Fact-Tools Documentation

Fact-Tools Dokumentation mit JavaDocs

Christian Bockermann Kai Bruegge

June 25, 2013

Contents

Class Hierarchy **2**

1 Package fact.image.monitors **4**

1.1 Class AverageBarPlotter . . . 5
1.1.1 Declaration . . . 5
1.1.2 Constructors . . . 5
1.1.3 Methods . . . 5
1.2 Class BarPlotPanel . . . 6
1.2.1 Declaration . . . 6
1.2.2 Constructors . . . 6
1.2.3 Methods . . . 6
1.3 Class CameraPlotter . . . 6
1.3.1 Declaration . . . 6
1.3.2 Constructors . . . 6
1.3.3 Methods . . . 6
1.4 Class CustomBarRender . . . 7
1.4.1 Declaration . . . 7
1.4.2 Constructors . . . 7
1.4.3 Methods . . . 7
1.5 Class CustomStatisticalBarRender . . . 7
1.5.1 Declaration . . . 8
1.5.2 Constructors . . . 8
1.5.3 Methods . . . 8
1.6 Class HistogramPlotter . . . 8
1.6.1 Declaration . . . 8
1.6.2 Constructors . . . 8
1.6.3 Methods . . . 8
1.7 Class LiveCameraPlotter . . . 9
1.7.1 Declaration . . . 9
1.7.2 Constructors . . . 9
1.7.3 Methods . . . 9
1.8 Class ScatterPlotPanel . . . 10
1.8.1 Declaration . . . 10
1.8.2 Constructors . . . 10
1.8.3 Methods . . . 10
1.9 Class ScatterPlotter . . . 10

1.9.1 Declaration . . . 10
1.9.2 Constructors . . . 10
1.9.3 Methods . . . 11
1.10 Class ShowerCameraPlotter . . . 11
1.10.1 Declaration . . . 11
1.10.2 Constructors . . . 11
1.10.3 Methods . . . 12
1.11 Class StatusWindow . . . 12
1.11.1 Declaration . . . 12
1.11.2 Constructors . . . 12
1.11.3 Methods . . . 13

2 Package fact.io **14**
2.1 Class BinaryFactWriter . . . 15
2.1.1 Declaration . . . 15
2.1.2 Constructors . . . 15
2.1.3 Methods . . . 15
2.2 Class ByteChunkStream . . . 16
2.2.1 Declaration . . . 16
2.2.2 All known subclasses . . . 16
2.2.3 Fields . . . 16
2.2.4 Constructors . . . 16
2.2.5 Methods . . . 16
2.3 Class CreateAnimatedGif . . . 17
2.3.1 Declaration . . . 17
2.3.2 Constructors . . . 17
2.3.3 Methods . . . 17
2.4 Class FitsEventSplitter . . . 18
2.4.1 Declaration . . . 18
2.4.2 Constructors . . . 18
2.4.3 Methods . . . 18
2.5 Class FitsStream . . . 18
2.5.1 Declaration . . . 18
2.5.2 Constructors . . . 18
2.5.3 Methods . . . 18
2.6 Class FitsStream.FitsHeader . . . 19
2.6.1 Declaration . . . 19
2.6.2 Constructors . . . 19
2.6.3 Methods . . . 19
2.7 Class ReadMCcsv . . . 19
2.7.1 Declaration . . . 19
2.7.2 Fields . . . 19
2.7.3 Constructors . . . 19
2.7.4 Methods . . . 20
2.8 Class RootASCIIWriter . . . 20
2.8.1 Declaration . . . 21

2.8.2 Constructors . . . 21
2.8.3 Methods . . . 21
2.9 Class SerializedEventStream . . . 21
2.9.1 Declaration . . . 21
2.9.2 Constructors . . . 22
2.9.3 Methods . . . 22
2.10 Class WeatherStream . . . 22
2.10.1 Declaration . . . 22
2.10.2 Fields . . . 23
2.10.3 Constructors . . . 23
2.10.4 Methods . . . 23
2.11 Class Weird8ByteChunkStream . . . 23
2.11.1 Declaration . . . 23
2.11.2 Constructors . . . 24
2.11.3 Methods . . . 24
2.12 Class WStream . . . 24
2.12.1 Declaration . . . 24
2.12.2 Fields . . . 24
2.12.3 Constructors . . . 24

3 Package fact.processors 25
3.1 Interface FactEvent . . . 26
3.1.1 Declaration . . . 27
3.1.2 Fields . . . 27
3.2 Class CreateHistogram . . . 27
3.2.1 Declaration . . . 27
3.2.2 Constructors . . . 27
3.2.3 Methods . . . 27
3.3 Class CutSlices . . . 28
3.3.1 Declaration . . . 28
3.3.2 Constructors . . . 28
3.3.3 Methods . . . 28
3.4 Class CutValues . . . 29
3.4.1 Declaration . . . 29
3.4.2 Constructors . . . 29
3.4.3 Methods . . . 29
3.5 Class Diff . . . 30
3.5.1 Declaration . . . 30
3.5.2 Constructors . . . 30
3.5.3 Methods . . . 30
3.6 Class DrsCalibration . . . 30
3.6.1 Declaration . . . 31
3.6.2 Constructors . . . 31
3.6.3 Methods . . . 31
3.7 Class ExFit . . . 32
3.7.1 Declaration . . . 32

3.7.2 Constructors 32
3.7.3 Methods 32
3.8 Class ExponentialSmoothing 32
3.8.1 Declaration 32
3.8.2 Constructors 32
3.8.3 Methods 32
3.9 Class FirFilter 33
3.9.1 Declaration 33
3.9.2 Constructors 33
3.9.3 Methods 33
3.10 Class InterpolateBadPixel 33
3.10.1 Declaration 33
3.10.2 Constructors 34
3.10.3 Methods 34
3.11 Class MaxAmplitude 34
3.11.1 Declaration 34
3.11.2 Constructors 34
3.11.3 Methods 34
3.12 Class MaxAmplitudePosition 35
3.12.1 Declaration 35
3.12.2 Constructors 35
3.12.3 Methods 35
3.13 Class MotionDiff 35
3.13.1 Declaration 35
3.13.2 Constructors 35
3.13.3 Methods 36
3.14 Class MovingAverage 36
3.14.1 Declaration 36
3.14.2 Constructors 36
3.14.3 Methods 36
3.15 Class MultiplyValues 36
3.15.1 Declaration 36
3.15.2 Constructors 36
3.15.3 Methods 37
3.16 Class PixelAverage 37
3.16.1 Declaration 37
3.16.2 Constructors 37
3.16.3 Methods 37
3.17 Class Quantiles 38
3.17.1 Declaration 38
3.17.2 Constructors 38
3.17.3 Methods 38
3.18 Class RemoveJumps 38
3.18.1 Declaration 38
3.18.2 Constructors 38
3.18.3 Methods 38

