

IBM® Netezza® Analytics
Release 11.x

*C++ Analytic Executables
API Reference*



Note: Before using this information and the product that it supports, read the information in "[Notices and Trademarks](#)" on page 225.

Contents

Preface

| | |
|------------------------------------|------|
| Audience for This Guide..... | xiii |
| Purpose of This Guide..... | xiii |
| Conventions..... | xiii |
| If You Need Help..... | xiii |
| Comments on the Documentation..... | xiv |

Module Documentation

| | |
|--------------------------------------|----|
| Initialization APIs | 15 |
| Classes..... | 15 |
| Modules | 15 |
| Detailed Description | 16 |
| Remote Initialization..... | 16 |
| Data Connection APIs..... | 16 |
| Modules | 17 |
| Detailed Description | 17 |
| Function | 17 |
| Aggregate | 17 |
| Shaper and Sizer | 17 |
| Record and Data Type Support..... | 18 |
| Classes..... | 18 |
| Modules | 18 |
| Enumerations | 18 |
| Detailed Description | 19 |
| Enumeration Type Documentation | 19 |
| Integer Fields..... | 19 |
| Numeric Fields..... | 20 |
| String Fields..... | 20 |
| Temporal Fields | 21 |
| Support APIs | 21 |
| Classes..... | 22 |
| Modules | 22 |
| Detailed Description | 22 |

| | |
|--|----|
| Runtime and Environment Information..... | 22 |
| Namespace Documentation | |
| nz..... | 23 |
| Namespaces | 23 |
| nz::ae..... | 23 |
| Functions..... | 23 |
| Function Documentation | 29 |

3 Class Documentation

| | |
|---|----|
| NzaeAggregate Class Reference | 65 |
| Public Types..... | 65 |
| Public Member Functions | 65 |
| Static Public Member Functions..... | 66 |
| Detailed Description | 66 |
| Enumeration Type Documentation | 66 |
| Typedef Documentation | 67 |
| Public Member Function Documentation..... | 67 |
| Static Public Member Function Documentation | 69 |
| NzaeAggregateInitialization Class Reference | 69 |
| Detailed Description | 69 |
| NzaeAggregateMessageHandler Interface Reference | 69 |
| Public Member Functions | 70 |
| Detailed Description | 70 |
| Public Member Function Documentation..... | 70 |
| NzaeApi Class Reference | 71 |
| Public Types..... | 72 |
| Public Member Functions | 72 |
| Public Attributes | 72 |
| Detailed Description | 72 |
| Enumeration Type Documentation | 72 |
| Public Member Function Documentation..... | 73 |
| Member Data Documentation | 73 |
| NzaeApiGenerator Class Reference | 73 |
| Public Member Functions | 73 |
| Detailed Description | 74 |
| Public Member Function Documentation..... | 74 |

| | |
|---|-----|
| NzaeBoolField Class Reference | 78 |
| Public Member Functions | 78 |
| Detailed Description | 78 |
| Public Member Function Documentation | 78 |
| NzaeCallbackResult Struct Reference | 80 |
| Public Attributes | 80 |
| Detailed Description | 80 |
| Member Data Documentation | 80 |
| NzaeConnectionPoint Class Reference | 81 |
| Public Member Functions | 81 |
| Static Public Member Functions | 82 |
| Detailed Description | 82 |
| Public Member Function Documentation | 82 |
| Static Public Member Function Documentation | 84 |
| NzaeDataTypes Class Reference | 84 |
| Public Types | 84 |
| Detailed Description | 85 |
| Enumeration Type Documentation | 85 |
| NzaeDateField Class Reference | 86 |
| Public Member Functions | 86 |
| Static Public Member Functions | 87 |
| Detailed Description | 88 |
| Public Member Function Documentation | 88 |
| Static Public Member Function Documentation | 93 |
| NzaeDoubleField Class Reference | 95 |
| Public Member Functions | 95 |
| Detailed Description | 96 |
| Public Member Function Documentation | 96 |
| NzaeEnvironment Class Reference | 98 |
| Public Member Functions | 98 |
| Static Public Member Functions | 98 |
| Detailed Description | 98 |
| Public Member Function Documentation | 98 |
| Static Public Member Function Documentation | 99 |
| NzaeException Class Reference | 99 |
| Public Member Functions | 100 |
| Static Public Member Functions | 100 |
| Detailed Description | 100 |

| | |
|--|-----|
| Public Member Function Documentation..... | 100 |
| Static Public Member Function Documentation | 100 |
| NzaeFactory Class Reference..... | 100 |
| Public Member Functions | 100 |
| Static Public Member Functions..... | 101 |
| Detailed Description | 101 |
| Public Member Function Documentation..... | 101 |
| Static Public Member Function Documentation | 104 |
| NzaeField Interface Reference | 104 |
| Public Member Functions | 105 |
| Detailed Description | 105 |
| Public Member Function Documentation..... | 106 |
| NzaeFixedStringField Class Reference | 107 |
| Public Member Functions | 107 |
| Detailed Description | 107 |
| Public Member Function Documentation..... | 107 |
| NzaeFloatField Class Reference | 108 |
| Public Member Functions | 108 |
| Detailed Description | 108 |
| Public Member Function Documentation..... | 109 |
| NzaeFunction Class Reference..... | 110 |
| Public Types..... | 110 |
| Public Member Functions | 110 |
| Static Public Member Functions..... | 111 |
| Detailed Description | 111 |
| Enumeration Type Documentation | 112 |
| Public Member Function Documentation..... | 112 |
| Static Public Member Function Documentation | 115 |
| NzaeFunctionInitialization Class Reference | 115 |
| Detailed Description | 116 |
| NzaeFunctionMessageHandler Interface Reference | 116 |
| Public Member Functions | 116 |
| Detailed Description | 116 |
| Public Member Function Documentation..... | 116 |
| NzaeGeometryStringField Class Reference | 117 |
| Public Member Functions | 117 |
| Detailed Description | 117 |

| | |
|---|-----|
| Public Member Function Documentation..... | 117 |
| NzaeInt16Field Class Reference..... | 117 |
| Public Member Functions | 118 |
| Detailed Description | 118 |
| Public Member Function Documentation..... | 118 |
| NzaeInt32Field Class Reference..... | 120 |
| Public Member Functions | 120 |
| Detailed Description | 121 |
| Public Member Function Documentation..... | 121 |
| NzaeInt64Field Class Reference..... | 122 |
| Public Member Functions | 123 |
| Detailed Description | 123 |
| Public Member Function Documentation..... | 123 |
| NzaeInt8Field Class Reference | 125 |
| Public Member Functions | 125 |
| Detailed Description | 126 |
| Public Member Function Documentation..... | 126 |
| NzaeIntervalField Class Reference..... | 127 |
| Public Member Functions | 127 |
| Detailed Description | 128 |
| Public Member Function Documentation..... | 129 |
| NzaeLibrary Class Reference | 132 |
| Public Types..... | 132 |
| Public Member Functions | 132 |
| Static Public Member Functions..... | 133 |
| Detailed Description | 133 |
| Enumeration Type Documentation | 133 |
| Typedef Documentation | 133 |
| Public Member Function Documentation..... | 133 |
| Static Public Member Function Documentation | 135 |
| NzaeLibraryInfo Class Reference | 135 |
| Public Attributes..... | 135 |
| Detailed Description | 136 |
| Member Data Documentation | 136 |
| NzaeMetadata Class Reference | 136 |
| Public Types..... | 136 |
| Public Member Functions | 136 |

| | |
|--|-----|
| Detailed Description | 137 |
| Enumeration Type Documentation | 137 |
| Typedef Documentation | 138 |
| Public Member Function Documentation..... | 138 |
| NzaeNationalFixedStringField Class Reference | 141 |
| Public Member Functions | 141 |
| Detailed Description | 141 |
| Public Member Function Documentation..... | 141 |
| NzaeNationalVariableStringField Class Reference..... | 142 |
| Public Member Functions | 142 |
| Detailed Description | 142 |
| Public Member Function Documentation..... | 142 |
| NzaeNumeric128Field Class Reference..... | 143 |
| Public Member Functions | 143 |
| Detailed Description | 144 |
| Public Member Function Documentation..... | 144 |
| NzaeNumeric32Field Class Reference..... | 148 |
| Public Member Functions | 148 |
| Detailed Description | 149 |
| Public Member Function Documentation..... | 149 |
| NzaeNumeric64Field Class Reference..... | 153 |
| Public Member Functions | 153 |
| Detailed Description | 154 |
| Public Member Function Documentation..... | 154 |
| NzaeNumericField Class Reference | 158 |
| Public Member Functions | 158 |
| Static Public Member Functions..... | 161 |
| Detailed Description | 161 |
| Public Member Function Documentation..... | 161 |
| Static Public Member Function Documentation | 172 |
| NzaeParameters Class Reference | 173 |
| Public Member Functions | 174 |
| Static Public Member Functions..... | 174 |
| Detailed Description | 174 |
| Public Member Function Documentation..... | 174 |
| Static Public Member Function Documentation | 175 |
| NzaeRecord Class Reference | 175 |

