com.linkedin.databus.core

Class DbusEvent

```
java.lang.Object
  └ com.linkedin.databus.core.DbusEvent
```

All Implemented Interfaces:

DataChangeEvent, java.lang.Cloneable

```
public class DbusEvent
extends java.lang.Object
implements DataChangeEvent, java.lang.Cloneable
```

This class represents a Databus event stored in a `ByteBuffer`.

Binary Serialization Format

The table below summarizes the serialization format of Databus events as stored into memory and sent over the wire.

Field	Offset	Type	Size	Description
Header (61 bytes for events with long keys, 57 bytes + key length)				
MagicByte	0	byte	1	A special value denoting the beginning of a message
HeaderCrc	1	int	4	CRC computed over the header
Length	5	int	4	Total length of the event in bytes (fixed-length header + variable-length payload)
Attributes	9	short	2	Event attributes bitmap (see below)
Sequence	11	long	8	Sequence number for the event window in which this event was generated

PhysicalPartitionId	19	short	2	physical partition-id -> represents a sequence generator
LogicalPartitionId	21	short	2	logical partition-id -> represents a logical partition of the physical stream
NanoTimestamp	23	long	8	Time (in nanoseconds) at which the event was generated
srcId	31	short	2	Databus source id for the event
SchemaId*	33	byte[]	16	hash for the schema used to generate the event
ValueCrc	49	int	4	a CRC computed over the variable-length payload of the event
Key	53	long	8	key value for events with long keys
KeySize	53	int	4	key length for byte[]-valued keys
Variable-size payload				
Key	57	byte[]	4 or KeySize	32 LSB from key value for long-typed keys or key value for byte[] keys
Value	61 or 57 + KeySize	byte[]	Length - Offset(Value)	Serialized event payload

JSON serialization format

The table below summarizes the JSON serialization format.

Attribute	Type	Description	Optional
opcode	String	UPSERT or DELETE	Yes
keyBytes	String	Base-64 encoding of the key byte sequence for string keys	One of the two needs to be present
key	Long	key value for numeric keys	
sequence	Long	event sequence number	No

logicalPartitionId	Short	logical partition id	No
physicalPartitionId	Short	physical partition id	No
timestampInNanos	Long	creation timestamp in nanoseconds since the Unix Epoch	No
srcId	Short	source id	No
schemaId	String	Base-64 encoding of the event serialization schema hash id	No
valueEnc	String	value encoding format: JSON or JSON_PLAIN	No
endOfPeriod	Boolean	true iff the event denotes end of event window	Yes; default is false
value	String	Literal value string for JSON_PLAIN encoding or Base-64 encoding of the value byte sequence for JSON encoding	Yes; default is false

Event attributes

The table below summarizes the Databus event attribute bits

Attribute	Bit N	Description
OpCode0	0	Bit 0 of event opcode
OpCode1	1	bit 1 of event opcode
Trace	2	The event is a trace event
ByteKey	3	The event has a byte[] key
EoP	4	The event is the last event in a event window
ExtReplEvent	8	The event was generated through external replication (e.g. originating in a different data center)

Event opcodes

Currently, Databus supports two choices of event opcodes

- 1 - UPSERT
- 2 - DELETE

Databus source ids

The possible values for Databus source ids are partitioned into several ranges. In general, all positive source ids are used to uniquely represent a Databus source. The source ids are used in Databus data messages. All non-positive values are reserved for Databus system use. These source ids are used in Databus control messages.

- [1, java.lang.Short.MAX_VALUE] - data source ids
- [java.lang.Short.MIN_VALUE, 0] - system source ids
  - [PRIVATE_RANGE_MAX_SRCID + 1, 0] - global system source ids. These control messages will be transmitted to other Databus components over the network
    - -3 - Checkpoint event
  - [java.lang.Short.MIN_VALUE, PRIVATE_RANGE_MAX_SRCID] - private system source ids. These messages are used for internal communication inside a Databus component and are not transmitted to other Databus components.