3.19 Class Short2Float . . . 39
 3.19.1 Declaration . . . 39
 3.19.2 Constructors . . . 39
 3.19.3 Methods . . . 39
3.20 Class SimpleFactEventProcessor . . . 39
 3.20.1 Declaration . . . 39
 3.20.2 All known subclasses . . . 39
 3.20.3 Constructors . . . 40
 3.20.4 Methods . . . 40
3.21 Class SimpleFactPixelProcessor . . . 40
 3.21.1 Declaration . . . 40
 3.21.2 All known subclasses . . . 41
 3.21.3 Constructors . . . 41
 3.21.4 Methods . . . 41
3.22 Class SliceNormalization . . . 41
 3.22.1 Declaration . . . 41
 3.22.2 Constructors . . . 41
 3.22.3 Methods . . . 41
3.23 Class SliceQuantileDiscretization . . . 42
 3.23.1 Declaration . . . 42
 3.23.2 Constructors . . . 42
 3.23.3 Methods . . . 42
3.24 Class StdClean . . . 43
 3.24.1 Declaration . . . 43
 3.24.2 Constructors . . . 43
 3.24.3 Methods . . . 43
3.25 Class StdDeviation . . . 44
 3.25.1 Declaration . . . 44
 3.25.2 Constructors . . . 44
 3.25.3 Methods . . . 44
3.26 Class SumKey . . . 45
 3.26.1 Declaration . . . 45
 3.26.2 Constructors . . . 45
 3.26.3 Methods . . . 45
3.27 Class ThresholdEventCounter . . . 45
 3.27.1 Declaration . . . 45
 3.27.2 Constructors . . . 45
 3.27.3 Methods . . . 45
3.28 Class ThresholdPixelCounter . . . 46
 3.28.1 Declaration . . . 46
 3.28.2 Constructors . . . 46
 3.28.3 Methods . . . 46

4 Package fact.processors.parfact **47**
4.1 Class CalcSourcePosition . . . 47
4.1.1 Declaration . . . 47
4.1.2 Constructors . . . 48
4.1.3 Methods . . . 48
4.2 Class CalculatePhotonCharge . . . 48
4.2.1 Declaration . . . 48
4.2.2 Constructors . . . 49
4.2.3 Methods . . . 49
4.3 Class CoreNeighborClean . . . 49
4.3.1 Declaration . . . 49
4.3.2 Constructors . . . 49
4.3.3 Methods . . . 50
4.4 Class HillasParameter . . . 50
4.4.1 Declaration . . . 50
4.4.2 Constructors . . . 51
4.4.3 Methods . . . 51
4.5 Class MonteCarloCalibration . . . 52
4.5.1 Declaration . . . 52
4.5.2 Constructors . . . 52
4.5.3 Methods . . . 52
4.6 Class RemoveSpikesMars . . . 53
4.6.1 Declaration . . . 53
4.6.2 Constructors . . . 53
4.6.3 Methods . . . 53
4.7 Class RisingEdge . . . 53
4.7.1 Declaration . . . 54
4.7.2 Constructors . . . 54
4.7.3 Methods . . . 54
4.8 Class ShowerEllipse . . . 55
4.8.1 Declaration . . . 55
4.8.2 Fields . . . 55
4.8.3 Constructors . . . 55
4.8.4 Methods . . . 56

Class Hierarchy

Classes

- java.lang.Object
    - AbstractLineStream
        - fact.io.ReadMCcsv (in 2.7, page 19)
    - AbstractProcessor
        - fact.processors.SliceNormalization (in 3.22, page 41)
    - AbstractStream
        - fact.io.ByteChunkStream (in 2.2, page 16)
            - fact.io.WStream (in 2.12, page 24)
            - fact.io.WeatherStream (in 2.10, page 22)
        - fact.io.FitsStream (in 2.5, page 18)
        - fact.io.SerializedEventStream (in 2.9, page 21)
    - BarRenderer
        - fact.image.monitors.CustomBarRender (in 1.4, page 7)
    - CsvWriter
        - fact.io.BinaryFactWriter (in 2.1, page 15)
        - fact.io.RootASCIIWriter (in 2.8, page 20)
    - DataVisualizer
        - fact.image.monitors.AverageBarPlotter (in 1.1, page 5)
        - fact.image.monitors.CameraPlotter (in 1.3, page 6)
        - fact.image.monitors.HistogramPlotter (in 1.6, page 8)
        - fact.image.monitors.LiveCameraPlotter (in 1.7, page 9)
        - fact.image.monitors.ScatterPlotter (in 1.9, page 10)
        - fact.image.monitors.ShowerCameraPlotter (in 1.10, page 11)
        - fact.image.monitors.StatusWindow (in 1.11, page 12)
    - StatisticalBarRenderer
        - fact.image.monitors.CustomStatisticalBarRender (in 1.5, page 7)
    - fact.io.CreateAnimatedGif (in 2.3, page 17)
    - fact.io.FitsEventSplitter (in 2.4, page 18)
    - fact.io.FitsStream.FitsHeader (in 2.6, page 19)
    - fact.processors.CutSlices (in 3.3, page 28)
    - fact.processors.CutValues (in 3.4, page 29)
    - fact.processors.Diff (in 3.5, page 30)
    - fact.processors.DrsCalibration (in 3.6, page 30)
    - fact.processors.PixelAverage (in 3.16, page 37)

