**  
public void setShower(java.lang.String shower)
- **setSize**  
public void setSize(java.lang.String size)
- **setWeights**  
public void setWeights(java.lang.String weights)

## 6.5 Class HillasDistance

### 6.5.1 Declaration

```
public class HillasDistance  
extends java.lang.Object
```

### 6.5.2 Constructors

- **HillasDistance**  
public HillasDistance()

### 6.5.3 Methods

- **getDistribution**  
public java.lang.String getDistribution()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **getSourcePosition**  
public java.lang.String getSourcePosition()
- **process**  
public Data process(Data input)
- **setDistribution**  
public void setDistribution(java.lang.String distribution)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setSourcePosition**  
public void setSourcePosition(java.lang.String sourcePosition)

## 6.6 Class HillasLength

### 6.6.1 Declaration

```
public class HillasLength  
extends java.lang.Object
```

### 6.6.2 Constructors

- **HillasLength**  
public HillasLength()

### 6.6.3 Methods

- **getDistribution**  
public java.lang.String getDistribution()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **process**  
public Data process(Data input)
- **setDistribution**  
public void setDistribution(java.lang.String distribution)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)

## 6.7 Class HillasParameter

Calculates the Hillas Parameter from the Ellipse. Some processor to identify showerpixels has to be run before. void HillasParameter::CalculateParameter() { if (mVerbosityLevel >9) { cout "HillasParameter::CalculateParameter() called" endl; } SetAllParameterToZero(); CalculateNumberOfIslands(); CalculateSize(); CalculateLeakage(); CalculateConcentration(); CalculateCenterOfGravity(); CalculateEllipse(); CalculateSourceParameter(); CalculateAsymmetry(); }

### 6.7.1 Declaration

```
public class HillasParameter
extends java.lang.Object
```

### 6.7.2 Constructors

- **HillasParameter**  
public HillasParameter()

### 6.7.3 Methods

- **calculateSize**  
public float calculateSize(int[] showerPixelArray, float[] photonCharges)  
  
– **Description**  
calculates the sum of all photoncharges for all showerPixel
- **getKey**  
public java.lang.String getKey()
- **getMinSize**  
public int getMinSize()

- **getOutputKey**  
public java.lang.String getOutputKey()
- **getPhotonEquivalent**  
public java.lang.String getPhotonEquivalent()
- **getPixels**  
public java.lang.String getPixels()
- **getShowerThreshold**  
public double getShowerThreshold()
- **getSourcePosition**  
public java.lang.String getSourcePosition()
- **process**  
public Data process(Data input)
  - See also
    - \* stream.DataProcessor#process(stream.Data)
- **processEvent**  
public void processEvent(Data input, java.lang.String key)
- **processSeries**  
public void processSeries(float[] value)
- **setKey**  
public void setKey(java.lang.String key)
- **setMinSize**  
public void setMinSize(int minSize)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setPhotonEquivalent**  
public void setPhotonEquivalent(java.lang.String photonEquivalent)
- **setPixels**  
public void setPixels(java.lang.String pixels)
- **setShowerThreshold**  
public void setShowerThreshold(double showerThreshold)
- **setSourcePosition**  
public void setSourcePosition(java.lang.String sourcePosition)

## 6.8 Class HillasWidth

blavsdflsfdslfs fsdfsdfsd

Hallo ich bin markdown **Fett**

$$F(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz$$

### 6.8.1 Declaration

```
public class HillasWidth
extends java.lang.Object
```

### 6.8.2 Constructors

- **HillasWidth**  
public HillasWidth()

### 6.8.3 Methods

- **getDistribution**  
public java.lang.String getDistribution()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **getSourcePosition**  
public java.lang.String getSourcePosition()
- **process**  
public Data process(Data input)
- **setDistribution**  
public void setDistribution(java.lang.String distribution)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setSourcePosition**  
public void setSourcePosition(java.lang.String sourcePosition)

## 6.9 Class Leakage

### 6.9.1 Declaration

```
public class Leakage
extends java.lang.Object
```

### 6.9.2 Constructors

- **Leakage**  
`public Leakage()`

### 6.9.3 Methods

- **getOutputKey**  
`public java.lang.String getOutputKey()`
- **getShower**  
`public java.lang.String getShower()`
- **getWeights**  
`public java.lang.String getWeights()`
- **process**  
`public Data process(Data input)`
- **setOutputKey**  
`public void setOutputKey(java.lang.String outputKey)`
- **setShower**  
`public void setShower(java.lang.String shower)`
- **setWeights**  
`public void setWeights(java.lang.String weights)`

## 6.10 Class MaxAmplitude

This processor simply calculates the maximum value for all time slices in each Pixel. The output is a float array with an entry for each Pixel.