| | |
|--|-----|
| Public Member Functions | 175 |
| Detailed Description | 175 |
| Public Member Function Documentation..... | 175 |
| NzaeRemoteProtocol Class Reference..... | 176 |
| Public Member Functions | 176 |
| Detailed Description | 177 |
| Public Member Function Documentation..... | 177 |
| NzaeRemoteProtocolCallback Class Reference | 179 |
| Public Types..... | 179 |
| Public Member Functions | 179 |
| Detailed Description | 179 |
| Enumeration Type Documentation | 179 |
| Public Member Function Documentation..... | 179 |
| NzaeRuntime Class Reference | 180 |
| Public Types..... | 180 |
| Public Member Functions | 180 |
| Public Attributes..... | 181 |
| Detailed Description | 181 |
| Enumeration Type Documentation | 181 |
| Public Member Function Documentation..... | 182 |
| Member Data Documentation | 184 |
| NzaeShaper Class Reference | 184 |
| Public Types..... | 185 |
| Public Member Functions | 185 |
| Static Public Member Functions..... | 186 |
| Detailed Description | 186 |
| Enumeration Type Documentation | 186 |
| Public Member Function Documentation..... | 186 |
| Static Public Member Function Documentation | 191 |
| NzaeShaperInitialization Class Reference | 191 |
| Detailed Description | 191 |
| NzaeShaperMessageHandler Interface Reference | 191 |
| Public Member Functions | 191 |
| Detailed Description | 191 |
| Public Member Function Documentation..... | 191 |
| NzaeShaperOutputColumn Class Reference | 192 |
| Detailed Description | 192 |
| NzaeShaperOutputColumnInfo Class Reference | 192 |

| | |
|---|-----|
| Public Attributes | 192 |
| Member Data Documentation | 192 |
| NzaeStringField Class Reference..... | 193 |
| Public Member Functions | 193 |
| Detailed Description | 194 |
| Public Member Function Documentation..... | 194 |
| NzaeTimeField Class Reference | 196 |
| Public Member Functions | 196 |
| Static Public Member Functions..... | 197 |
| Detailed Description | 197 |
| Public Member Function Documentation..... | 198 |
| Static Public Member Function Documentation | 203 |
| NzaeTimestampField Class Reference | 203 |
| Public Member Functions | 203 |
| Static Public Member Functions..... | 205 |
| Detailed Description | 205 |
| Public Member Function Documentation..... | 206 |
| Static Public Member Function Documentation | 213 |
| NzaeTimeTzField Class Reference..... | 214 |
| Public Member Functions | 214 |
| Static Public Member Functions..... | 215 |
| Detailed Description | 216 |
| Public Member Function Documentation..... | 216 |
| Static Public Member Function Documentation | 222 |
| NzaeVarbinaryStringField Class Reference..... | 223 |
| Public Member Functions | 223 |
| Detailed Description | 223 |
| Public Member Function Documentation..... | 223 |
| NzaeVariableStringField Class Reference | 223 |
| Public Member Functions | 223 |
| Detailed Description | 224 |
| Public Member Function Documentation..... | 224 |

Notices and Trademarks

| | |
|---------------------------------|-----|
| Notices | 225 |
| Trademarks..... | 226 |
| Regulatory and Compliance | 227 |

| | |
|--------------------------------------|-----|
| Regulatory Notices | 227 |
| Homologation Statement | 227 |
| FCC - Industry Canada Statement..... | 227 |
| CE Statement (Europe) | 227 |
| VCCI Statement | 227 |

Index

Preface

This guide provides an API reference for C++ AE programmers.

Audience for This Guide

The *C++ Analytic Executables API Reference* is written for programmers who intend to create Analytic Executables for IBM Netezza Analytics using the C++ language. This guide does not provide a tutorial on AE concepts. More information about AEs can be found in the *User-Defined Analytic Process De-veloper's Guide*.

Purpose of This Guide

This guide describes the C++ AE API, which is a language adapter provided as part of IBM Netezza Analytics. The C++ AE API provides programmatic access to the AE interface for C++ programmers.

Conventions

Note on Terminology: The terms User-Defined Analytic Process (UDAP) and Analytic Executable (AE) are synonymous.

The following conventions apply:

Italics for emphasis on terms and user-defined values, such as user input.

Upper case for SQL commands, for example, INSERT or DELETE.

Bold for command line input, for example, **nzsystem stop**.

Bold to denote parameter names, argument names, or other named references.

Angle brackets (< >) to indicate a placeholder (variable) that should be replaced with actual text, for example, **nzmat** <- **nz.matrix("<matrix_name>")**.

A single backslash ("\") at the end of a line of code to denote a line continuation. Omit the back-slash when using the code at the command line, in a SQL command, or in a file.

When referencing a sequence of menu and submenu selections, the ">" character denotes the different menu options, for example *Menu Name > Submenu Name > Selection*.

If You Need Help

If you are having trouble using the IBM Netezza appliance, IBM Netezza Analytics or any of its components:

Retry the action, carefully following the instructions in the documentation.

Go to the IBM Support Portal at <http://www.ibm.com/support>. Log in using your IBM ID and password. You can search the Support Portal for solutions. To submit a support re-quest, click the 'Service Requests & PMRs' tab.

If you have an active service contract maintenance agreement with IBM, you can contact customer support teams via telephone. For individual countries, please visit the Technical Support section of the IBM Directory of worldwide contacts

Comments on the Documentation

We welcome any questions, comments, or suggestions that you have for the IBM Netezza document-ation. Please send us an e-mail message at netezza-doc@wwpdl.vnet.ibm.com and include the fol-lowing information:

- The name and version of the manual that you are using
 - Any comments that you have about the manual
 - Your name, address, and phone number
- We appreciate your comments.

CHAPTER 1

Module Documentation

Initialization APIs

This API family is used to make an open data connection or to get an AE Environment, which can then be used to open a data connection.

Classes

class NzaeAggregateInitialization

Not implemented. Placeholder reserved for future use.

class NzaeApiGenerator

Helper class for getting an API object.

class NzaeFactory

This class is used to get an API object.

class NzaeFunctionInitialization

Not implemented. This class is a placeholder for future functionality.

class NzaeShaperInitialization

Not implemented. This class is a placeholder for future functionality.

Modules

Remote Initialization

Initialization classes related to Remote AEs. They are used to:

- Create a connection point.

- Listen using that connection point.

- Accept a Data Connection API handle or accept an AE Environment.

Detailed Description

This API family is used to make an open data connection or to get an AE Environment, which can then be used to open a data connection.

These classes are called first in an AE program to perform initialization tasks. For initialization using default system values see class `NzaeApiGenerator` . For initialization using custom options see `Nza-eFactory` .

`NzaeApiGenerator` supports both a "standard input / output" data flow paradigm and a call back paradigm.

Remote Initialization

Initialization classes related to Remote AEs. They are used to:

- Create a connection point.

- Listen using that connection point.

- Accept a Data Connection API handle or accept an AE Environment.

Classes

struct `NzaeCallbackResult`

Struct used to specify the callback result.

class `NzaeConnectionPoint`

Class to encapsulate the connection point for remote mode AEs.

class `NzaeRemoteProtocol`

Class to get an API object in Remote Mode.

class `NzaeRemoteProtocolCallback`

Class to handle callbacks for remote protocol mode.

Detailed Description

Initialization classes related to Remote AEs. They are used to:

- Create a connection point.

- Listen using that connection point.

- Accept a Data Connection API handle or accept an AE Environment.

Remote AEs may also be used to setup a remote protocol callback handler to handle status, ping, stop and signal.

Data Connection APIs

This API family is used to process data after a data connection has been opened.

Modules

Function

Function AEs are called from SQL Scalar or Table Functions.

Aggregate

Aggregate AEs are called from SQL Aggregate Functions.

Shaper and Sizer

Shapers are optionally called for Table Function AEs.

Detailed Description

This API family is used to process data after a data connection has been opened.

See Also

▲ Initialization APIs

Function

Function AEs are called from SQL Scalar or Table Functions.

Classes

class NzaeFunction

This class provides Function functionality and is used to implement Function AEs.

interface NzaeFunctionMessageHandler

This class allows implementation of higher level functions.

class NzaeMetadata

This class provides AE Metadata information, containing data about the AE, including input and output column attributes. Column indexes are zero-based.

Detailed Description

Function AEs are called from SQL Scalar or Table Functions.