- fact.processors.Quantiles (in 3.17, page 38)
- fact.processors.RemoveJumps (in 3.18, page 38)
- fact.processors.Short2Float (in 3.19, page 39)
- fact.processors.SimpleFactEventProcessor (in 3.20, page 39)
    - fact.processors.CreateHistogram (in 3.2, page 27)
    - fact.processors.ExFit (in 3.7, page 32)
    - fact.processors.ExponentialSmoothing (in 3.8, page 32)
    - fact.processors.FirFilter (in 3.9, page 33)
    - fact.processors.InterpolateBadPixel (in 3.10, page 33)
    - fact.processors.MaxAmplitudePosition (in 3.12, page 35)
    - fact.processors.MotionDiff (in 3.13, page 35)
    - fact.processors.MovingAverage (in 3.14, page 36)
    - fact.processors.MultiplyValues (in 3.15, page 36)
    - fact.processors.SimpleFactPixelProcessor (in 3.21, page 40)
        - fact.processors.MaxAmplitude (in 3.11, page 34)
    - fact.processors.SumKey (in 3.26, page 45)
    - fact.processors.ThresholdEventCounter (in 3.27, page 45)
    - fact.processors.ThresholdPixelCounter (in 3.28, page 46)
    - fact.processors.parfact.RemoveSpikesMars (in 4.6, page 53)
- fact.processors.SliceQuantileDiscretization (in 3.23, page 42)
- fact.processors.StdClean (in 3.24, page 43)
- fact.processors.StdDeviation (in 3.25, page 44)
- fact.processors.parfact.CalcSourcePosition (in 4.1, page 47)
- fact.processors.parfact.CalculatePhotonCharge (in 4.2, page 48)
- fact.processors.parfact.CoreNeighborClean (in 4.3, page 49)
- fact.processors.parfact.HillasParameter (in 4.4, page 50)
- fact.processors.parfact.MonteCarloCalibration (in 4.5, page 52)
- fact.processors.parfact.RisingEdge (in 4.7, page 53)
- fact.processors.parfact.ShowerEllipse (in 4.8, page 55)
- java.awt.Component
    - java.awt.Container
        - javax.swing.JComponent
            - javax.swing.JPanel
                - fact.image.monitors.BarPlotPanel (in 1.2, page 6)
                - fact.image.monitors.ScatterPlotPanel (in 1.8, page 10)
- java.io.InputStream
    - fact.io.Weird8ByteChunkStream (in 2.11, page 23)

Interfaces

- fact.processors.FactEvent (in 3.1, page 26)

Chapter 1

Package fact.image.monitors

Package Contents *Page*

Classes

AverageBarPlotter 5
This class can plot a bar graph with errorBars by calculating the mean and standarddeviation for a each key and event.

BarPlotPanel 6

CameraPlotter 6

CustomBarRender 7
shamelessly stolen from http://javabeanz.wordpress.com/2007/07/04/creating-barcharts-with-custom-colours-using-jfreechart/

CustomStatisticalBarRender 7
shamelessly stolen from http://javabeanz.wordpress.com/2007/07/04/creating-barcharts-with-custom-colours-using-jfreechart/

HistogramPlotter 8

LiveCameraPlotter 9

ScatterPlotPanel 10

ScatterPlotter 10

ShowerCameraPlotter 11

StatusWindow 12
This opens up a small window containing some information about the currently running stream like number of items per second or the names of the keys in the item.

This package contains a number of classes which are supposed to display some (visual) information about the current state of the stream. The **StatusWindow** for example opens up a small window displaying the number of analyzed items per second and a list of keys

the items contain. The plotters in this package provide a convenient way to quickly analyze the semantics of the ongoing stream. These classes should make it possible to quickly get a meaningful overview of the streamed and processed data. However these classes are *not*meant to provide the same feature set as a real scientific plotting application like GNUPLot.
Most plotters in this package extend the**DataVisualizer**class from the Streams-Framework and rely on the **JFreeChart** library for graphical output.

1.1 Class AverageBarPlotter

This class can plot a bar graph with errorBars by calculating the mean and standarddeviation for a each key and event. If one of the keys refer to an array. The same calculation will be done for every item in the array.

1.1.1 Declaration

public class AverageBarPlotter
extends DataVisualizer

1.1.2 Constructors

- **AverageBarPlotter**
  `public` **AverageBarPlotter**`()`

1.1.3 Methods

- **finish**
  `public void` **finish**`() throws java.lang.Exception`
- **getKeys**
  `public java.lang.String[]` **getKeys**`()`
- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`
- **isDrawErrors**
  `public boolean` **isDrawErrors**`()`
- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`
- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`
- **setDrawErrors**
  `public void` **setDrawErrors**`(boolean` **drawErrors**`)`
- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`

- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`

1.2 Class BarPlotPanel

1.2.1 Declaration

public class BarPlotPanel
extends javax.swing.JPanel

1.2.2 Constructors

- **BarPlotPanel**
  `public` **BarPlotPanel**`(boolean` **drawError**`)`

1.2.3 Methods

- **getDataset**
  `public DefaultStatisticalCategoryDataset` **getDataset**`()`

- **setDataset**
  `public void` **setDataset**`(DefaultStatisticalCategoryDataset` **dataset**`)`

1.3 Class CameraPlotter

1.3.1 Declaration

public class CameraPlotter
extends DataVisualizer

1.3.2 Constructors

- **CameraPlotter**
  `public` **CameraPlotter**`()`

1.3.3 Methods

- **getKeys**
  `public java.lang.String[]` **getKeys**`()`

- **getPanelSize**
  `public java.lang.Double` **getPanelSize**`()`

- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`

- **isDrawErrors**
  `public boolean` **isDrawErrors**`()`

- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`
- **isShowAverage**
  `public boolean` **isShowAverage**`()`
- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`
- **setDrawErrors**
  `public void` **setDrawErrors**`(boolean` **drawErrors**`)`
- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`
- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`
- **setPanelSize**
  `public void` **setPanelSize**`(java.lang.Double` **panelSize**`)`
- **setShowAverage**
  `public void` **setShowAverage**`(boolean` **showAverage**`)`

1.4 Class CustomBarRender

shamelessly stolen from http://javabeanz.wordpress.com/2007/07/04/creating-barcharts-with-custom-colours-using-jfreechart/

1.4.1 Declaration

public class CustomBarRender
extends BarRenderer

1.4.2 Constructors

- **CustomBarRender**
  `public` **CustomBarRender**`()`

1.4.3 Methods

- **getItemPaint**
  `public java.awt.Paint` **getItemPaint**`(int` **row**`, int` **column**`)`