Nested Class Summary

static class	**DbusEvent.EventScanStatus**
static class	**DbusEvent.HeaderScanStatus**

Field Summary

static short	**BOOTSTRAPTOOOLD_ERROR_SRCID**
static java.nio.ByteOrder	**byteOrder**
static short	**CHECKPOINT_SRCID**
static java.lang.Byte	**CurrentMagicValue** Serialization Format is : MagicByte (1 byte) HeaderCrc (4 bytes) // Crc to protect the header from being corrupted Length (4 bytes) Attributes (2 byte) // Key-type, Trace marker, Event-opcode Sequence (8 bytes) // Sequence number for the event window in which this event was generated Physical PartitionId (2 byte) // Short physical partition-id -> represents a sequence generator Logical PartitionId (2 byte) // Short logical

	partition-id -> represents a logical partition of the physical stream NanoTimestamp (8 bytes) // Time (in nanoseconds) at which the event was generated SrcId (short) // SourceId for the event SchemaId (short) // 16-byte hash for the schema used to generate the event ValueCrc (4 bytes) // Crc to protect the value from being corrupted Key (8 bytes long key) or KeySize (4 bytes for byte[] key) Key Bytes (k bytes for byte[] key) Value (N bytes) // Serialized Event
static short	**DISPATCHER_RETRIES_EXPIRED**
static byte[]	**emptymd5**
static short	**EOPMarkerSrcId** Denotes the end of the range of srcid values reserved for private use: (Short.MIN_VALUE, PRIVATE_RANGE_MAX_SRCID]
static byte[]	**EOPMarkerValue**
static int	**LengthLength**
static int	**LengthOffset**
static org.apache.log4j.Logger	**LOG**
static int	**LogicalPartitionIdLength**
static int	**LogicalPartitionIdOffset**
static int	**MagicLength**
static java.lang.String	**MODULE**
static int	**PhysicalPartitionIdLength**
static int	**PhysicalPartitionIdOffset**
static short	**PRIVATE_RANGE_MAX_ERROR_SRCID**

static short	**PRIVATE_RANGE_MAX_SRCID**
static short	**PRIVATE_RANGE_MIN_ERROR_SRCID**
static short	**PULLER_RETRIES_EXPIRED**
static short	**SCN_REGRESS**
static int	**SequenceLength**
static int	**SequenceOffset**
static int	**SrcIdLength**
static int	**SrcIdOffset**
static int	**TimestampLength**
static int	**TimestampOffset**

Constructor Summary

DbusEvent()
DbusEvent(java.nio.ByteBuffer buf, int position)

Method Summary

static int	**appendToEventBuffer**(java.io.BufferedReader jsonStream, DbusEventBufferAppendable eventBuffer, DbusEventsStatisticsCollector statsCollector, boolean startWindow)
static int	**appendToEventBuffer**(java.lang.String jsonString, DbusEventBufferAppendable eventBuffer, DbusEventsStatisticsCollector statsCollector, boolean startWindow) Appends a single event to the buffer.
void	**applyCrc**()

DbusEvent	**clone**(DbusEvent reuse) Creates a copy of the current event.
static DbusEvent	**createCheckpointEvent**(Checkpoint checkpoint) Utility method to create a DbusEvent with an embedded checkpoint in it.
DbusEvent	**createCopy**()
static DbusEvent	**createErrorEvent**(DbusErrorEvent errorEvent)
static DbusEvent	**createSCNRegressEvent**(SCNRegressMessage message) Utility method to create a DbusEvent with a SCNRegress message in it.
boolean	**equals**(java.lang.Object obj)
static Checkpoint	**getCheckpointFromEvent**(DbusEvent event) Utility method to extract a checkpoint from a DbusEvent Note: Ensure that this is a Checkpoint event before calling this method.
static DbusErrorEvent	**getErrorEventFromDbusEvent**(DbusEvent event)
DbusOpcode	**getOpcode**() Returns the opcode of the data event; null for non-data events
java.nio.ByteBuffer	**getRawBytes**()
static SCNRegressMessage	**getSCNRegressFromEvent**(DbusEvent event) Utility method to extract a SCN regress message from a DbusEvent Note: Ensure that this is a SCNRegressMessage event before calling this method.
int	**hashCode**()
long	**headerCrc**()
int	**headerLength**()
boolean	**inited**()
boolean	**isCheckpointMessage**() Checks if the event is a checkpoint message