Aggregate

Aggregate AEs are called from SQL Aggregate Functions.

Classes

class NzaeAggregate

This class provides Aggregate functionality and is used to implement Aggregation AEs.

interface NzaeAggregateMessageHandler

This class provides Aggregate functionality.

Detailed Description

Aggregate AEs are called from SQL Aggregate Functions.

Shaper and Sizer

Shapers are optionally called for Table Function AEs.

Classes

class NzaeShaper

This class provides Shaper or Sizer functionality.

interface NzaeShaperMessageHandler

This class provides higher level shaper implementation.

class NzaeShaperOutputColumn

This class provides Shaper output information.

Detailed Description

Shapers are optionally called for Table Function AEs.

Sizers are optionally called for Scalar Function AEs.

Record and Data Type Support

All the data APIs work with records that are collections of data fields.

Classes

interface NzaeField

Provides the field interface.

class NzaeRecord

This class provides an AE record.

Modules

Integer Fields

These are fields that are integral.

Numeric Fields

These are fields that are numeric.

String Fields

These are fields that are strings.

Temporal Fields

These are fields that are temporal types.

Enumerations

```
enum Types {  
  NZUDSUDX_UNKNOWN= -1, NZUDSUDX_FIXED= 0, NZUDSUDX_VARIABLE= 1, NZUDSUDX_NA-  
  TIONAL_FIXED= 2, NZUDSUDX_NATIONAL_VARIABLE= 3, NZUDSUDX_BOOL= 4,  
  NZUDSUDX_DATE= 5, NZUDSUDX_TIME= 6, NZUDSUDX_TIMETZ= 7, NZUDSUDX_NUMERIC32=  
  8, NZUDSUDX_NUMERIC64= 9, NZUDSUDX_NUMERIC128= 10, NZUDSUDX_FLOAT= 11, NZUD-  
  SUDX_DOUBLE= 12, NZUDSUDX_INTERVAL= 13, NZUDSUDX_INT8= 14, NZUDSUDX_INT16= 15,  
  NZUDSUDX_INT32= 16, NZUDSUDX_INT64= 17, NZUDSUDX_TIMESTAMP= 18,
```

NZUDSUDX_GEOMETRY= 19, NZUDSUDX_VARBINARY= 20, NZUDSUDX_MAX_TYPE= 21 } Data types that match the Netezza system types.

Detailed Description

All the data APIs work with records that are collections of data fields.

For overloaded operators for data types see `nz::ae`

Enumeration Type Documentation

enum Types

Data types that match the Netezza system types.

NZUDSUDX_UNKNOWN Unknown data type

NZUDSUDX_FIXED Fixed string **NZUDSUDX_VARIABLE**

Variable string **NZUDSUDX_NATIONAL_FIXED** Fixed

national string **NZUDSUDX_NATIONAL_VARIABLE**

Variable national string **NZUDSUDX_BOOL** Boolean

NZUDSUDX_DATE Date

NZUDSUDX_TIME Time

NZUDSUDX_TIMETZ Time zone

NZUDSUDX_NUMERIC32 Numeric 32

NZUDSUDX_NUMERIC64 Numeric 64

NZUDSUDX_NUMERIC128 Numeric 128

NZUDSUDX_FLOAT Float

NZUDSUDX_DOUBLE Double

NZUDSUDX_INTERVAL Interval

NZUDSUDX_INT8 1 byte integer

NZUDSUDX_INT16 2 byte integer

NZUDSUDX_INT32 4 byte integer

NZUDSUDX_INT64 8 byte integer

NZUDSUDX_TIMESTAMP Time stamp

NZUDSUDX_GEOMETRY Geometry

NZUDSUDX_VARBINARY Variable Binary

NZUDSUDX_MAX_TYPE Greater than any data type enum value

Integer Fields

These are fields that are integral.

Classes

class NzaeBoolField

This class provides field access for type bool.

class NzaeInt16Field

This class provides field access for type int16.

class NzaeInt32Field

This class provides field access for type int32.

class NzaeInt64Field

This class provides field access for type int64.

class NzaeInt8Field

This class provides field access for type int8.

Detailed Description

These are fields that are integral.

Numeric Fields

These are fields that are numeric.

Classes

class NzaeDoubleField

This class provides field access for type double.

class NzaeFloatField

This class provides field access for type float.

class NzaeNumeric128Field

This class provides field access for type Numeric128.

class NzaeNumeric32Field

This class provides field access for type Numeric32.

class NzaeNumeric64Field

This class provides field access for type Numeric64.

class NzaeNumericField

This class provides a common base class for the NzaeNumeric32Field , NzaeNumeric64Field , and NzaeNumeric128Field field classes.

Detailed Description

These are fields that are numeric.

String Fields

These are fields that are strings.

Classes

class NzeFixedStringField

This class provides field access for type fixed string.

class NzeGeometryStringField

This class provides field access for type geometry string.

class NzeNationalFixedStringField

This class provides field access for type national fixed string.

class NzeNationalVariableStringField

This class provides field access for type national variable string.

class NzeStringField

This class provides a common base class for the NzeFixedStringField , NzeVariableStringField , NzeNationalFixedStringField , NzeNationalVariableStringField , NzeGeometryStringField and NzeVarbinaryStringField classes.

class NzeVarbinaryStringField

This class provides field access for type varbinary string.

class NzeVariableStringField

This class provides field access for type variable string.

Detailed Description

These are fields that are strings.

Temporal Fields

These are fields that are temporal types.

Classes

class NzeDateField

This class provides field access for type date.

class NzeIntervalField

This class provides field access for type interval.

class NzeTimeField

This class provides field access for type time.

class NzeTimestampField

This class provides field access for type timestamp.

class NzeTimeTzField

This class provides field access for type timetz.

Detailed Description

These are fields that are temporal types.

Support APIs

Support Classes used in other API categories.

Classes

class NzaeException

This class is used for all C++ AE Exceptions.

Modules

Runtime and Environment Information

Runtime, Environment, and Shared Library Information. Runtime environment information after a data API has been obtained.

Detailed Description

Support Classes used in other API categories.

Runtime and Environment Information

Runtime, Environment, and Shared Library Information. Runtime environment information after a data API has been obtained.

Classes

class NzaeEnvironment

This class provides the AE Environment and lookup access to the AE environment.

class NzaeLibrary

This class provides access to the AE shared library information.

class NzaeLibraryInfo

This class provides information about an AE shared library.

class NzaeParameters

This class provides access to AE Parameters.

class NzaeRuntime

This class provides Runtime functionality.

Detailed Description

Runtime, Environment, and Shared Library Information. Runtime environment information after a data API has been obtained.

See Also

▲ [Data Connection APIs](#)

CHAPTER 2

Namespace Documentation

nz

Namespaces

`nz::ae`

nz::ae

Functions

`int nz::ae::operator!(const NzaeNumericField &lhs)` Logical Negation.

`NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(double lhs, NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(int64_t lhs, NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(int32_t lhs, NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, const NzaeNumericField &rhs)`
Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(double lhs, const NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(int64_t lhs, const NzaeNumericField &rhs)`

Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, NzaeNumericField &rhs)`

Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(int32_t lhs, const NzaeNumericField &rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, const NzaeNumericField &rhs)`

Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, double rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, double rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, int64_t rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, int64_t rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, int32_t rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, int32_t rhs)` Perform a modulus operation using the two specified values.

`NzaeNumeric128Field nz::ae::operator*(int32_t lhs, NzaeNumericField &rhs)` Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, NzaeNumericField &rhs)` Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, const NzaeNumericField &rhs)`

Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, double rhs)` Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, NzaeNumericField &rhs)`

Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(double lhs, const NzaeNumericField &rhs)` Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, double rhs)` Multiply the two specified values.

`NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, const NzaeNumericField &rhs)`

Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(double lhs, NzaeNumericField &rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, int64_t rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(int64_t lhs, const NzaeNumericField &rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, int64_t rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(int64_t lhs, NzaeNumericField &rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, int32_t rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(int32_t lhs, const NzaeNumericField &rhs) Multiply the two specified values.

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, int32_t rhs) Multiply the two specified values.

NzaeTimestampField nz::ae::operator+(const NzaeTimeField &time, const NzaeDateField &date) Add date and time.

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(int32_t lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(double lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, int64_t rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(int64_t lhs, const NzaeNumericField &rhs) Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs) Unary plus.

NzaeDateField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeDateField &date) Add an interval and a date.

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, double rhs) Perform an addition operation using the specified values.

NzaeTimestampField nz::ae::operator+(const NzaeDateField &date, const NzaeTimeField &time)

Add a date and a time.

`NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, int64_t rhs)` Perform an addition operation using the specified values.

`NzaeTimestampField nz::ae::operator+(const NzaeDateField &date, const NzaeTimeTzField &time)`

Add a date and a time.