1.5 Class CustomStatisticalBarRender

shamelessly stolen from http://javabeanz.wordpress.com/2007/07/04/creating-barcharts-with-custom-colours-using-jfreechart/

1.5.1 Declaration

public class CustomStatisticalBarRender
extends StatisticalBarRenderer

1.5.2 Constructors

- **CustomStatisticalBarRender**
  `public` **CustomStatisticalBarRender**`()`

1.5.3 Methods

- **getItemPaint**
  `public java.awt.Paint` **getItemPaint**`(int` **row**`, int` **column**`)`

1.6 Class HistogramPlotter

1.6.1 Declaration

public class HistogramPlotter
extends DataVisualizer

1.6.2 Constructors

- **HistogramPlotter**
  `public` **HistogramPlotter**`()`

1.6.3 Methods

- **finish**
  `public void` **finish**`() throws java.lang.Exception`
- **getKey**
  `public java.lang.String` **getKey**`()`
- **getMaxBin**
  `public float` **getMaxBin**`()`
- **getMinBin**
  `public float` **getMinBin**`()`
- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`
- **isDrawErrors**
  `public boolean` **isDrawErrors**`()`
- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`

- **isLogAxis**
  `public boolean` **isLogAxis**`()`

- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`

- **setDrawErrors**
  `public void` **setDrawErrors**`(boolean` **drawErrors**`)`

- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`

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

- **setLogAxis**
  `public void` **setLogAxis**`(boolean` **logAxis**`)`

- **setMaxBin**
  `public void` **setMaxBin**`(float` **maxBin**`)`

- **setMinBin**
  `public void` **setMinBin**`(float` **minBin**`)`

1.7 Class LiveCameraPlotter

1.7.1 Declaration

public class LiveCameraPlotter
extends DataVisualizer

1.7.2 Constructors

- **LiveCameraPlotter**
  `public` **LiveCameraPlotter**`()`

1.7.3 Methods

- **getKey**
  `public java.lang.String` **getKey**`()`

- **getPanelSize**
  `public java.lang.Double` **getPanelSize**`()`

- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`

- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`

- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`
- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`
- **setKey**
  `public void` **setKey**`(java.lang.String` **key**`)`
- **setPanelSize**
  `public void` **setPanelSize**`(java.lang.Double` **panelSize**`)`

1.8 Class ScatterPlotPanel

1.8.1 Declaration

public class ScatterPlotPanel
extends javax.swing.JPanel

1.8.2 Constructors

- **ScatterPlotPanel**
  `public` **ScatterPlotPanel**`(java.lang.String` **key**`)`

1.8.3 Methods

- **getDataset**
  `public XYDataset` **getDataset**`()`
- **setDataset**
  `public void` **setDataset**`(XYDataset` **dataset**`)`

1.9 Class ScatterPlotter

1.9.1 Declaration

public class ScatterPlotter
extends DataVisualizer

1.9.2 Constructors

- **ScatterPlotter**
  `public` **ScatterPlotter**`()`

1.9.3 Methods

- **finish**
  `public void` **finish**`() throws java.lang.Exception`
- **getCompValue**
  `public java.lang.String` **getCompValue**`()`
- **getKeys**
  `public java.lang.String[]` **getKeys**`()`
- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`
- **isDrawErrors**
  `public boolean` **isDrawErrors**`()`
- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`
- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`
- **setCompValue**
  `public void` **setCompValue**`(java.lang.String` **compValue**`)`
- **setDrawErrors**
  `public void` **setDrawErrors**`(boolean` **drawErrors**`)`
- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`
- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`

1.10 Class ShowerCameraPlotter

1.10.1 Declaration

public class ShowerCameraPlotter
extends DataVisualizer

1.10.2 Constructors

- **ShowerCameraPlotter**
  `public` **ShowerCameraPlotter**`()`

1.10.3 Methods

- **getKey**
  `public java.lang.String` **getKey**`()`
- **getPanelSize**
  `public java.lang.Double` **getPanelSize**`()`
- **getShowerKey**
  `public java.lang.String` **getShowerKey**`()`
- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`
- **isKeepOpen**
  `public boolean` **isKeepOpen**`()`
- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`
- **setKeepOpen**
  `public void` **setKeepOpen**`(boolean` **keepOpen**`)`
- **setKey**
  `public void` **setKey**`(java.lang.String` **key**`)`
- **setPanelSize**
  `public void` **setPanelSize**`(java.lang.Double` **panelSize**`)`
- **setShowerKey**
  `public void` **setShowerKey**`(java.lang.String` **showerKey**`)`

1.11 Class StatusWindow

This opens up a small window containing some information about the currently running stream like number of items per second or the names of the keys in the item. (image file not found) (image file not found) (image file not found) (image file not found) (image file not found) (image file not found)(image file not found)(image file not found)(image file not found)(image file not found)

1.11.1 Declaration

public class StatusWindow
extends DataVisualizer

1.11.2 Constructors

- **StatusWindow**
  `public` **StatusWindow**`()`

1.11.3 Methods

- **finish**
  `public void` **finish**`() throws java.lang.Exception`

- **getEvery**
  `public int` **getEvery**`()`

- **init**
  `public void` **init**`(ProcessContext` **ctx**`) throws java.lang.Exception`

- **processMatchingData**
  `public Data` **processMatchingData**`(Data` **data**`)`

- **setEvery**
  `public void` **setEvery**`(int` **every**`)`

Chapter 2

Package fact.io

Package Contents *Page*

Classes

BinaryFactWriter 15

This class writes out FACT events in binary format.

ByteChunkStream 16

This class implements a fast byte-oriented stream of byte chunks.

CreateAnimatedGif 17

FitsEventSplitter 18

FitsStream 18

FitsStream.FitsHeader 19

ReadMCcsv 19

RootASCIIWriter 20

This class writes out FACT events in CSV format.

SerializedEventStream 21

WeatherStream 22

Weird8ByteChunkStream 23

WStream 24

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

2.1 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.