### 6.10.1 Declaration

```
public class MaxAmplitude
extends fact.utils.SimpleFactPixelProcessor (in 2.9, page 10)
```

### 6.10.2 Constructors

- **MaxAmplitude**  
`public MaxAmplitude()`

### 6.10.3 Methods

- **getMaxValue**  
`public float getMaxValue()`
- **getMinValue**  
`public float getMinValue()`

- **processPixel**  
public abstract float processPixel(float[] pixelData)
- **setMaxValue**  
public void setMaxValue(float max\_value)
- **setMinValue**  
public void setMinValue(float min\_value)

## 6.11 Class MaxAmplitudePosition

This processor simply calculates the position of the maximum value for all time slices in each Pixel. outputs an int array

### 6.11.1 Declaration

```
public class MaxAmplitudePosition
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 6.11.2 Constructors

- **MaxAmplitudePosition**  
public MaxAmplitudePosition()

### 6.11.3 Methods

- **getMaxValue**  
public float getMaxValue()
- **getMinValue**  
public float getMinValue()
- **processSeries**  
public int[] processSeries(float[] data)
- **setMaxValue**  
public void setMaxValue(float max\_value)
- **setMinValue**  
public void setMinValue(float min\_value)

## 6.12 Class NumberOfIslands

If key refers to an int[] of showerpixel. this will calculate the number of islands

### 6.12.1 Declaration

```
public class NumberOfIslands
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 6.12.2 Constructors

- **NumberOfIslands**  
`public NumberOfIslands()`

### 6.12.3 Methods

- **processSeries**  
`public java.lang.Integer processSeries(int[] data)`

## 6.13 Class NumberOfPixelInShower

### 6.13.1 Declaration

```
public class NumberOfPixelInShower
extends java.lang.Object
```

### 6.13.2 Constructors

- **NumberOfPixelInShower**  
`public NumberOfPixelInShower()`

### 6.13.3 Methods

- **getOutputKey**  
`public java.lang.String getOutputKey()`
- **getShowerKey**  
`public java.lang.String getShowerKey()`
- **process**  
`public Data process(Data input)`
- **setOutputKey**  
`public void setOutputKey(java.lang.String outputKey)`
- **setShowerKey**  
`public void setShowerKey(java.lang.String showerKey)`

## 6.14 Class PhotonCharge

This processor Calculates PhotonCharge by doing the following: 1. Use the MaxAmplitude Processor to find the maximum Value in the slices. 2. In the area between amplitudePosition...amplitudePositon-25 search for the position having 0.5 of the original max-Amplitude. 3. Now for some reason sum up all slices between half\_max\_pos and half\_max\_pos + 30. 4. Divide the sum by the integralGain and save the result. Treatment of edge Cases is currently very arbitrary since Pixels with these values should not be considered as showerPixels anyways.



### 6.14.1 Declaration

```
public class PhotonCharge
extends java.lang.Object
```

### 6.14.2 Constructors

- **PhotonCharge**  
public **PhotonCharge()**

### 6.14.3 Methods

- **getAlpha**  
public int **getAlpha()**
- **getColor**  
public java.lang.String **getColor()**
- **getIntegralGain**  
public float **getIntegralGain()**
- **getKey**  
public java.lang.String **getKey()**
- **getOutputKey**  
public java.lang.String **getOutputKey()**
- **process**  
public Data **process(Data input)**
- **setAlpha**  
public void **setAlpha(int alpha)**
- **setColor**  
public void **setColor(java.lang.String color)**
- **setIntegralGain**  
public void **setIntegralGain(float integralGain)**
- **setKey**  
public void **setKey(java.lang.String key)**
- **setOutputKey**  
public void **setOutputKey(java.lang.String outputKey)**
- **setPositions**  
public void **setPositions(java.lang.String positions)**