`NzaeTimeField nz::ae::operator+(const NzaeTimeField &time, const NzaeIntervalField &iv)` Add an interval and a time.

`NzaeTimeTzField nz::ae::operator+(const NzaeTimeTzField &time, const NzaeIntervalField &iv)` Add an interval and a timetz.

`NzaeTimestampField nz::ae::operator+(const NzaeTimestampField &time, const NzaeIntervalField &iv)`

Add an interval and a timestamp.

`NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, const NzaeNumericField &rhs)`

Perform an addition operation using the specified values.

`NzaeDateField nz::ae::operator+(const NzaeDateField &date, const NzaeIntervalField &iv)` Add an interval and a date.

`NzaeNumeric128Field nz::ae::operator+(int64_t lhs, NzaeNumericField &rhs)` Perform an addition operation using the specified values.

`NzaeTimestampField nz::ae::operator+(const NzaeTimeTzField &time, const NzaeDateField &date)`

Add a date and a timetz.

`NzaeNumeric128Field nz::ae::operator+(double lhs, const NzaeNumericField &rhs)` Perform an addition operation using the specified values.

`NzaeTimeField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimeField &time)` Add an interval and a time.

`NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, int32_t rhs)` Perform an addition operation using the specified values.

`NzaeTimestampField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimestampField &time)`

Add an interval and a timestamp.

`NzaeTimeTzField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimeTzField &time)` Add an interval and a timetz.

`NzaeNumeric128Field nz::ae::operator+(int32_t lhs, const NzaeNumericField &rhs)` Perform an addition operation using the specified values.

`NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, double rhs)` Perform an addition operation using the specified values.

`NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, int32_t rhs)`

Perform an addition operation using the specified values.

NzaeNumeric128Field nz::ae::operator++(NzaeNumericField &lhs, int rhs) Increments one value by one.

NzaeTimeTzField nz::ae::operator-(const NzaeTimeTzField &time, const NzaeIntervalField &iv) Subtract an interval from timetz.

NzaeNumeric128Field nz::ae::operator-(int32_t lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeIntervalField nz::ae::operator-(const NzaeTimeField &time, const NzaeTimeField &t2) Subtract time from time.

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, double rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, double rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, int64_t rhs) Perform a subtraction operation using the specified values.

NzaeDateField nz::ae::operator-(const NzaeDateField &date, const NzaeIntervalField &iv) Subtract an interval from a date.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(int64_t lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, int64_t rhs) Perform a subtraction operation using the specified values.

NzaeTimestampField nz::ae::operator-(const NzaeTimestampField &time, const NzaeIntervalField &iv) Subtract an interval from a timestamp.

NzaeNumeric128Field nz::ae::operator-(double lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeTimeField nz::ae::operator-(const NzaeTimeField &time, const NzaeIntervalField &iv) Subtract an interval from a time.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(double lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(int64_t lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, int32_t rhs)

Perform a subtraction operation using the specified values.

NzaeIntervalField nz::ae::operator-(const NzaeTimestampField &time, const NzaeTimestampField &t2)

Subtract timestamp from timestamp.

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, int32_t rhs) Perform a subtraction operation using the specified values.

NzaeNumeric128Field nz::ae::operator-(int32_t lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

NzaeIntervalField nz::ae::operator-(const NzaeDateField &date, const NzaeDateField &d2) Subtract a date from a date.

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs) Unary minus.

NzaeNumeric128Field nz::ae::operator--(NzaeNumericField &lhs, int rhs) Decrements one value by one.

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, double rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, NzaeNumericField &rhs)

Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(int64_t lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, double rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, int32_t rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(int64_t lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(double lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, int64_t rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(int32_t lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

NzaeNumeric128Field nz::ae::operator/(int32_t lhs, NzaeNumericField &rhs)

Perform a division operation using the specified values.

NzaeNumeric128Field **nz::ae::operator/(NzaeNumericField &lhs, int32_t rhs)** Perform a division operation using the specified values.

NzaeNumeric128Field **nz::ae::operator/(double lhs, NzaeNumericField &rhs)** Perform a division operation using the specified values.

NzaeNumeric128Field **nz::ae::operator/(const NzaeNumericField &lhs, const NzaeNumericField &rhs)** Perform a division operation using the specified values.

NzaeNumeric128Field **nz::ae::operator/(NzaeNumericField &lhs, int64_t rhs)** Perform a division operation using the specified values.

Function Documentation

int **nz::ae::operator!(const NzaeNumericField &lhs)** Logical Negation.

Parameters

NzaeNumericField **lhs**
value

Returns

A value of 1 if lhs is equal to 0, 0 otherwise.

NzaeNumeric128Field **nz::ae::operator%(NzaeNumericField &lhs, NzaeNumericField &rhs)** Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField **lhs**
Value 1.

NzaeNumericField **rhs** Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field **nz::ae::operator%(double lhs, NzaeNumericField &rhs)** Perform a modulus operation using the two specified values.

Parameters

lhs
Value 1.

NzaeNumericField **rhs** Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(int64_t lhs, NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(int32_t lhs, NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(double lhs, const NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(int64_t lhs, const NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(int32_t lhs, const NzaeNumericField &rhs) Perform a modulus operation using the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, double rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, double rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, int64_t rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, int64_t rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(const NzaeNumericField &lhs, int32_t rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator%(NzaeNumericField &lhs, int32_t rhs) Perform a modulus operation using the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs modulo rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(int32_t lhs, NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, NzaeNumericField &rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, const NzaeNumericField &rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, double rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, NzaeNumericField &rhs)

Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(double lhs, const NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, double rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, const NzaeNumericField &rhs)

Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(double lhs, NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, int64_t rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(int64_t lhs, const NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, int64_t rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(int64_t lhs, NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(const NzaeNumericField &lhs, int32_t rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(int32_t lhs, const NzaeNumericField &rhs) Multiply the two specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator*(NzaeNumericField &lhs, int32_t rhs) Multiply the two specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The product of lhs multiplied by rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeTimeField &time, const NzaeDateField &date)

Add date and time.

Parameters

NzaeTimeField time

The time.

NzaeDateField

date The date.

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, NzaeNumericField &rhs)

Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(int32_t lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(double lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, int64_t rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(int64_t lhs, const NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs) Unary plus.

Parameters

NzaeNumericField lhs

Value.

Returns

NzaeNumeric128Field

The new NzaeNumeric128Field object.

Exceptions

NzaeException

NzaeDateField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeDateField &date)

Add an interval and a date.

Parameters

NzaeIntervalField

iv The interval.

NzaeDateField

date The date.

Returns

NzaeDateField

The new date.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, double rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeDateField &date, const NzaeTimeField &time) Add a date and a time.

Parameters

NzaeDateField

date The date

NzaeTimeField

time The time

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, int64_t rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeDateField &date, const NzaeTimeTzField &time)

Add a date and a time.

Parameters

NzaeDateField

date The date

NzaeTimeTzField

time The time

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeTimeField nz::ae::operator+(const NzaeTimeField &time, const NzaeIntervalField &iv) Add an interval and a time.

Parameters

NzaeTimeField time

The time.

NzaeIntervalField

iv The interval.

Returns

NzaeTimeField

The time.

Exceptions

NzaeException

NzaeTimeTzField nz::ae::operator+(const NzaeTimeTzField &time, const NzaeIntervalField &iv)

Add an interval and a timetz.

Parameters

NzaeTimeTzField

time The timetz.

NzaeIntervalField

iv The interval.

Returns

NzaeTimeTzField

The time.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeTimestampField &time, const NzaeIntervalField &iv)

Add an interval and a timestamp.

Parameters

NzaeTimestampField

time The timestamp.

NzaeIntervalField

iv The interval.

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeDateField nz::ae::operator+(const NzaeDateField &date, const NzaeIntervalField &iv) Add an interval and a date.

Parameters

NzaeDateField

date The date.

NzaeIntervalField

iv The interval.

Returns

NzaeDateField

The new date.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(int64_t lhs, NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeTimeTzField &time, const NzaeDateField &date)

Add a date and a timetz.

Parameters

NzaeTimeTzField

time The timetz.

NzaeDateField

date The date.

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(double lhs, const NzaeNumericField &rhs)

Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimeField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimeField &time) Add an interval and a time.

Parameters

NzaeIntervalField

iv The interval.

NzaeTimeField

time The time.

Returns

NzaeTimeField

The time.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(const NzaeNumericField &lhs, int32_t rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimestampField &time)

Add an interval and a timestamp.

Parameters

NzaeIntervalField

iv The interval.

NzaeTimestampField

time The timestamp.

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeTimeTzField nz::ae::operator+(const NzaeIntervalField &iv, const NzaeTimeTzField &time)

Add an interval and a timetz.