boolean	**isControlMessage**() Checks if the event is a control message
static boolean	**isControlSrcId**(short srcId)
boolean	**isEndOfPeriodMarker**() Checks if the event denotes the end of an event window
boolean	**isErrorEvent**()
boolean	**isExtReplicatedEvent**()
protected boolean	**isHeaderPartial**(boolean logErrors)
protected boolean	**isHeaderValid**(boolean logErrors) Checks if the event header - containing length is vsalid
boolean	**isKeyNumber**() Returns true iff the key of the event is a numeric (long)
boolean	**isKeyString**() Returns true iff the key of th event is a string (byte sequence)
boolean	**isPartial**()
protected boolean	**isPartial**(boolean logErrors)
boolean	**isPrivateControlMessage**() Checks if the event is a private control message
boolean	**isSCNRegressMessage**() Checks if the event is a SCNRegressMessage
boolean	**isTraceEnabled**() Returns true if tracing has been enabled for the event
boolean	**isValid**() Returns true iff the event points to a valid Databus event
protected boolean	**isValid**(boolean logErrors)

long	**key**() Returns key value for events with numeric keys; undefined for events with string keys.
byte[]	**keyBytes**() Returns the key value for events with string keys; undefined for events with numeric keys.
int	**keyLength**() Returns the length of the event key
static int	**length**(DbusEventKey key, byte[] value)
short	**logicalPartitionId**() Returns the logical partition id for the event
java.lang.Byte	**magic**()
int	**payloadLength**()
short	**physicalPartitionId**() Returns the physical partition id for the event
void	**reset**(java.nio.ByteBuffer buf, int position)
DbusEvent.EventScanStatus	**scanEvent**()
protected DbusEvent.EventScanStatus	**scanEvent**(boolean logErrors)
DbusEvent.HeaderScanStatus	**scanHeader**()
protected DbusEvent.HeaderScanStatus	**scanHeader**(boolean logErrors)
byte[]	**schemaId**() Returns a byte array with the hash id of the event serialization schema.
void	**schemaId**(byte[] md5) Stores the hash id of the event serialization schema in an existing byte array.
short	**schemaVersion**()
long	**sequence**() Returns the sequence number of the event

static int	**serializeEndOfPeriodMarker**(java.nio.ByteBuffer serializationBuffer, DbusEventInfo eventInfo) Serializes an End-Of-Period Marker onto the ByteBuffer passed in.
static int	**serializeEvent**(DbusEventKey key, java.nio.ByteBuffer serializationBuffer, DbusEventInfo dbusEventInfo)
static int	**serializeEvent**(DbusEventKey key, short pPartitionId, short lPartitionId, long timeStampInNanos, short srcId, byte[] schemaId, byte[] value, boolean enableTracing, java.nio.ByteBuffer serializationBuffer) non-threadsafe : serializationBuffer needs to be protected if multiple threads are writing to it concurrently
static int	**serializeFullEvent**(DbusEventKey key, java.nio.ByteBuffer serializationBuffer, DbusEventInfo eventInfo)
static void	**setExtReplicationFlag**(byte[] attribute)
void	**setHeaderCrc**(long crc)
void	**setSchemaId**(byte[] schemaId) put a byte[] schemaId into the buffer .
void	**setSchemaVersion**(short schemaVersion)
void	**setSequence**(long sequence)
void	**setSize**(int sz) Setter for size
void	**setSrcId**(short srcId)
static void	**setTraceFlag**(byte[] attribute)
void	**setValue**(byte[] bytes)
void	**setValueCrc**(long crc)
int	**size**() Returns the total size of the event binary serialization

short	**srcId**() Returns the Databus source id of the event
long	**timestampInNanos**() Returns the creation timestamp of the event in nanoseconds from Unix epoch
java.lang.String	**toString**()
void	**unsetInited**()
java.nio.ByteBuffer	**value**() Obtains the data payload of the event
long	**valueCrc**()
int	**valueLength**() Returns the length of the data event value (data payload)
int	**writeTo**(java.nio.channels.WritableByteChannel writeChannel, Encoding encoding) Serializes the event to a channel using the specified encoding

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

MODULE

public static final java.lang.String **MODULE**

LOG

public static final org.apache.log4j.Logger **LOG**

EOPMarkerSrcId

public static final short **EOPMarkerSrcId**

Denotes the end of the range of srcid values reserved for private use: (Short.MIN_VALUE, PRIVATE_RANGE_MAX_SRCID]