## 6.15 Class RisingEdge

### 6.15.1 Declaration

```
public class RisingEdge
extends fact.utils.SimpleFactEventProcessor (in 2.8, page 9)
```

### 6.15.2 Constructors

- **RisingEdge**  
public RisingEdge()

### 6.15.3 Methods

- **getSearchWindowLeft**  
public int getSearchWindowLeft()
- **getSearchWindowRight**  
public int getSearchWindowRight()
- **processSeries**  
public int[] processSeries(float[] data)
- **setSearchWindowLeft**  
public void setSearchWindowLeft(int searchWindowLeft)
- **setSearchWindowRight**  
public void setSearchWindowRight(int searchWindowRight)

## 6.16 Class ShowerEllipse

### 6.16.1 Declaration

```
public class ShowerEllipse
extends java.lang.Object
```

### 6.16.2 Fields

- public double **centerX**
- public double **centerY**
- public float **mCenterOfGravityX**
- public float **mCenterOfGravityY**
- public double **length**
- public double **width**
- public double **area**

- public double **mDelta**
- public double **mAsymmetryLong**
- public double **mAsymmetryTrans**
- public double **mDistance**
- public float **alpha**
- public float **alphaOff1**
- public float **alphaOff2**
- public float **alphaOff3**

### 6.16.3 Constructors

- **ShowerEllipse**  
 public **ShowerEllipse**(int[] showerPixel, float[] photonCharge, float source, float source2)

### 6.16.4 Methods

- **calculateAlpha**  
 public float **calculateAlpha**(double source\_x, double source\_y)
- **calculateAsymmetry**  
 public void **calculateAsymmetry**()
- **calculateCenterOfGravity**  
 public void **calculateCenterOfGravity**(int[] showerPixel)
- **calculateEllipseMars**  
 public void **calculateEllipseMars**()
  - **Description**  
 original code copied from F. Temme's DoFact program.
- **calculateSourceParameter**  
 public void **calculateSourceParameter**()

## 6.17 Class Size

### 6.17.1 Declaration

```
public class Size
extends java.lang.Object
```

### 6.17.2 Constructors

- **Size**  
 public **Size**()

### 6.17.3 Methods

- **getOutputKey**  
public java.lang.String getOutputKey()
- **getPhotonChargeKey**  
public java.lang.String getPhotonChargeKey()
- **getShowerKey**  
public java.lang.String getShowerKey()
- **process**  
public Data process(Data input)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setPhotonChargeKey**  
public void setPhotonChargeKey(java.lang.String photonChargeKey)
- **setShowerKey**  
public void setShowerKey(java.lang.String showerKey)

### 6.18 Class SizeInInterval

Sum up all the weights for pixel between the max and min values. The output of this processor is the sum of the pixel weights in the shower array iff the weight is >min and max.

#### 6.18.1 Declaration

```
public class SizeInInterval
extends java.lang.Object
```

#### 6.18.2 Constructors

- **SizeInInterval**  
public SizeInInterval()

#### 6.18.3 Methods

- **getMax**  
public float getMax()
- **getMin**  
public float getMin()
- **getOutputKey**  
public java.lang.String getOutputKey()
- **getPhotonChargeKey**  
public java.lang.String getPhotonChargeKey()

- **getShowerKey**  
public java.lang.String getShowerKey()
- **process**  
public Data process(Data input)
- **setMax**  
public void setMax(float max)
- **setMin**  
public void setMin(float min)
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)
- **setPhotonChargeKey**  
public void setPhotonChargeKey(java.lang.String photonChargeKey)
- **setShowerKey**  
public void setShowerKey(java.lang.String showerKey)

### 6.19 Class SourcePosition

This is supposed to calculate the position of the source in the camera. The Telescope usually does not look directly at the source but somewhere close by. That means the image of the source projected by the mirrors onto the camera is not exactly in the center but at some point (X,Y). This point will be called source position from now on. The point (0.0, 0.0) is the center of the camera. In order to calculate the source position we need to know where the telescope is looking. This data is written by the telescope drive system into an auxiliary .fits file called DRIVE\_TRACKING\_POSITION.