Parameters

NzaeIntervalField

iv The interval.

NzaeTimeTzField

time The timetz.

Returns

NzaeTimeTzField

The time.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(int32_t lhs, const NzaeNumericField &rhs) Perform an addition operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, double rhs)

Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator+(NzaeNumericField &lhs, int32_t rhs) Perform an addition operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The sum of lhs + rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator++(NzaeNumericField &lhs, int rhs) Increments one value by one.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Dummy

Returns

NzaeNumeric128Field

The result of lhs incremented by one as a Numeric128.

Exceptions

NzaeException

NzaeTimeTzField nz::ae::operator-(const NzaeTimeTzField &time, const NzaeIntervalField &iv) Subtract an interval from timetz.

Parameters

NzaeTimeTzField

time The time.

NzaeIntervalField

iv The interval.

Returns

NzaeTimeTzField

The timetz.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(int32_t lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeIntervalField nz::ae::operator-(const NzaeTimeField &time, const NzaeTimeField &t2) Subtract time from time.

Parameters

NzaeTimeField

time Time 1.

NzaeTimeField t2

Time 2.

Returns

NzaeIntervalField

The interval between the specified time values.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, double rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, double rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, int64_t rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeDateField nz::ae::operator-(const NzaeDateField &date, const NzaeIntervalField &iv) Subtract an interval from a date.

Parameters

NzaeDateField

date The date.

NzaeIntervalField

iv The interval

Returns

NzaeDateField

The date.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, NzaeNumericField &rhs)

Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(int64_t lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, int64_t rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimestampField nz::ae::operator-(const NzaeTimestampField &time, const NzaeIntervalField &iv)

Subtract an interval from a timestamp.

Parameters

NzaeTimestampField

time The timestamp.

NzaeIntervalField

iv The interval.

Returns

NzaeTimestampField

The timestamp.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(double lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeTimeField nz::ae::operator-(const NzaeTimeField &time, const NzaeIntervalField &iv) Subtract an interval from a time.

Parameters

NzaeTimeField time

The time.

NzaeIntervalField

iv The interval.

Returns

NzaeTimeField

The time.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(double lhs, const NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(int64_t lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs, int32_t rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeIntervalField nz::ae::operator-(const NzaeTimestampField &time, const NzaeTimestampField &t2)

Subtract timestamp from timestamp.

Parameters

NzaeTimestampField

time Timestamp 1.

NzaeTimestampField

t2 Timestamp 2.

Returns

NzaeIntervalField

The interval between the specified timestamps.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(NzaeNumericField &lhs, int32_t rhs) Perform a subtraction operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(int32_t lhs, NzaeNumericField &rhs) Perform a subtraction operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField
rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs minus rhs as a Numeric128.

Exceptions

NzaeException

NzaeIntervalField nz::ae::operator-(const NzaeDateField &date, const NzaeDateField &d2) Subtract a date from a date.

Parameters

NzaeDateField
date Date 1.

NzaeDateField
d2 Date 2.

Returns

NzaeIntervalField

The interval between the specified dates.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator-(const NzaeNumericField &lhs) Unary minus.

Parameters

NzaeNumericField lhs
 Value.

Returns

NzaeNumeric128Field

The new NzaeNumeric128Field object.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator--(NzaeNumericField &lhs, int rhs) Decrements one value by one.

Parameters

NzaeNumericField lhs
 Value 1.

rhs
 Dummy

Returns

NzaeNumeric128Field

The result of lhs decremented by one as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, double rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(int64_t lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, double rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, int32_t rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(int64_t lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(double lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, int64_t rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(int32_t lhs, const NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(int32_t lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, int32_t rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(double lhs, NzaeNumericField &rhs) Perform a division operation using the specified values.

Parameters

lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(const NzaeNumericField &lhs, const NzaeNumericField &rhs)

Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

NzaeNumericField

rhs Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

NzaeNumeric128Field nz::ae::operator/(NzaeNumericField &lhs, int64_t rhs) Perform a division operation using the specified values.

Parameters

NzaeNumericField lhs

Value 1.

rhs

Value 2.

Returns

NzaeNumeric128Field

The result of lhs divided by rhs as a Numeric128.

Exceptions

NzaeException

CHAPTER 3

Class Documentation

NzaeAggregate Class Reference

This class provides Aggregate functionality and is used to implement Aggregation AEs.

Public Types

```
enum LogLevel {  
    LOG_TRACE=1, LOG_DEBUG=2  
} The Log Level.  
  
enum NzaeAggType {  
    NzaeAggUnknown, NzaeAggGrouped, NzaeAggAnalytic  
} Aggregate types.  
  
NzaeAggType
```

Public Member Functions

```
virtual void close()=0  
Closes the AE and releases its resources.  
  
virtual const NzaeEnvironment& getEnvironment() const  
=0 Gets environment information for the AE.  
  
virtual const NzaeLibrary& getLibrary() const  
=0 Gets library information for the AE.  
  
virtual NzaeAggregateMessageHandler& getMessageHandler() const  
=0 Returns the message handler class object.  
  
virtual const NzaeParameters& getParameters() const  
=0 Gets parameter information for the AE.  
  
virtual const NzaeRuntime& getRuntime() const =0
```

Gets runtime information for the AE, including information about the Netezza software.

virtual void log(LogLevel logLevel, const char *message) const
=0 Logs the specified message at the specified log level.

virtual std::string logFileName() const
=0 Returns the log file name.

virtual void ping() const =0
Indicates that the AE is still active and not hanging.

virtual void runAggregation(NzaeAggregateMessageHandler
*messageHandler)=0 Begins the Aggregation Message Processing.

virtual NzaeAggType type() const =0
Returns the type of the aggregate.

virtual void userError(const char *message) const =0
Indicates that the AE has encountered an error condition.

virtual ~NzaeAggregate()

Static Public Member Functions

static NzaeAggregate* newInstance(NzaeAggregateInitialization &arg,
NZAEGG_HANDLE handle)

Detailed Description

This class provides Aggregate functionality and is used to implement Aggregation AEs.

See Also

- NzaeAggregateMessageHandler
- NzaeFactory
- NzaeApi
- NzaeLibrary
- NzaeParameters
- NzaeEnvironment

Enumeration Type Documentation

enum LogLevel
The Log Level.

LOG_TRACE

LOG_DEBUG

enum NzaeAggType
Aggregate types.

NzaeAggUnknown

NzaeAggGrouped**NzaeAggAnalytic**

Typedef Documentation

```
typedef enum nz::ae::NzaeAggregate::NzaeAggType NzaeAggTypeNzaeAggType
```

Public Member Function Documentation

virtual void close()=0

Closes the AE and releases its resources.

Releases all resources associated with the aggregate.

virtual const NzaeEnvironment& getEnvironment() const**=0** Gets environment information for the AE.

Returns

NzaeEnvironment

The instance of NzaeEnvironment .

See Also

NzaeEnvironment

virtual const NzaeLibrary& getLibrary() const**=0** Gets library information for the AE.

Returns

NzaeLibrary

The instance of NzaeLibrary .

See Also

NzaeLibrary

virtual NzaeAggregateMessageHandler& getMessageHandler() const**=0** Returns the message handler class object.

Returns

NzaeAggregateMessageHandler

The instance of NzaeAggregateMessageHandler .

The message handler is where custom aggregate logic is implemented.

▲ See Also

► NzaeAggregateMessageHandler

virtual const NzaeParameters& getParameters() const**=0** Gets parameter information for the AE.

Returns

NzaeParameters

The instance of NzaeParameters .

See Also

NzaeParameters

virtual const NzaeRuntime& getRuntime() const =0

Gets runtime information for the AE, including information about the Netezza software.

Returns

NzaeRuntime

The instance of NzaeRuntime .

See Also

NzaeRunTime

virtual void log(LogLevel logLevel, const char *message) const

=0 Logs the specified message at the specified log level.

Parameters

LogLevel logLevel The
log level constant.

message
The message to log.

virtual std::string logFileName() const

=0 Returns the log file name.

Returns

The log file name.

virtual void ping() const =0

Indicates that the AE is still active and not hanging.

virtual void runAggregation(NzaeAggregateMessageHandler

***messageHandler)=0** Begins the Aggregation Message Processing.

Parameters

NzaeAggregateMessageHandler
messageHandler The message handler.

Runs the aggregate using the message handler. The message handler is where custom aggreg-ate logic is implemented.

See Also

NzaeAggregateMessageHandler

virtual NzaeAggType type() const =0

Returns the type of the aggregate.

Returns

NzaeAggType

The aggregate type.

virtual void userError(const char *message) const =0

Indicates that the AE has encountered an error condition.

Parameters

message

The message to send back to the Netezza software.