2.1.1 Declaration

public class BinaryFactWriter
extends CsvWriter

2.1.2 Constructors

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

2.1.3 Methods

- **finish**
  `public void` **finish**`() throws java.lang.Exception`
  - **See also**
    - stream.io.CsvWriter#close()
- **getFile**
  `public java.lang.String` **getFile**`()`
  - **Returns** – the file
- **getKey**
  `public java.lang.String` **getKey**`()`
  - **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)
- **setFile**
  `public void` **setFile**`(java.lang.String` **file**`)`
  - **Parameters**

    * `file` – the file to set

- **setKey**
  `public void` **`setKey`**`(java.lang.String` **`key`**`)`
  - **Parameters**
    * `key` – the key to set

2.2 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.

2.2.1 Declaration

public abstract class ByteChunkStream
extends AbstractStream

2.2.2 All known subclasses

WStream (in 2.12, page 24), WeatherStream (in 2.10, page 22)

2.2.3 Fields

- public static final byte **GIF_SIGNATURE**
- public static final byte **JPG_SIGNATURE**
- public static final int **DEFAULT_BUFFER_SIZE**

2.2.4 Constructors

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

2.2.5 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

2.3 Class CreateAnimatedGif

2.3.1 Declaration

public class CreateAnimatedGif
extends java.lang.Object

2.3.2 Constructors

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

2.3.3 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` –

2.4 Class FitsEventSplitter

2.4.1 Declaration

public class FitsEventSplitter
extends java.lang.Object

2.4.2 Constructors

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

2.4.3 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`

2.5 Class FitsStream

2.5.1 Declaration

public class FitsStream
extends AbstractStream

2.5.2 Constructors

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

2.5.3 Methods

- **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 ingnore 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 bytebuffer 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

2.6 Class FitsStream.FitsHeader

2.6.1 Declaration

public class FitsStream.FitsHeader
extends java.lang.Object

2.6.2 Constructors

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

2.6.3 Methods

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

2.7 Class ReadMCcsv

2.7.1 Declaration

public class ReadMCcsv
extends AbstractLineStream

2.7.2 Fields

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

2.7.3 Constructors

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

2.7.4 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**`)`

2.8 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.

2.8.1 Declaration

public class RootASCIIWriter
extends CsvWriter

2.8.2 Constructors

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

2.8.3 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**`)`

2.9 Class SerializedEventStream

2.9.1 Declaration

public class SerializedEventStream
extends AbstractStream

2.9.2 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`

2.9.3 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**`)`

2.10 Class WeatherStream

2.10.1 Declaration

public class WeatherStream
extends fact.io.ByteChunkStream (in 2.2, page 16)

2.10.2 Fields

- public static final int **MAX_MESSAGE_LENGTH**

2.10.3 Constructors

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

2.10.4 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

2.11 Class Weird8ByteChunkStream

2.11.1 Declaration

public class Weird8ByteChunkStream
extends java.io.InputStream

2.11.2 Constructors

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

2.11.3 Methods

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

2.12 Class WStream

2.12.1 Declaration

public class WStream
extends fact.io.ByteChunkStream (in 2.2, page 16)

2.12.2 Fields

- public static final byte **SIG**

2.12.3 Constructors

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

Chapter 3

Package fact.processors

Package Contents *Page*

Interfaces

FactEvent 26
This is an implementation of the Data item interface that provides easy access to all pixels of an event by their SoftID.

Classes

CreateHistogram 27

CutSlices 28

CutValues 29
This operator simply cuts all values below and above the min and maxValue.

Diff 30
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.

DrsCalibration 30
This processor handles the DRS calibration.

ExFit 32
This operator does a very simple fit of an exp-function to the data in each pixel.

ExponentialSmoothing 32
Calculates first Order exponential Smoothing Let y be the original Series and s be the smoothed one.

FirFilter 33
This class implements a simple Fir-Filter.

InterpolateBadPixel 33
This Processor interpolates all values for a broken Pixel by the average values of its neighboring Pixels.

MaxAmplitude 34
This processor simply calculates the maximum value for all time slices in each Pixel.

MaxAmplitudePosition 35

This processor simply calculates the position of the maximum value for all time slices in each Pixel.
MotionDiff 35
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 36

MultiplyValues 36
This operator simply multiplies all values by the given factor.
PixelAverage 37
This operator calculates the average of all the slices in each Pixel and stores the result as a double array.
Quantiles 38

RemoveJumps 38

Short2Float 39

SimpleFactEventProcessor 39

SimpleFactPixelProcessor 40
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.
SliceNormalization 41

SliceQuantileDiscretization 42

StdClean 43
This processor identifies showerPixel in the image array by comparing the data in a pixel with some other time series.
StdDeviation 44
This Processor calculates the Standarddeviation of the slices in each pixel.
SumKey 45
This operator simply sums up all values with the given key.
ThresholdEventCounter 45

ThresholdPixelCounter 46
This processor counts the number of Pixels in each event that have a value >maxValue.

3.1 Interface FactEvent

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

3.1.1 Declaration

public interface FactEvent

3.1.2 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**

3.2 Class CreateHistogram

3.2.1 Declaration

public class CreateHistogram
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.2.2 Constructors

- **CreateHistogram**
  `public` **CreateHistogram**`()`

3.2.3 Methods

- **getMax**
  `public float` **getMax**`()`
- **getMin**
  `public float` **getMin**`()`
- **getNumberOfBins**
  `public int` **getNumberOfBins**`()`
- **init**
  `public void` **init**`(ProcessContext` **context**`)`
- **processSeries**
  `public int[]` **processSeries**`(float[]` **data**`)`
- **setMax**
  `public void` **setMax**`(float` **maxBin**`)`
- **setMin**
  `public void` **setMin**`(float` **minbin**`)`

- **setNumberOfBins**
  `public void` **setNumberOfBins**`(int` **numberOfBins**`)`

3.3 Class CutSlices

3.3.1 Declaration

public class CutSlices
extends java.lang.Object

3.3.2 Constructors

- **CutSlices**
  `public` **CutSlices**`()`

3.3.3 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
- **main**
  `public static void` **main**`(java.lang.String[]` **args**`)`
- **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

3.4 Class CutValues

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

3.4.1 Declaration

public class CutValues
extends java.lang.Object

3.4.2 Constructors

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

3.4.3 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)`

3.5 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.

3.5.1 Declaration

public class Diff
extends java.lang.Object

3.5.2 Constructors

- **Diff**
  `public` **Diff**`()`

3.5.3 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**`)`

3.6 Class DrsCalibration

This processor handles the DRS calibration. It requires a DRS data souce 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.