#### 6.19.1 Declaration

```
public class SourcePosition
extends java.lang.Object
```

#### 6.19.2 Constructors

- **SourcePosition**  
public SourcePosition()

#### 6.19.3 Methods

- **finish**  
public void finish() throws java.lang.Exception
- **getOutputKey**  
public java.lang.String getOutputKey()
- **getPhysicalSource**  
public java.lang.String getPhysicalSource()

- **getSourceDeclination**  
public double getSourceDeclination()
- **getSourceRightAscension**  
public double getSourceRightAscension()
- **getX**  
public java.lang.Float getX()
- **getY**  
public java.lang.Float getY()
- **init**  
public void init(ProcessContext arg0) throws java.lang.Exception

- **Description**

In the init method we read the complete TRACKING\_POSITION file and save the values in the locList. For the calculation of the appropriate sky coordinates (that is Azimuth and Zenith) we only need the values "Time", "Ra" and "Dec". There are also values for "Az" and "Zd" in file. These are calculated by the drive system itself. They can be used for a sanity check. These values differ by what seems to be a constant amount in both Az and Zd. About 1 to 3 degrees for the files that I checked. The time unit in the TRACKING file is in unixtime/86400.0. Its still called MJD for some reason. The correct conversion would be:  $mjd = timestamp/86400.0 + 2440587.5d$  for some effing reason. To get the correct coordinates we have to do it like this:  $mjd = timestamp/86400.0 + 2440587.0d$  Thats an offset of half a day.

- **process**  
public Data process(Data data)

- **Description**

Here we read the eventtime from the current dataitem and convert it to 1. unixtime 2. mjd 3. gmst The conversion steps are necessary because I stole the mjd2gmst conversion from Fabian Temme and dont know how to get gmst directly from unixtime. The unixtimestamp in the data file is saved as an array with two elements. {seconds, miroseconds} it is unclear what to do with the second one. I simply used the sum of both in seconds.. Eventhough the numbers are small enough to NOT make a difference anyways. After reading the EventTime from the data we check which datapoint from the slowcontroll file we have to use by comparing the times. We use the point closest in time to the current dataitem.

- **See also**

\* fact.data.FactProcessor#process(stream.Data)

- **resetState**  
public void resetState() throws java.lang.Exception
- **setOutputKey**  
public void setOutputKey(java.lang.String outputKey)

- **setPhysicalSource**  
public void setPhysicalSource(java.lang.String physicalSource)
- **setSourceDeclination**  
public void setSourceDeclination(double sourceDeclination)
- **setSourceRightAscension**  
public void setSourceRightAscension(double sourceRightAscension)
- **setUrl**  
public void setUrl(java.lang.String urlString)
- **setUrl**  
public void setUrl(java.net.URL url)
- **setX**  
public void setX(java.lang.Float x)
- **setY**  
public void setY(java.lang.Float y)

## 6.20 Class TimeDependentParameter

This class calculates time dependent parameters: The symbol '\*' means implemented, '?' means in progress or unsure  
 Calculated helper values \* The COGX and COGY for every slice of event  
 \* The  $\text{COG}\{X, Y \text{ and } \sqrt{X^2 + Y^2}\}$ -Velocity for every slice transition ? The velocity in shower coordinate system  
 \* The variance of COGX and COGY ? The variance of COG-Velocity  
 Calculated values for separation ? MaxVelocity in both systems ? MeanVelocity and error in interval [arrival - 10 slices, arrival + 10 slices] ? Mean Velocity at position of minimal variance of COG +/- 5 slices ? "Rhode"-Parameter (working title) This parameter is a measure for the source-gamma-ness of an event

### 6.20.1 Declaration

```
public class TimeDependentParameter
extends java.lang.Object
```

### 6.20.2 Constructors

- **TimeDependentParameter**  
public TimeDependentParameter()

### 6.20.3 Methods

- **getArrivalTime**  
public java.lang.String getArrivalTime()
- **getDataCalibrated**  
public java.lang.String getDataCalibrated()