Implies NzaeDone.

virtual ~NzaeAggregate()

Static Public Member Function Documentation

static NzaeAggregate* newInstance(NzaeAggregateInitialization &arg, NZAEAGG_HANDLE handle)

Returns

NzaeAggregate

NzaeAggregateInitialization Class Reference

Not implemented. Placeholder reserved for future use.

Detailed Description

Not implemented. Placeholder reserved for future use.

See Also

NzaeFactory

NzaeAggregate

NzaeApi

NzaeAggregateMessageHandler Interface Reference

This class provides Aggregate functionality.

Public Member Functions

virtual void accumulate(NzaeAggregate &api, NzaeRecord &input, NzaeRecord &state)=0 Modifies the state based on input.

virtual void finalResult(NzaeAggregate &api, NzaeRecord &inputState, NzaeRecord &result)=0 Sets the final result based on the input state.

virtual void initializeState(NzaeAggregate &api, NzaeRecord &state)=0 Initializes the state.

virtual void merge(NzaeAggregate &api, NzaeRecord &inputState, NzaeRecord &state)=0 Merges the specified input state into state.

virtual ~NzaeAggregateMessageHandler()

Detailed Description

This class provides Aggregate functionality.

Implement this class to handle NzaeAggregation messages.

See Also

runAggregation
NzaeRecord

Public Member Function Documentation

virtual void accumulate(NzaeAggregate &api, NzaeRecord &input, NzaeRecord &state)=0 Modifies the state based on input.

Parameters

NzaeAggregate api
The aggregate object.

NzaeRecord input
The input record.

NzaeRecord state
The state record.

Accumulate into state from input.

See Also

NzaeAggregate
NzaeRecord

virtual void finalResult(NzaeAggregate &api, NzaeRecord &inputState, NzaeRecord &result)=0

Sets the final result based on the input state.

Parameters

NzaeAggregate api

The aggregate object.

NzaeRecord inputState

The input state record.

NzaeRecord result

The result record.

Provides the result from inputState. The final result record may contain only one field.

See Also

NzaeAggregate

NzaeRecord

virtual void initializeState(NzaeAggregate &api, NzaeRecord &state)=0 Initializes the state.

Parameters

NzaeAggregate api

The aggregate object.

NzaeRecord state

The state record.

Initializes the state object before processing.

See Also

NzaeAggregate

NzaeRecord

virtual void merge(NzaeAggregate &api, NzaeRecord &inputState, NzaeRecord &state)=0 Merges the specified input state into state.

Parameters

NzaeAggregate api

The aggregate object.

NzaeRecord inputState

The input state record.

NzaeRecord state

The state record.

Merge from inputState into state.

See Also

NzaeAggregate

NzaeRecord

virtual ~NzaeAggregateMessageHandler()

NzaeApi Class Reference

This class holds API objects.

Public Types

```
enum ApiType {  
    UNKNOWN=0, FUNCTION, AGGREGATION, SHAPER,  
    ANY } The API type.
```

Public Member Functions

```
NzaeApi()  
Constructor that creates the appropriate API object.  
  
~NzaeApi()  
Destructor that deletes the appropriate API object.
```

Public Attributes

```
aeAggregate  
Aggregate object.  
  
aeFunction  
Function object.  
  
aeShaper  
Shaper object.  
  
apiType  
The API type.
```

Detailed Description

This class holds API objects.

See Also
 NzaeFunction
 NzaeAggregate
 NzaeShaper

Enumeration Type Documentation

```
enum ApiType  
The API type.
```

UNKNOWN

FUNCTION

AGGREGATION

SHAPER ANY

Public Member Function Documentation

NzaeApi()

Constructor that creates the appropriate API object.

~NzaeApi()

Destructor that deletes the appropriate API object.

Member Data Documentation

NzaeAggregate* aeAggregate

Aggregate object.

See Also

NzaeAggregate

NzaeFunction* aeFunction

Function object.

See Also

NzaeFunction

NzaeShaper* aeShaper

Shaper object.

See Also

NzaeShaper

ApiType apiType

The API type.

NzaeApiGenerator Class Reference

Helper class for getting an API object.

Public Member Functions

NzaeApi& getApi(nz::ae::NzaeApi::ApiType type) Gets an API object.

NzaeApi* getApi(nz::ae::NzaeApi::ApiType type, bool fork) Gets an API object.

NzaeRemoteProtocolCallback* getCallbackHandler()
Gets the remote protocol callback handler.

`bool isLocal()`
Return true if this is a local AE process.

`bool isRemote()`
Return true if this is a remote AE process.

`NzaeApiGenerator()`
Constructor.

`bool ownsAPI()`
Returns TRUE if the helper owns the API.

`void setCallbackHandler(NzaeRemoteProtocolCallback *handler)` Sets the remote protocol callback handler.

`virtual void setDataSliceId(int dataSliceId)` Sets the remote connection point dataslice ID.

`virtual void setName(const char *name)`
Sets the remote connection point name.

`void setOwnsAPI(bool owns)`
Sets whether this object should manage API.

`virtual void setSessionId(int sessionId)`
Sets the remote connection point session ID.

`virtual void setTransactionId(int64_t transactionId)`
Sets the remote connection transaction ID.

`~NzaeApiGenerator()`
Destructor.

Detailed Description

Helper class for getting an API object.

This class is used to hide much of the complexity of getting an API object for both local and remote mode AEs. In the API program flow, getting an API object is the first step.

See Also

- NzaeApi
- NzaeFactory
- NzaeConnectionPoint
- NzaeRemoteProtocol
- NzaeRemoteProtocolCallback

Public Member Function Documentation

`nz::ae::NzaeApi& getApi(nz::ae::NzaeApi::ApiType type)` Gets an API object.

▲ Parameters

ApiType type

Specified API type or ANY.

Returns

NzaeApi

The API object.

Exceptions

NzaeException

Returns an API object in either local or remote modes. Returns one of the specified type, or throws an exception. The API is owned by the helper object.

The API object is the main object for an AE program.

See Also

NzaeApi

nz::ae::NzaeApi* getApi(nz::ae::NzaeApi::ApiType type, bool fork) Gets an API object.

Parameters

ApiType type

Specified API type or ANY.

fork

Forks new process to handle if TRUE.

Returns

NzaeApi

API object is NULL in parent if fork is TRUE and the AE is a remote AE.

Exceptions

NzaeException

Returns an API in either local or remote modes. Returns one of the specified types or throws an exception. The API may be owned by the helper or the caller, depending on the setting for ownsAPI .

The API object is the main object for an AE program.

See Also

NzaeApi

ownsAPI

NzaeRemoteProtocolCallback* getCallbackHandler()

Gets the remote protocol callback handler.

Returns

NzaeRemoteProtocolCallback

The callback handler.

A remote protocol handler class is used to handle remote commands such as stop, status, and ping.

See Also

► NzaeRemoteProtocolCallback

bool isLocal()

Return true if this is a local AE process.

Returns

True if the AE is local

bool isRemote()

Return true if this is a remote AE process.

Returns

TRUE if the AE is remote.

NzaeApiGenerator()

Constructor.

bool ownsAPI()

Returns TRUE if the helper owns the API.

Returns

TRUE if the helper owns the API.

If TRUE, the API is deleted when a new one is accepted or the helper is deleted.

void setCallbackHandler(NzaeRemoteProtocolCallback *handler) Sets the remote protocol callback handler.

Parameters

NzaeRemoteProtocolCallback

handler The remote protocol handler.

A remote protocol handler class is used to handle remote commands such as stop, status and ping.

See Also

NzaeRemoteProtocolCallback

virtual void setDataSliceId(int dataSliceId)

Sets the remote connection point dataslice ID.

Parameters

dataSliceId

The dataslice ID of the remote connection point.

This function does not override the remote values from the launcher available in NzaeConnectionPoint class.

See Also

► NzaeConnectionPoint

virtual void setName(const char *name)

Sets the remote connection point name.

Parameters

name

The remote connection point name.

This function does not override the remote values from the launcher available in the NzaeConnection-Point class.

See Also

NzaeConnectionPoint

void setOwnsAPI(bool owns)

Sets whether this object should manage API.

Parameters

owns

TRUE if the helper owns the API.

If TRUE, the API is deleted when a new one is accepted or the helper is deleted.

virtual void setSessionId(int sessionId) Sets the remote connection point session ID.

Parameters

sessionId

The remote connection point session ID.

This function does not override the remote values from the launcher available in NzaeConnectionPoint class.

See Also

NzaeConnectionPoint

virtual void setTransactionId(int64_t transactionId)

Sets the remote connection transaction ID.

Parameters

transactionId

The remote connection point transaction ID.

This function does not override the remote values from the launcher available in NzaeConnectionPoint class.

▲ See Also

► NzaeConnectionPoint

~NzaeApiGenerator()

Destructor.