3.6.1 Declaration

public class DrsCalibration
extends java.lang.Object

3.6.2 Constructors

- **DrsCalibration**
  `public` **DrsCalibration**`()`

3.6.3 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**`)`

3.7 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 postion 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.

3.7.1 Declaration

public class ExFit
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.7.2 Constructors

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

3.7.3 Methods

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

3.8 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

3.8.1 Declaration

public class ExponentialSmoothing
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.8.2 Constructors

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

3.8.3 Methods

- **getAlpha**
  `public float` **`getAlpha`**`()`

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

- **setAlpha**
  `public void` **setAlpha**`(float` **alpha**`)`

3.9 Class FirFilter

This class implements a simple Fir-Filter. See 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.

3.9.1 Declaration

public class FirFilter
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.9.2 Constructors

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

3.9.3 Methods

- **getCoefficents**
  `public double[]` **getCoefficents**`()`

- **getTemplate**
  `public java.lang.String` **getTemplate**`()`

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

- **setCoefficents**
  `public void` **setCoefficents**`(double[]` **coefficents**`)`

- **setTemplate**
  `public void` **setTemplate**`(java.lang.String` **templateString**`)`

3.10 Class InterpolateBadPixel

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

3.10.1 Declaration

public class InterpolateBadPixel
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.10.2 Constructors

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

3.10.3 Methods

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

3.11 Class MaxAmplitude

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

3.11.1 Declaration

public class MaxAmplitude
extends fact.processors.SimpleFactPixelProcessor (in 3.21, page 40)

3.11.2 Constructors

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

3.11.3 Methods

- **getMaxValue**
  `public float` **getMaxValue**`()`
- **getMinValue**
  `public float` **getMinValue**`()`
- **processPixel**
  `public abstract float` **processPixel**`(float[]` **pixelData**`)`
- **setMaxValue**
  `public void` **setMaxValue**`(float` **maxValue**`)`
- **setMinValue**
  `public void` **setMinValue**`(float` **minValue**`)`

3.12 Class MaxAmplitudePosition

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

3.12.1 Declaration

public class MaxAmplitudePosition
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.12.2 Constructors

- **MaxAmplitudePosition**
  `public` **MaxAmplitudePosition**`()`

3.12.3 Methods

- **getMaxValue**
  `public float` **getMaxValue**`()`
- **getMinValue**
  `public float` **getMinValue**`()`
- **processSeries**
  `public int[]` **processSeries**`(float[]` **data**`)`
- **setMaxValue**
  `public void` **setMaxValue**`(float` **maxValue**`)`
- **setMinValue**
  `public void` **setMinValue**`(float` **minValue**`)`

3.13 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.

3.13.1 Declaration

public class MotionDiff
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.13.2 Constructors

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

3.13.3 Methods

- **getOffset**
  `public int` **getOffset**`()`

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

- **setOffset**
  `public void` **setOffset**`(int` **offset**`)`

3.14 Class MovingAverage

3.14.1 Declaration

public class MovingAverage
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.14.2 Constructors

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

3.14.3 Methods

- **getLength**
  `public int` **getLength**`()`

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

- **setLength**
  `public void` **setLength**`(int` **length**`)`

3.15 Class MultiplyValues

This operator simply multiplies all values by the given factor.

3.15.1 Declaration

public class MultiplyValues
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.15.2 Constructors

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

3.15.3 Methods

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

3.16 Class PixelAverage

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

3.16.1 Declaration

public class PixelAverage
extends java.lang.Object

3.16.2 Constructors

- **PixelAverage**
  `public` **PixelAverage**`()`
- **PixelAverage**
  `public` **PixelAverage**`(java.lang.String` **key**`)`

3.16.3 Methods

- **getKey**
  `public java.lang.String` **getKey**`()`
- **getOutput**
  `public java.lang.String` **getOutput**`()`
- **process**
  `public Data` **process**`(Data` **input**`)`
  - **See also**
    - stream.DataProcessor#process(stream.Data)
- **processEvent**
  `public double[]` **processEvent**`(Data` **input**`, java.lang.String` **key**`)`
- **processSeries**
  `public double[]` **processSeries**`(float[]` **series**`)`

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

- **setOutput**
  `public void` **setOutput**`(java.lang.String` **output**`)`

3.17 Class Quantiles

3.17.1 Declaration

public class Quantiles
extends java.lang.Object

3.17.2 Constructors

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

3.17.3 Methods

- **getQuantile**
  `public float` **getQuantile**`(double` **phi**`)`

- **print**
  `public void` **print**`(java.lang.Double[]` **phis**`)`

3.18 Class RemoveJumps

3.18.1 Declaration

public class RemoveJumps
extends java.lang.Object

3.18.2 Constructors

- **RemoveJumps**
  `public` **RemoveJumps**`()`

3.18.3 Methods

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

- **getKey**
  `public java.lang.String` **getKey**`()`

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

- **process**
  `public Data process(Data input)`
  - **Description**
    Each event contains the StartCellData array which contains the current starcell 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)`
- **setOutputKey**
  `public void setOutputKey(java.lang.String outputKey)`

3.19 Class Short2Float

3.19.1 Declaration

public class Short2Float
extends java.lang.Object

3.19.2 Constructors

- **Short2Float**
  `public Short2Float()`

3.19.3 Methods

- **process**
  `public Data process(Data item)`

3.20 Class SimpleFactEventProcessor

3.20.1 Declaration

public abstract class SimpleFactEventProcessor
extends java.lang.Object

3.20.2 All known subclasses

MovingAverage (in 3.14, page 36), InterpolateBadPixel (in 3.10, page 33), FirFilter (in 3.9, page 33), MotionDiff (in 3.13, page 35), MaxAmplitude (in 3.11, page 34), ThresholdEventCounter (in 3.27, page 45), ExponentialSmoothing (in 3.8, page 32), MultiplyValues (in 3.15, page 36), ThresholdPixelCounter (in 3.28, page 46), SumKey (in 3.26, page 45), CreateHistogram (in 3.2, page 27), SimpleFactPixelProcessor (in 3.21, page 40), ExFit (in 3.7, page 32), MaxAmplitudePosition (in 3.12, page 35), RemoveSpikesMars (in 4.6, page 53)

3.20.3 Constructors

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

3.20.4 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**`)`

3.21 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.