- **getShowerPixel**  
public java.lang.String getShowerPixel()
- **getSourcePosition**  
public java.lang.String getSourcePosition()
- **process**  
public Data process(Data input)
- **setArrivalTime**  
public void setArrivalTime(java.lang.String arrivalTime)
- **setDataCalibrated**  
public void setDataCalibrated(java.lang.String dataCalibrated)
- **setShowerPixel**  
public void setShowerPixel(java.lang.String showerPixel)
- **setSourcePosition**  
public void setSourcePosition(java.lang.String sourcePosition)

## 7 Package fact.io

*Package Contents*

*Page*

### Classes

<b>BinaryFactWriter</b> .....	50
This class writes out FACT events in binary format.	
<b>ByteChunkStream</b> .....	51
This class implements a fast byte-oriented stream of byte chunks.	
<b>CreateAnimatedGif</b> .....	52
<b>FitsEventSplitter</b> .....	53
<b>FitsStream</b> .....	53
<b>FitsStream.FitsHeader</b> .....	54
<b>ReadMCcsv</b> .....	55
<b>RootASCIIWriter</b> .....	56
This class writes out FACT events in CSV format.	
<b>SerializedEventStream</b> .....	57
<b>WeatherStream</b> .....	58
<b>Weird8ByteChunkStream</b> .....	59



<b>WStream</b> .....	<a href="#">59</a>
----------------------	--------------------

This package contains classes which provide IO functionality. Usually they take some sort of Data provided by the FACT-Telescope and put out a Stream of **Data items**. These data items can then be analyzed by any class extending the **Processor** interface of the Streams-Framework.

## 7.1 Class BinaryFactWriter

This class writes out FACT events in binary format. The format for each event is exactly 1440 \* ROI double values. By default the data is expected to be contained in the "Data" property of the input.

### 7.1.1 Declaration

```
public class BinaryFactWriter
extends CsvWriter
```

### 7.1.2 Constructors

- **BinaryFactWriter**  
public **BinaryFactWriter**()

### 7.1.3 Methods

- **finish**  
public void **finish**() throws java.lang.Exception  
– **See also**  
\* stream.io.CsvWriter#close()
- **getKeys**  
public java.lang.String[] **getKeys**()  
– **Returns** – the key
- **init**  
public void **init**(ProcessContext ctx) throws java.lang.Exception  
– **See also**  
\* stream.io.CsvWriter#init(stream.ProcessContext)
- **process**  
public Data **process**(Data data)  
– **See also**  
\* stream.DataProcessor#process(stream.Data)
- **setKeys**  
public void **setKeys**(java.lang.String[] keys)

- **Parameters**

- \* **key** – the key to set

## 7.2 Class ByteChunkStream

This class implements a fast byte-oriented stream of byte chunks. The chunks are found by checking for a start-signature (i.e. byte array). The stream returns a sequence of data items, each holding a chunk of bytes.

### 7.2.1 Declaration

```
public abstract class ByteChunkStream
extends AbstractStream
```

### 7.2.2 All known subclasses

WStream (in 7.12, page 59), WeatherStream (in 7.10, page 58)

### 7.2.3 Fields

- public static final byte **GIF\_SIGNATURE**
- public static final byte **JPG\_SIGNATURE**
- public static final int **DEFAULT\_BUFFER\_SIZE**

### 7.2.4 Constructors

- **ByteChunkStream**  

```
public ByteChunkStream(SourceURL url, byte[] signature) throws
java.lang.Exception
```

### 7.2.5 Methods

- **close**  

```
public void close() throws java.lang.Exception
```

  - **See also**
    - \* stream.io.DataStream#close()
- **getBufferSize**  

```
public int getBufferSize()
```

  - **Returns** – the bufferSize
- **init**  

```
public void init() throws java.lang.Exception
```

  - **See also**
    - \* stream.io.AbstractDataStream#init()

- **readNext**  
`public synchronized Data readNext() throws java.lang.Exception`  
  - **See also**  
    - \* `stream.io.AbstractDataStream#readItem(stream.data.Data)`
- **setBufferSize**  
`public void setBufferSize(ByteSize bufferSize)`  
  - **Parameters**  
    - \* `bufferSize` – the `bufferSize` to set