Deletes the API object if it is owned. Deletes the connection point and remote protocol objects.

NzaeBoolField Class Reference

This class provides field access for type bool.

Inherits NzaeField

Public Member Functions

`void fromString(std::string str)`

Constructs the field from the string.

`NzaeBoolField()`

Constructs a NULL bool field.

`NzaeBoolField(NzaeBoolField &field)`

Constructs a bool field with value field.

`NzaeBoolField(bool val)`

Constructs a bool field with value val.

`operator bool()`

Returns bool field value.

`NzaeBoolField& operator=(NzaeBoolField &field)`

Assigns the value of the argument to the field object.

`NzaeBoolField& operator=(NzaeField &field)`

Assigns the value of the argument to the field object.

`NzaeBoolField& operator=(bool val)`

Assigns the value of the argument to the field object.

`std::string toString() const`

Returns the string representation of the field.

`virtual NzaeDataTypes::Types type()`

`const` Returns the type of the field.

Detailed Description

This class provides field access for type bool.

See Also

▲ NzaeField

Public Member Function Documentation

`void fromString(std::string str)`

Constructs the field from the string. ▲ Parameters

str

The string to assign from.

NzaeBoolField()

Constructs a NULL bool field.

NzaeBoolField(NzaeBoolField &field)

Constructs a bool field with value field.

Parameters

NzaeBoolField field

The NzaeBoolField value.

NzaeBoolField(bool val)

Constructs a bool field with value val.

Parameters

val

The boolean value.

operator bool() Returns

bool field value.

Returns

The boolean value.

NzaeBoolField& operator=(NzaeBoolField &field)

Assigns the value of the argument to the field object.

Parameters

NzaeBoolField field

The field to assign.

Returns

NzaeBoolField**NzaeBoolField& operator=(NzaeField &field)** Assigns the value of the argument to the field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeBoolField

The field argument may be a different type, as long as it is compatible.

NzaeBoolField& operator=(bool val)

Assigns the value of the argument to the field object.

Parameters

val

The value to assign.

Returns

NzaeBoolField

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeCallbackResult Struct Reference

Struct used to specify the callback result.

Public Attributes

bFreeData

data

dataLength

returnCode

Detailed Description

Struct used to specify the callback result.

Member Data Documentation

int bFreeData

Must be set to TRUE if data has been allocated via malloc.

char* data

Data. Must be allocated via malloc.

int dataLength
Data length. May be 0.

int returnCode
Return Code. A 0 value is normal.

NzaeConnectionPoint Class Reference

Class to encapsulate the connection point for remote mode AEs.

Public Member Functions

virtual std::string buildFileName()=0
Gets the connection point file name.

virtual void close()=0
Releases connection point resources.

virtual int getDataSliceId()=0
Gets the connection point dataslice ID.

virtual NZAECONPT_HANDLE getHandle()=0

virtual std::string getName()=0
Gets the connection Ppoint name.

virtual int getRemoteDataSliceId()=0
Gets the remote dataslice ID used in the launcher.

virtual std::string getRemoteName()=0 Gets
the remote name used in the launcher.

virtual int getRemoteSessionId()=0
Gets the remote session ID used in the launcher.

virtual int64_t getRemoteTransactionId()=0
Gets the remote transaction ID used in the launcher.

virtual int getSessionId()=0
Gets the connection point session ID.

virtual int64_t getTransactionId()=0
Gets the connection point transaction ID.

virtual void setDataSliceId(int dataSliceId)=0
Sets the connection point dataslice ID.

virtual void setName(const char *name)=0
Sets the connection point name.

virtual void setSessionId(int sessionId)=0

Sets the connection point session ID.

```
virtual void setTransactionId(int64_t transactionId)=0
```

Sets the connection point transaction ID.

```
virtual ~NzaeConnectionPoint()
```

Static Public Member Functions

```
static NzaeConnectionPoint* newInstance()
```

Detailed Description

Class to encapsulate the connection point for remote mode AEs.

This class is used to specify the connection point parameters such as name, transaction ID, data-slice ID and session ID used to construct a unique connection point name.

A remote AE listens on a connection point and accepts remote AE data connections.

Users may prefer to use the simpler NzaeApiGenerator object.

See Also

NzaeApiGenerator

NzaeFactory

Public Member Function Documentation

```
virtual std::string buildFileName()=0
```

Gets the connection point file name.

Returns

The connection point file name.

The value is constructed from the connection point parameters.

```
virtual void close()=0
```

Releases connection point resources.

Release all resources associated with the connection point.

```
virtual int getDataSliceId()=0
```

Gets the connection point dataslice ID.

Returns

The connection point dataslice ID.

```
virtual NZAECONPT_HANDLE getHandle()=0
```

```
virtual std::string getName()=0
```

Gets the connection Ppoint name.

Returns
The connection point name.

virtual int getRemoteDataSlicId()=0

Gets the remote dataslice ID used in the launcher.

Returns
The remote dataslice ID or -1 if not set.

virtual std::string getRemoteName()=0 Gets

the remote name used in the launcher.

Returns
The remote name or an empty string if not set.

virtual int getRemoteSessionId()=0

Gets the remote session ID used in the launcher.

Returns
The remote session ID or -1 if not set.

virtual int64_t getRemoteTransactionId()=0

Gets the remote transaction ID used in the launcher.

Returns
The remote transaction ID or -1 if not set.

virtual int getSessionId()=0

Gets the connection point session ID.

Returns
The connection point session ID.

virtual int64_t getTransactionId()=0 Gets

the connection point transaction ID.

Returns
The connection point transaction ID.

virtual void setDataSlicId(int dataSlicId)=0

Sets the connection point dataslice ID.

Parameters
dataSlicId
The connection point dataslice ID.

Determines if the connection point uses the dataslice ID.

virtual void setName(const char *name)=0

Sets the connection point name.

Parameters

name

The connection point name.

A connection point name is the only required parameter for a connection point.

virtual void setSessionId(int sessionId)=0

Sets the connection point session ID.

Parameters

sessionId

The connection point session ID.

Determines if the connection point uses the session ID.

virtual void setTransactionId(int64_t transactionId)=0

Sets the connection point transaction ID.

Parameters

transactionId

The connection point transaction ID.

Determines if the connection point uses the transaction ID.

virtual ~NzaeConnectionPoint()

Static Public Member Function Documentation

static NzaeConnectionPoint* newInstance()

Returns

NzaeConnectionPoint

NzaeDataTypes Class Reference

This class provides the data type enums.

Public Types

enum Types {

```

NZUDSUDX_UNKNOWN= -1, NZUDSUDX_FIXED= 0, NZUDSUDX_VARIABLE= 1,
NZUDSUDX_NATIONAL_FIXED= 2, NZUDSUDX_NATIONAL_VARIABLE= 3, NZUDSUDX_BOOL= 4, NZUD-
SUDX_DATE= 5, NZUDSUDX_TIME= 6, NZUDSUDX_TIMETZ= 7, NZUDSUDX_NUMERIC32= 8,
NZUDSUDX_NUMERIC64= 9, NZUDSUDX_NUMERIC128= 10, NZUDSUDX_FLOAT= 11,
NZUDSUDX_DOUBLE= 12, NZUDSUDX_INTERVAL= 13, NZUDSUDX_INT8= 14, NZUDSUDX_INT16= 15,
NZUDSUDX_INT32= 16, NZUDSUDX_INT64= 17, NZUDSUDX_TIMESTAMP= 18, NZUDSUDX_GEOMETRY=
19, NZUDSUDX_VARBINARY= 20, NZUDSUDX_MAX_TYPE= 21 }

```

Data types that match the Netezza system types.

Detailed Description

This class provides the data type enums.

Enumeration Type Documentation

enum Types

Data types that match the Netezza system types.

NZUDSUDX_UNKNOWN Unknown data type

NZUDSUDX_FIXED Fixed string **NZUDSUDX_VARIABLE**

Variable string **NZUDSUDX_NATIONAL_FIXED** Fixed

national string **NZUDSUDX_NATIONAL_VARIABLE**

Variable national string **NZUDSUDX_BOOL** Boolean

NZUDSUDX_DATE Date

NZUDSUDX_TIME Time

NZUDSUDX_TIMETZ Time zone

NZUDSUDX_NUMERIC32 Numeric 32

NZUDSUDX_NUMERIC64 Numeric 64

NZUDSUDX_NUMERIC128 Numeric 128

NZUDSUDX_FLOAT Float

NZUDSUDX_DOUBLE Double

NZUDSUDX_INTERVAL Interval

NZUDSUDX_INT8 1 byte integer

NZUDSUDX_INT16 2 byte integer

NZUDSUDX_INT32 4 byte integer

NZUDSUDX_INT64 8 byte integer

NZUDSUDX_TIMESTAMP Time stamp

NZUDSUDX_GEOMETRY Geometry

NZUDSUDX_VARBINARY Variable Binary

NZUDSUDX_MAX_TYPE Greater than any data type enum value

NzaeDateField Class Reference

This class provides field access for type date.