See Also:
Constant Field Values

CHECKPOINT_SRCID

`public static final short CHECKPOINT_SRCID`

See Also:
Constant Field Values

PRIVATE_RANGE_MAX_ERROR_SRCID

`public static final short PRIVATE_RANGE_MAX_ERROR_SRCID`

See Also:
Constant Field Values

BOOTSTRAPTOOOLD_ERROR_SRCID

`public static final short BOOTSTRAPTOOOLD_ERROR_SRCID`

See Also:
Constant Field Values

PULLER_RETRIES_EXPIRED

`public static final short PULLER_RETRIES_EXPIRED`

See Also:
Constant Field Values

DISPATCHER_RETRIES_EXPIRED

`public static final short DISPATCHER_RETRIES_EXPIRED`

See Also:
Constant Field Values

PRIVATE_RANGE_MIN_ERROR_SRCID

public static final short **PRIVATE_RANGE_MIN_ERROR_SRCID**

See Also:
Constant Field Values

PRIVATE_RANGE_MAX_SRCID

public static final short **PRIVATE_RANGE_MAX_SRCID**

See Also:
Constant Field Values

SCN_REGRESS

public static final short **SCN_REGRESS**

See Also:
Constant Field Values

---

byteOrder

public static volatile java.nio.ByteOrder **byteOrder**

CurrentMagicValue

public static final java.lang.Byte **CurrentMagicValue**

Serialization Format is : MagicByte (1 byte) HeaderCrc (4 bytes) // Crc to protect the header from being corrupted Length (4 bytes) Attributes (2 byte) // Key-type, Trace marker, Event-opcode Sequence (8 bytes) // Sequence number for the event window in which this event was generated Physical PartitionId (2 byte) // Short physical partition-id -> represents a sequence generator Logical PartitionId (2 byte) // Short logical partition-id -> represents a logical partition of the physical stream NanoTimestamp (8 bytes) // Time (in nanoseconds) at which the event was generated SrcId (short) // SourceId for the event SchemaId (short) // 16-byte hash for the schema used to generate the event ValueCrc (4 bytes) // Crc to protect the value from being corrupted Key (8 bytes long key) or KeySize (4 bytes for byte[] key) Key Bytes (k bytes for byte[] key)

Value (N bytes) // Serialized Event

MagicLength

public static final int **MagicLength**

See Also:
Constant Field Values

LengthOffset

public static final int **LengthOffset**

See Also:
Constant Field Values

LengthLength

public static final int **LengthLength**

See Also:
Constant Field Values

SequenceOffset

public static final int **SequenceOffset**

See Also:
Constant Field Values

SequenceLength

public static final int **SequenceLength**

See Also:
Constant Field Values

PhysicalPartitionIdOffset

public static final int **PhysicalPartitionIdOffset**

See Also:
Constant Field Values

PhysicalPartitionIdLength

public static final int **PhysicalPartitionIdLength**

See Also:
Constant Field Values

LogicalPartitionIdOffset

public static final int **LogicalPartitionIdOffset**

See Also:
Constant Field Values

LogicalPartitionIdLength

public static final int **LogicalPartitionIdLength**

See Also:
Constant Field Values

TimestampOffset

public static final int **TimestampOffset**

See Also:
Constant Field Values

TimestampLength

public static final int **TimestampLength**

See Also:
Constant Field Values

SrcIdOffset

public static final int **SrcIdOffset**

See Also:
Constant Field Values

SrcIdLength

public static final int **SrcIdLength**

See Also:
Constant Field Values

---

emptymd5

public static byte[] **emptymd5**

EOPMarkerValue

public static final byte[] **EOPMarkerValue**

Constructor Detail

DbusEvent

public **DbusEvent**(java.nio.ByteBuffer buf,
int position)

DbusEvent

public **DbusEvent**()

Method Detail

createErrorEvent

public static DbusEvent **createErrorEvent**(DbusErrorEvent errorEvent)

---

getErrorEventFromDbusEvent

public static DbusErrorEvent **getErrorEventFromDbusEvent**(DbusEvent event)

---

createSCNRegressEvent

public static DbusEvent **createSCNRegressEvent**(SCNRegressMessage message)

Utility method to create a DbusEvent with a SCNRegress message in it.