### 7.3 Class CreateAnimatedGif

#### 7.3.1 Declaration

```
public class CreateAnimatedGif
extends java.lang.Object
```

#### 7.3.2 Constructors

- **CreateAnimatedGif**  
`public CreateAnimatedGif()`

#### 7.3.3 Methods

- **createAnimatedGif**  
`public static java.io.File createAnimatedGif(java.io.File out,  
java.util.Date date, java.lang.Integer run, Data event, int start, int  
end, int step)`
- **main**  
`public static void main(java.lang.String[] args) throws  
java.lang.Exception`  
  - **Parameters**  
    - \* `args` –

### 7.4 Class FitsEventSplitter

#### 7.4.1 Declaration

```
public class FitsEventSplitter
extends java.lang.Object
```

#### 7.4.2 Constructors

- **FitsEventSplitter**  
`public FitsEventSplitter()`

### 7.4.3 Methods

- **main**  
public static void **main**(java.lang.String[] args) throws  
java.lang.Exception
  - **Parameters**
    - \* args –
- **store**  
public static void **store**(Data item, java.io.File file) throws  
java.lang.Exception

## 7.5 Class FitsStream

### 7.5.1 Declaration

```
public class FitsStream  
extends AbstractStream
```

### 7.5.2 Constructors

- **FitsStream**  
public **FitsStream**(SourceURL url)

### 7.5.3 Methods

- **getBufferSize**  
public int **getBufferSize**()
- **init**  
public void **init**() throws java.lang.Exception
  - **Description**

This consists of 3 steps 1. Get the size of the fits header. A header contains 2 subheaders. We ignore the first one and read the second one until we reach "END" From the line read we get the header size since we know its a multiple of the blocksize (2880) 2. Then we parse the headers for the number of fields the fits file contains. 3. Each file has a name, datatype and a number of elements. The header is parsed again
- **readHeader**  
public FitsStream.FitsHeader **readHeader**(java.io.InputStream in) throws  
java.io.IOException
- **readNext**  
public Data **readNext**() throws java.lang.Exception

- **Description**

this parses an event from the datastream and the bytearray in case we read alot of shorts(more than 128) We use a NIO buffer to load a complete bunch of bytes and intepret them as a short array

- **setBufferSize**  
public void setBufferSize(int bufferSize)

## 7.6 Class FitsStream.FitsHeader

### 7.6.1 Declaration

```
public class FitsStream.FitsHeader
extends java.lang.Object
```

### 7.6.2 Constructors

- **FitsStream.FitsHeader**  
public FitsStream.FitsHeader(byte[] data)

### 7.6.3 Methods

- **getLines**  
public java.lang.String[] getLines()
- **toString**  
public java.lang.String toString()

## 7.7 Class ReadMCcsv

### 7.7.1 Declaration

```
public class ReadMCcsv
extends AbstractLineStream
```

### 7.7.2 Fields

- public static java.lang.String **newline**

### 7.7.3 Constructors

- **ReadMCcsv**  
public ReadMCcsv(SourceURL url)

### 7.7.4 Methods

- **getCommentString**  
public java.lang.String getCommentString()

- **getDelimiter**  
public java.lang.String getDelimiter()
- **getFileUrl**  
public java.lang.String getFileUrl()
- **getKeys**  
public java.lang.String[] getKeys()
- **getPreprocessors**  
public java.util.List getPreprocessors()
- **getTemplate**  
public java.lang.String getTemplate()
- **init**  
public void init() throws java.lang.Exception
- **readNext**  
public Data readNext() throws java.lang.Exception
- **readNext**  
public Data readNext(Data datum) throws java.lang.Exception
- **setCommentString**  
public void setCommentString(java.lang.String commentString)
- **setDelimiter**  
public void setDelimiter(java.lang.String delimiter)
- **setFileUrl**  
public void setFileUrl(java.lang.String gnuPlotPath)
- **setKeys**  
public void setKeys(java.lang.String[] keys)
- **setTemplate**  
public void setTemplate(java.lang.String template)

## 7.8 Class RootASCIIWriter

This class writes out FACT events in CSV format. The format for each event is exactly 1440 \* ROI double values. By default the data is expected to be contained in the "Data" property of the input.