3.21.1 Declaration

public abstract class SimpleFactPixelProcessor
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.21.2 All known subclasses

MaxAmplitude (in 3.11, page 34)

3.21.3 Constructors

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

3.21.4 Methods

- **processPixel**
  `public abstract float` **processPixel**`(float[]` **pixelData**`)`

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

3.22 Class SliceNormalization

3.22.1 Declaration

public class SliceNormalization
extends AbstractProcessor

3.22.2 Constructors

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

- **SliceNormalization**
  `public` **SliceNormalization**`(java.lang.String[]` **keys**`)`

3.22.3 Methods

- **getKeys**
  `public java.lang.String[]` **getKeys**`()`
  - **Returns** – the keys

- **isOverwrite**
  `public boolean` **isOverwrite**`()`

- **process**
  `public Data` **process**`(Data` **input**`)`
  - **See also**
    - stream.Processor#process(stream.Data)

- **processEvent**
  `public float[]` **processEvent**`(Data` **input**`, java.lang.String` **key**`)`

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

- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`

  - **Parameters**
    - `keys` – the keys to set

- **setOverwrite**
  `public void` **setOverwrite**`(boolean` **overwrite**`)`

3.23 Class SliceQuantileDiscretization

3.23.1 Declaration

public class SliceQuantileDiscretization
extends java.lang.Object

3.23.2 Constructors

- **SliceQuantileDiscretization**
  `public` **SliceQuantileDiscretization**`()`

3.23.3 Methods

- **getBins**
  `public java.lang.Integer` **getBins**`()`

  - **Returns** – the bins

- **getKey**
  `public java.lang.String` **getKey**`()`

  - **Returns** – the key

- **process**
  `public Data` **process**`(Data` **input**`)`

  - **See also**
    - stream.Processor#process(stream.Data)

- **setBins**
  `public void` **setBins**`(java.lang.Integer` **bins**`)`

  - **Parameters**
    - `bins` – the bins to set

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

- **Parameters**
  - `key` – the key to set

3.24 Class StdClean

This processor identifies showerPixel in the image array by comparing the data in a pixel with some other time series. To compare two series the squared absolute difference between the two series is calculated. If the difference is less than the given showerthreshold the pixel will be added to the showerpixel list. The operator also calculates the number unconnected subsets in the showerPixel set.

3.24.1 Declaration

public class StdClean
extends java.lang.Object

3.24.2 Constructors

- **StdClean**
  `public` **StdClean**`()`

3.24.3 Methods

- **getInKey**
  `public java.lang.String` **getInKey**`()`

- **getKey**
  `public java.lang.String` **getKey**`()`

- **getOutput**
  `public java.lang.String` **getOutput**`()`

- **getShowerThreshold**
  `public double` **getShowerThreshold**`()`

- **process**
  `public Data` **process**`(Data` **input**`)`
  - **See also**
    - stream.DataProcessor#process(stream.Data)

- **processEvent**
  `public java.util.ArrayList` **processEvent**`(Data` **input**`, java.lang.String` **key**`)`

- **processSeries**
  `public java.util.ArrayList` **processSeries**`(float[]` **data**`)`

- **processSeries**
  `public java.util.ArrayList` **processSeries**`(float[]` **series**`, double` **t**`)`
- **setInKey**
  `public void` **setInKey**`(java.lang.String` **inKey**`)`
- **setKey**
  `public void` **setKey**`(java.lang.String` **key**`)`
- **setOutput**
  `public void` **setOutput**`(java.lang.String` **output**`)`
- **setShowerThreshold**
  `public void` **setShowerThreshold**`(double` **showerThreshold**`)`

3.25 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.

3.25.1 Declaration

public class StdDeviation
extends java.lang.Object

3.25.2 Constructors

- **StdDeviation**
  `public` **StdDeviation**`()`
- **StdDeviation**
  `public` **StdDeviation**`(java.lang.String[]` **keys**`)`

3.25.3 Methods

- **getKeys**
  `public java.lang.String[]` **getKeys**`()`
  - **Returns** – the keys
- **process**
  `public Data` **process**`(Data` **input**`)`
  - **See also**
    - stream.DataProcessor#process(stream.Data)
- **processEvent**
  `public double[]` **processEvent**`(Data` **input**`, java.lang.String` **key**`)`

- **processSeries**
  `public double[]` **processSeries**`(float[]` **data**`)`
- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`
  - **Parameters**
    - `keys` – the keys to set

3.26 Class SumKey

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

3.26.1 Declaration

public class SumKey
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.26.2 Constructors

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

3.26.3 Methods

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

3.27 Class ThresholdEventCounter

3.27.1 Declaration

public class ThresholdEventCounter
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.27.2 Constructors

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

3.27.3 Methods

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

3.28 Class ThresholdPixelCounter

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

3.28.1 Declaration

public class ThresholdPixelCounter
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

3.28.2 Constructors

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

3.28.3 Methods

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

Chapter 4

Package fact.processors.parfact

Package Contents *Page*

Classes

CalcSourcePosition 47

This is supposed to calculate the position of the source in the camera.

CalculatePhotonCharge 48

This processor Calculates PhotonCharge by doing the following: 1.

CoreNeighborClean 49

CoreNeighborClean.

HillasParameter 50

Calculates the Hillas Parameter from the Ellipse.

MonteCarloCalibration 52

This operator simply multiplies all values by the given factor.

RemoveSpikesMars 53

Supposedly removes all spikes in the data.

RisingEdge 53

TODO: this needs to be redone.

ShowerEllipse 55

4.1 Class CalcSourcePosition

This is supposed to calculate the position of the source in the camera. Original C++ Code by F.Temme

4.1.1 Declaration

public class CalcSourcePosition
extends java.lang.Object

4.1.2 Constructors

- **CalcSourcePosition**
  `public` **CalcSourcePosition**`()`

4.1.3 Methods

- **getOutputKey**
  `public java.lang.String` **getOutputKey**`()`
- **getSourceDeclination**
  `public double` **getSourceDeclination**`()`
- **getSourceRightAscension**
  `public double` **getSourceRightAscension**`()`
- **process**
  `public Data` **process**`(Data` **data**`)`
  - **See also**
    - fact.data.FactProcessor#process(stream.Data)
- **setOutputKey**
  `public void` **setOutputKey**`(java.lang.String` **outputKey**`)`
- **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**`)`

4.2 Class CalculatePhotonCharge

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 maxAmplitude. 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.