Parameters:

checkpoint -

Returns:

---

getSCNRegressFromEvent

public static SCNRegressMessage **getSCNRegressFromEvent**(DbusEvent event)

Utility method to extract a SCN regress message from a DbusEvent Note: Ensure that this is a SCNRegressMessage event before calling this method.

Parameters:

event -

Returns:

---

createCheckpointEvent

public static DbusEvent **createCheckpointEvent**(Checkpoint checkpoint)

Utility method to create a DbusEvent with an embedded checkpoint in it.

Parameters:

checkpoint -

Returns:

---

getCheckpointFromEvent

public static Checkpoint **getCheckpointFromEvent**(DbusEvent event)

Utility method to extract a checkpoint from a DbusEvent Note: Ensure that this is a Checkpoint event before calling this method.

Parameters:

event -

Returns:

---

serializeEndOfPeriodMarker

```
public static int serializeEndOfPeriodMarker(java.nio.ByteBuffer serializationBuffer,
                                              DbusEventInfo eventInfo)
```

Serializes an End-Of-Period Marker onto the ByteBuffer passed in.

Parameters:

sequence - - The sequence to store on the EOP marker
timeStamp - - The timestamp to use for the EOP marker
serializationBuffer - - The ByteBuffer to serialize the event in. The buffer must have enough space to accommodate the event. (76 bytes)

Returns:

the number of bytes written

---

serializeEvent

```
public static int serializeEvent(DbusEventKey key,
                                 short pPartitionId,
                                 short lPartitionId,
                                 long timeStampInNanos,
                                 short srcId,
                                 byte[] schemaId,
                                 byte[] value,
                                 boolean enableTracing,
                                 java.nio.ByteBuffer serializationBuffer)
                          throws KeyTypeNotImplementedException
```

non-threadsafe : serializationBuffer needs to be protected if multiple threads are writing to it concurrently

Parameters:

key -
logicalPartitionId -
timeStampInNanos -
srcId -
schemaId -
value -
enableTracing -
serializationByteBuffer -

Throws:

KeyTypeNotImplementedException

---

serializeEvent

```
public static int serializeEvent(DbusEventKey key,
                                 java.nio.ByteBuffer serializationBuffer,
                                 DbusEventInfo dbusEventInfo)
                          throws KeyTypeNotImplementedException
```

Throws:

KeyTypeNotImplementedException

---

serializeFullEvent

```
public static int serializeFullEvent(DbusEventKey key,
                                     java.nio.ByteBuffer serializationBuffer,
                                     DbusEventInfo eventInfo)
```

---

getOpcode

```
public DbusOpcode getOpcode()
```

Description copied from interface: DataChangeEvent

Returns the opcode of the data event; null for non-data events

Specified by:

getOpcode in interface DataChangeEvent

---

setTraceFlag

```
public static void setTraceFlag(byte[] attribute)
```

---

isTraceEnabled

```
public boolean isTraceEnabled()
```

Description copied from interface: DataChangeEvent

Returns true if tracing has been enabled for the event

Specified by:

isTraceEnabled in interface DataChangeEvent

---

isKeyString

public boolean **isKeyString**()

Description copied from interface: DataChangeEvent
Returns true iff the key of th event is a string (byte sequence)

Specified by:
isKeyString in interface DataChangeEvent

---

isKeyNumber

public boolean **isKeyNumber**()

Description copied from interface: DataChangeEvent
Returns true iff the key of the event is a numeric (long)

Specified by:
isKeyNumber in interface DataChangeEvent

---

isControlMessage

public boolean **isControlMessage**()

Description copied from interface: DataChangeEvent
Checks if the event is a control message

Specified by:
isControlMessage in interface DataChangeEvent

---

isControlSrcId

public static boolean **isControlSrcId**(short srcId)

---

isEndOfPeriodMarker

public boolean **isEndOfPeriodMarker**()

Description copied from interface: DataChangeEvent
Checks if the event denotes the end of an event window

Specified by:

isEndOfPeriodMarker in interface DataChangeEvent

---

isPrivateControlMessage

public boolean **isPrivateControlMessage**()

Description copied from interface: DataChangeEvent
Checks if the event is a private control message

Specified by:
isPrivateControlMessage in interface DataChangeEvent

---

isCheckpointMessage

public boolean **isCheckpointMessage**()