### 7.8.1 Declaration

```
public class RootASCIIWriter
extends CsvWriter
```

### 7.8.2 Constructors

- **RootASCIIWriter**  
`public RootASCIIWriter()`

### 7.8.3 Methods

- **finish**  
`public void finish() throws java.lang.Exception`
  - **See also**
    - \* `stream.io.CsvWriter#close()`
- **getKeys**  
`public java.lang.String[] getKeys()`
  - **Returns** – the key
- **init**  
`public void init(ProcessContext ctx) throws java.lang.Exception`
- **isWriteTreeDescriptor**  
`public boolean isWriteTreeDescriptor()`
- **process**  
`public Data process(Data data)`
  - **See also**
    - \* `stream.DataProcessor#process(stream.Data)`
- **setKeys**  
`public void setKeys(java.lang.String[] keys)`
  - **Parameters**
    - \* `key` – the key to set
- **setWriteTreeDescriptor**  
`public void setWriteTreeDescriptor(boolean writeTreeDescriptor)`

## 7.9 Class SerializedEventStream

### 7.9.1 Declaration

```
public class SerializedEventStream
extends AbstractStream
```

### 7.9.2 Constructors

- **SerializedEventStream**  
`public SerializedEventStream(java.io.File file) throws java.lang.Exception`
  - **Parameters**
    - \* `url` –
  - **Throws**
    - \* `java.lang.Exception` –
- **SerializedEventStream**  
`public SerializedEventStream(SourceURL sUrl) throws java.lang.Exception`

### 7.9.3 Methods

- **close**  
`public void close()`
  - **See also**
    - \* `stream.io.DataStream#getPreprocessors()`
- **getId**  
`public java.lang.String getId()`
- **init**  
`public void init() throws java.lang.Exception`
- **readNext**  
`public Data readNext() throws java.lang.Exception`
  - **See also**
    - \* `stream.io.DataStream#readNext()`
- **readNext**  
`public Data readNext(Data datum) throws java.lang.Exception`
  - **See also**
    - \* `stream.io.DataStream#readNext(stream.Data)`
- **setId**  
`public void setId(java.lang.String id)`

## 7.10 Class WeatherStream

### 7.10.1 Declaration

```
public class WeatherStream
extends fact.io.ByteChunkStream (in 7.2, page 51)
```



### 7.10.2 Fields

- `public static final int MAX_MESSAGE_LENGTH`

### 7.10.3 Constructors

- **WeatherStream**  
`public WeatherStream(SourceURL url) throws java.lang.Exception`

### 7.10.4 Methods

- **checksum**  
`public void checksum(byte[] msg)`
- **getHex**  
`public static java.lang.String getHex(byte[] bytes, int len)`
- **getHex**  
`public static java.lang.String getHex(byte[] bytes, int off, int len)`
- **init**  
`public void init() throws java.lang.Exception`
  - **See also**
    - \* `stream.io.AbstractStream#init()`
- **isDebug**  
`public boolean isDebug()`
  - **Returns** – the debug
- **read**  
`public Data read() throws java.lang.Exception`
  - **See also**
    - \* `stream.io.AbstractStream#readNext()`
- **readMessage**  
`public byte[] readMessage() throws java.lang.Exception`
- **setDebug**  
`public void setDebug(boolean debug)`
  - **Parameters**
    - \* `debug` – the debug to set

## 7.11 Class Weird8ByteChunkStream

### 7.11.1 Declaration

```
public class Weird8ByteChunkStream
extends java.io.InputStream
```

### 7.11.2 Constructors

- **Weird8ByteChunkStream**  
`public Weird8ByteChunkStream(java.io.InputStream in)`

### 7.11.3 Methods

- **read**  
`public int read() throws java.io.IOException`
  - See also
    - \* [java.io.InputStream.read\(\)](#)
- **sleep**  
`public void sleep(int ms)`

## 7.12 Class WStream

### 7.12.1 Declaration

`public class WStream`  
`extends fact.io.ByteChunkStream` (in [7.2](#), page [51](#))

### 7.12.2 Fields

- `public static final byte SIG`

### 7.12.3 Constructors

- **WStream**  
`public WStream(SourceURL url) throws java.lang.Exception`