4.2.1 Declaration

public class CalculatePhotonCharge
extends java.lang.Object

4.2.2 Constructors

- **CalculatePhotonCharge**
  `public` **CalculatePhotonCharge**`()`

4.2.3 Methods

- **getIntegralGain**
  `public float` **getIntegralGain**`()`

- **getKey**
  `public java.lang.String` **getKey**`()`

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

- **process**
  `public Data` **process**`(Data` **input**`)`

- **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**`)`

4.3 Class CoreNeighborClean

CoreNeighborClean. Identifies showerPixel in the image array. Cleaning in three Steps: 1) Identify all Core Pixel (Photoncharge higher than corePixelThreshold) 2) Remove all Single Core Pixel 3) Add all Neighbor Pixel, whose Photoncharge is higher than neighborPixelThreshold

4.3.1 Declaration

public class CoreNeighborClean
extends java.lang.Object

4.3.2 Constructors

- **CoreNeighborClean**
  `public` **CoreNeighborClean**`()`

4.3.3 Methods

- **getCorePixelThreshold**
  `public float` **getCorePixelThreshold**`()`
- **getKey**
  `public java.lang.String` **getKey**`()`
- **getMinSize**
  `public int` **getMinSize**`()`
- **getNeighborPixelThreshold**
  `public float` **getNeighborPixelThreshold**`()`
- **getOutputKey**
  `public java.lang.String` **getOutputKey**`()`
- **process**
  `public Data` **process**`(Data` **input**`)`
- **setCorePixelThreshold**
  `public void` **setCorePixelThreshold**`(float` **corePixelThreshold**`)`
- **setKey**
  `public void` **setKey**`(java.lang.String` **key**`)`
- **setMinSize**
  `public void` **setMinSize**`(int` **minSize**`)`
- **setNeighborPixelThreshold**
  `public void` **setNeighborPixelThreshold**`(float` **neighborPixelThreshold**`)`
- **setOutputKey**
  `public void` **setOutputKey**`(java.lang.String` **output**`)`

4.4 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(); }

4.4.1 Declaration

public class HillasParameter
extends java.lang.Object

4.4.2 Constructors

- **HillasParameter**
  `public` **HillasParameter**`()`

4.4.3 Methods

- **calculateSize**
  `public float` **calculateSize**`(int[]` **showerPixelArray**`, float[]` **photon-Charges**`)`
  - **Description**
    calculates the sum of all photoncharges for all showerPixel
- **getCorePixelThreshold**
  `public float` **getCorePixelThreshold**`()`
- **getKey**
  `public java.lang.String` **getKey**`()`
- **getMinSize**
  `public int` **getMinSize**`()`
- **getNeighborPixelThreshold**
  `public float` **getNeighborPixelThreshold**`()`
- **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**`)`

- **setCorePixelThreshold**
  `public void` **setCorePixelThreshold**`(float` **corePixelThreshold**`)`
- **setKey**
  `public void` **setKey**`(java.lang.String` **key**`)`
- **setMinSize**
  `public void` **setMinSize**`(int` **minSize**`)`
- **setNeighborPixelThreshold**
  `public void` **setNeighborPixelThreshold**`(float` **neighborPixelThreshold**`)`
- **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**`)`

4.5 Class MonteCarloCalibration

This operator simply multiplies all values by the given factor.

4.5.1 Declaration

public class MonteCarloCalibration
extends java.lang.Object

4.5.2 Constructors

- **MonteCarloCalibration**
  `public` **MonteCarloCalibration**`()`

4.5.3 Methods

- **getFactor**
  `public float` **getFactor**`()`
- **getKeys**
  `public java.lang.String[]` **getKeys**`()`

- **getOffset**
  `public float` **getOffset**`()`
- **isOverWrite**
  `public boolean` **isOverWrite**`()`
- **process**
  `public Data` **process**`(Data` **event**`)`
  - **See also**
    - stream.DataProcessor#process(stream.Data)
- **setFactor**
  `public void` **setFactor**`(float` **threshold**`)`
- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`
- **setOffset**
  `public void` **setOffset**`(float` **offset**`)`
- **setOverWrite**
  `public void` **setOverWrite**`(boolean` **overWrite**`)`

4.6 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

4.6.1 Declaration

public class RemoveSpikesMars
extends fact.processors.SimpleFactEventProcessor (in 3.20, page 39)

4.6.2 Constructors

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

4.6.3 Methods

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

4.7 Class RisingEdge

TODO: this needs to be redone. the orignal code is a joke. talk to fabian about outofbounds errors. Also CFD is better here

4.7.1 Declaration

public class RisingEdge
extends java.lang.Object

4.7.2 Constructors

- **RisingEdge**
  `public` **RisingEdge**`()`

- **RisingEdge**
  `public` **RisingEdge**`(java.lang.String[]` **keys**`)`

4.7.3 Methods

- **getKeys**
  `public java.lang.String[]` **getKeys**`()`
  - **Returns** – the keys

- **getSearchWindowLeft**
  `public int` **getSearchWindowLeft**`()`

- **getSearchWindowRight**
  `public int` **getSearchWindowRight**`()`

- **isOverwrite**
  `public boolean` **isOverwrite**`()`

- **process**
  `public Data` **process**`(Data` **input**`)`
  - **See also**
    - stream.DataProcessor#process(stream.Data)

- **processEvent**
  `public int[]` **processEvent**`(Data` **input**`, java.lang.String` **key**`)`

- **processSeries**
  `public int[]` **processSeries**`(float[]` **input**`)`

- **setKeys**
  `public void` **setKeys**`(java.lang.String[]` **keys**`)`
  - **Parameters**
    - `keys` – the keys to set

- **setOverwrite**
  `public void` **setOverwrite**`(boolean` **overwrite**`)`

- **setSearchWindowLeft**
  `public void` **setSearchWindowLeft**`(int` **searchWindowLeft**`)`

- **setSearchWindowRight**
  `public void` **setSearchWindowRight**`(int` **searchWindowRight**`)`

4.8 Class ShowerEllipse

4.8.1 Declaration

public class ShowerEllipse
extends java.lang.Object

4.8.2 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**

4.8.3 Constructors

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

4.8.4 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**`()`