Description copied from interface: DataChangeEvent
Checks if the event is a checkpoint message

Specified by:
isCheckpointMessage in interface DataChangeEvent

---

isSCNRegressMessage

public boolean **isSCNRegressMessage**()

Description copied from interface: DataChangeEvent
Checks if the event is a SCNRegressMessage

Specified by:
isSCNRegressMessage in interface DataChangeEvent

---

isErrorEvent

public boolean **isErrorEvent**()

---

isExtReplicatedEvent

public boolean **isExtReplicatedEvent**()

---

setExtReplicationFlag

```
public static void setExtReplicationFlag(byte[] attribute)
```

---

reset

```
public void reset(java.nio.ByteBuffer buf,
                  int position)
```

---

length

```
public static int length(DbusEventKey key,
                         byte[] value)
                  throws KeyTypeNotImplementedException
```

Throws:

KeyTypeNotImplementedException

---

unsetInited

```
public void unsetInited()
```

---

applyCrc

```
public void applyCrc()
```

---

headerLength

```
public int headerLength()
```

---

payloadLength

```
public int payloadLength()
```

---

setSequence

```
public void setSequence(long sequence)
```

---

sequence

public long **sequence**()

Description copied from interface: DataChangeEvent
Returns the sequence number of the event

Specified by:
sequence in interface DataChangeEvent

---

keyLength

public int **keyLength**()

Description copied from interface: DataChangeEvent
Returns the length of the event key

Specified by:
keyLength in interface DataChangeEvent

---

valueLength

public int **valueLength**()

Description copied from interface: DataChangeEvent
Returns the length of the data event value (data payload)

Specified by:
valueLength in interface DataChangeEvent

---

isValid

public boolean **isValid**()

Description copied from interface: DataChangeEvent
Returns true iff the event points to a valid Databus event

Specified by:
isValid in interface DataChangeEvent

---

isPartial

```
public boolean isPartial()
```

---

scanHeader

```
public DbusEvent.HeaderScanStatus scanHeader()
```

---

scanEvent

```
public DbusEvent.EventScanStatus scanEvent()
```

---

scanHeader

```
protected DbusEvent.HeaderScanStatus scanHeader(boolean logErrors)
```

Parameters:

logErrors -

Returns:

PARTIAL if the event appears to be a partial event; ERR if the header is corrupt; OK if the event header is intact and the event appears to be complete

---

scanEvent

```
protected DbusEvent.EventScanStatus scanEvent(boolean logErrors)
```

Parameters:

logErrors -

Returns:

one of ERR/ OK / PARTIAL

---

isPartial

```
protected boolean isPartial(boolean logErrors)
```

Parameters:

logErrors -

Returns:

true if the event appears to be partially read ; does not perform any header checks

---

isHeaderPartial

protected boolean **isHeaderPartial**(boolean logErrors)

---

isHeaderValid

protected boolean **isHeaderValid**(boolean logErrors)

Checks if the event header - containing length is vsalid

Parameters:

logErrors -

Returns:

true iff header is devoid of errors; the length field can be trusted.

---

isValid

protected boolean **isValid**(boolean logErrors)

Parameters:

logErrors - : whether to emit LOG.error messages for invalid results

Returns:

true if event is not partial and event is valid; Note that a partial event is deemed invalid;

---

toString

public java.lang.String **toString**()

Overrides:

toString in class java.lang.Object

---

magic

public java.lang.Byte **magic**()

---

size

public int **size**()

Description copied from interface: DataChangeEvent

Returns the total size of the event binary serialization

Specified by:
size in interface DataChangeEvent

---

setSize

public void **setSize**(int sz)

Setter for size

Parameters:
sz -

---

headerCrc

public long **headerCrc**()

---

setHeaderCrc

public void **setHeaderCrc**(long crc)

---

key

public long **key**()

Description copied from interface: DataChangeEvent
Returns key value for events with numeric keys; undefined for events with string keys.

Specified by:
key in interface DataChangeEvent

---

keyBytes

public byte[] **keyBytes**()

Description copied from interface: DataChangeEvent
Returns the key value for events with string keys; undefined for events with numeric keys.

Specified by:
keyBytes in interface DataChangeEvent

---

physicalPartitionId

public short **physicalPartitionId**()

Description copied from interface: DataChangeEvent
Returns the physical partition id for the event

Specified by:
physicalPartitionId in interface DataChangeEvent

---

logicalPartitionId

public short **logicalPartitionId**()

Description copied from interface: DataChangeEvent
Returns the logical partition id for the event

Specified by:
logicalPartitionId in interface DataChangeEvent

---

timestampInNanos

public long **timestampInNanos**()

Description copied from interface: DataChangeEvent
Returns the creation timestamp of the event in nanoseconds from Unix epoch

Specified by:
timestampInNanos in interface DataChangeEvent

---

setSchemaVersion

public void **setSchemaVersion**(short schemaVersion)

---

schemaVersion

public short **schemaVersion**()

---

setSrcId

```
public void setSrcId(short srcId)
```

---

srcId

```
public short srcId()
```

Description copied from interface: DataChangeEvent
Returns the Databus source id of the event

Specified by:
srcId in interface DataChangeEvent

---

setSchemaId

```
public void setSchemaId(byte[] schemaId)
```

put a byte[] schemaId into the buffer . Make sure CRC is recomputed after that

---

schemaId

```
public byte[] schemaId()
```

Description copied from interface: DataChangeEvent
Returns a byte array with the hash id of the event serialization schema.

NOTE: this will most likely lead to a memory allocation. The preferred way to access the schema id is through {@link DataChangeEvent#schemaId(byte[]).

Specified by:
schemaId in interface DataChangeEvent

---

schemaId

```
public void schemaId(byte[] md5)
```

Description copied from interface: DataChangeEvent
Stores the hash id of the event serialization schema in an existing byte

array.

NOTE: The byte array should be at least 16 bytes long.

Specified by:
schemaId in interface DataChangeEvent

---

valueCrc

public long **valueCrc**()

---

setValueCrc

public void **setValueCrc**(long crc)

---

value

public java.nio.ByteBuffer **value**()

Description copied from interface: DataChangeEvent
Obtains the data payload of the event

Specified by:
value in interface DataChangeEvent

---

setValue

public void **setValue**(byte[] bytes)

---

getRawBytes

public java.nio.ByteBuffer **getRawBytes**()

---

createCopy

public DbusEvent **createCopy**()

---

writeTo

```
public int writeTo(java.nio.channels.WritableByteChannel writeChannel,
                   Encoding encoding)
```

Serializes the event to a channel using the specified encoding

Parameters:

`writeChannel` - the channel to write to
`encoding` - the serialization encoding

Returns:

the number of bytes written to the channel

---

appendToEventBuffer

```
public static int appendToEventBuffer(java.lang.String jsonString,
                                      DbusEventBufferAppendable eventBuffer,
                                      DbusEventsStatisticsCollector statsCollector,
                                      boolean startWindow)
                               throws java.io.IOException,
                                      org.codehaus.jackson.JsonParseException,
                                      InvalidEventException,
                                      KeyTypeNotImplementedException
```

Appends a single event to the buffer. The event is

Parameters:

`jsonString` -
`eventBuffer` -

Returns:

Throws:

`java.io.IOException`
`org.codehaus.jackson.JsonParseException`
`InvalidEventException`
`KeyTypeNotImplementedException`

---

appendToEventBuffer

```
public static int appendToEventBuffer(java.io.BufferedReader jsonStream,
                                      DbusEventBufferAppendable eventBuffer,
                                      DbusEventsStatisticsCollector statsCollector,
                                      boolean startWindow)
                               throws java.io.IOException,
                                      org.codehaus.jackson.JsonParseException,
                                      InvalidEventException
```

Throws:

`java.io.IOException`

org.codehaus.jackson.JsonParseException
InvalidEventException

---

inited

public boolean **inited**()

---

equals

public boolean **equals**(java.lang.Object obj)

Overrides:

equals in class java.lang.Object

---

hashCode

public int **hashCode**()

Overrides:

hashCode in class java.lang.Object

---

clone

public DbusEvent **clone**(DbusEvent reuse)

Creates a copy of the current event.

Note: This method should be used with extreme care as the event serialization pointed by the object can be overwritten. It should be used only in buffers with BLOCK_ON_WRITE policy. Further, the object should not be used after DbusEventBuffer.DbusEventIterator.remove()

Parameters:

reuse - an existing object to reuse; if null, a new object will be created

Returns:

the event copy