Inherits NzaeField

Public Member Functions

NzaeTimestampField addTime(const NzaeTimeField &time)
const Constructs a TimestampField by adding time.

NzaeTimestampField addTimeTz(const NzaeTimeTzField &time)
const Constructs a TimestampField by adding timetz.

NzaeIntervalField age(const NzaeDateField &x) const
Constructs an IntervalField by subtracting dates.

void decodeDate(uint8_t *month, uint8_t *day, uint16_t *year, bool *errorFlag=NULL)
const Converts a Netezza-encoded Date value to m/d/y.

void decodeDate(time_t *result, bool *errorFlag=NULL) const
Converts a Netezza-encoded Date value to time_t and treats encoded date as if it is UTC.
The resulting time_t represents the time 00:00:00 on the specified date.

void decodeDate(struct tm *result, bool *errorFlag=NULL) const
Converts a Netezza-encoded Date value to struct tm. The resulting tm represents the
time 00:00:00 on the specified date, with an unknown daylight saving time status.

void encodeDate(uint32_t month, uint32_t day, uint32_t year, bool
*errorFlag=NULL) Converts a m/d/y Date value to a Netezza-encoded Date.

void encodeDate(time_t date, bool *errorFlag=NULL)
Converts a time_t Date value to a Netezza-encoded Date. Drops the hours, minutes
and seconds elapsed after the last whole day in the time_t value.

void encodeDate(const struct tm &date, bool *errorFlag=NULL)
Converts a struct tm value to a Netezza-encoded Date. Uses only the tm.tm_year,
tm.tm_mon and tm.tm_day fields of the date, ignoring the other fields. It is recommended
that the date passes isValidTimeStruct(), but it is not required.

void fromString(std::string str)
Constructs the field from the string.

bool isValidDate() const
Specifies whether a Netezza-encoded Date value is valid and within the Netezza Date range.

bool isValidEpochDate() const
Specifies whether a Netezza-encoded Date value is valid and within the time_t Epoch range.

NzaeDateField()

Constructs a NULL date field.

NzaeDateField(const NzaeDateField &field)

Constructs a date field with value field.

NzaeDateField(const NzaeTimestampField
&field) Constructs a date field with value field.

NzaeDateField(int32_t val) Constructs
a date field with value val.

operator int32_t() const

Returns an encoded field value.

operator NzaeTimestampField() const

Returns a timestamp field value.

NzaeDateField& operator=(int32_t val)

Assigns the value of the argument to a field object.

NzaeDateField& operator=(const NzaeTimestampField
&field) Assigns the value of the argument to a field object.

NzaeDateField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

NzaeDateField& operator=(const NzaeDateField &field)

Assigns the value of the argument to a field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Static Public Member Functions

static int32_t epochEnd()

Gets the encoded epoch end.

static int32_t epochStart()

Gets the encoded epoch start.

static uint32_t getYearDay(uint32_t month, uint32_t day, uint32_t year, bool
*errorFlag=NULL) Given a m/d/y format date, returns the day number of the year.

static bool isValidDate(uint32_t month, uint32_t day, uint32_t year)

Specifies whether a decoded m/d/y Date value is valid and within the Netezza Date range.

static int32_t max()

Gets the encoded max.

static int32_t min()

Gets the encoded min.

static uint8_t numDaysInMonth(uint32_t month, uint32_t year, bool

*errorFlag=NULL) Determine the total number of days in a given month.

static int16_t yearMax()

Gets the decoded year max.

```
static int16_t yearMin()
```

Gets the decoded year min.

Detailed Description

This class provides field access for type date.

See Also

▲ NzaeField

Public Member Function Documentation

NzaeTimestampField addTime(const NzaeTimeField &time)

const Constructs a TimestampField by adding time.

Parameters

NzaeTimeField time The
NzaeTimeField value.

Returns

NzaeTimestampField

The timestamp consisting of date plus time.

See Also

NzaeTimeField
NzaeTimestampField

NzaeTimestampField addTimeTz(const NzaeTimeTzField &time)

const Constructs a TimestampField by adding timetz.

Parameters

NzaeTimeTzField time The
NzaeTimeTzField value.

Returns

NzaeTimestampField

The timestamp consisting of date plus timetz.

See Also

NzaeTimeTzField
NzaeTimestampField

NzaeIntervalField age(const NzaeDateField &x) const

Constructs an IntervalField by subtracting dates.

Parameters

NzaeDateField x

The NzaeDateField value.

Returns

NzaeIntervalField

IntervalField consisting of date minus date.

See Also

NzaeIntervalField

void decodeDate(uint8_t *month, uint8_t *day, uint16_t *year, bool *errorFlag=NULL)

const Converts a Netezza-encoded Date value to m/d/y.

Parameters

day

The day count, 1 to 31 inclusive.

month

The month number, 1 to 12 inclusive.

year

The year number, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

errorFlag

If not NULL, *set to TRUE if isValidDate(encodedDate) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void decodeDate(time_t *result, bool *errorFlag=NULL) const

Converts a Netezza-encoded Date value to time_t and treats encoded date as if it is UTC. The resulting time_t represents the time 00:00:00 on the specified date.

Parameters

result

The time_t date representation. Forced to be signed int32.

errorFlag

If not NULL, *set to TRUE if isValidEpochDate(encodedDate) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void decodeDate(struct tm *result, bool *errorFlag=NULL) const

Converts a Netezza-encoded Date value to struct tm. The resulting tm represents the time 00:00:00 on the specified date, with an unknown daylight saving time status.

Parameters

result

The structure where the decoded Date is written, such that result->tm_year, result->tm_mon, result->tm_mday, result->tm_yday and result->tm_wday contain the appropriate fields in tm format. result->tm_isdst is set to -1. When applicable, all the other fields of result are set to 0.

errorFlag

If not NULL, *set to TRUE if isValidDate(encodedDate) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeDate(uint32_t month, uint32_t day, uint32_t year, bool *errorFlag=NULL) Converts a m/d/y Date value to a Netezza-encoded Date.

Parameters

day

The day count, 1 to 31 inclusive.

month

The month number, 1 to 12 inclusive.

year

The year number, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

errorFlag

If not NULL, *set to TRUE if isValidDate(month,day,year) is FALSE; *set to FALSE other-wise.

Exceptions

NzaeException

void encodeDate(time_t date, bool *errorFlag=NULL)

Converts a time_t Date value to a Netezza-encoded Date. Drops the hours, minutes and seconds elapsed after the last whole day in the time_t value.

Parameters

date

The time_t date value.

errorFlag

If not NULL, *set to TRUE if isValidEpoch(date) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeDate(const struct tm &date, bool *errorFlag=NULL)

Converts a struct tm value to a Netezza-encoded Date. Uses only the tm.tm_year, tm.tm_mon and tm.tm_mday fields of the date, ignoring the other fields. It is recommended that the date passes isValidTimeStruct(), but it is not required.

Parameters

date

The struct tm date value.

errorFlag

If not NULL, *set to TRUE if date.tm_mon<0 or date.tm_mday<1 or date.tm_year+1900<SQL_YEAR_MIN or isValidDate(date.tm_mon+1, date.tm_mday, date.tm_year) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

bool isValidDate() const

Specifies whether a Netezza-encoded Date value is valid and within the Netezza Date range.

Returns

FALSE if encoded date<ENC_DATE_MIN or encoded date>ENC_DATE_MAX. TRUE otherwise.

bool isValidEpochDate() const

Specifies whether a Netezza-encoded Date value is valid and within the time_t Epoch range.

Returns

FALSE if encoded date< EPOCH_START_AS_DATE or encoded date> EPOCH_END_AS_DATE.

TRUE otherwise.

NzaeDateField()

Constructs a NULL date field.

NzaeDateField(const NzaeDateField &field)

Constructs a date field with value field.

Parameters

NzaeDateField field

The NzaeDateField value.

NzaeDateField(const NzaeTimestampField &field) Constructs a date field with value field.

Parameters

NzaeTimestampField field The NzaeTimestampField value.

NzaeDateField(int32_t val)

Constructs a date field with value val.

Parameters

val

The encoded date value.

operator int32_t() const

Returns an encoded field value.

Returns
The encoded value.

operator NzaeTimestampField() const

Returns a timestamp field value.

Returns
The timestamp value converted from date.

See Also
NzaeTimestampField

NzaeDateField& operator=(int32_t val)

Assigns the value of the argument to a field object.

Parameters
val
The encoded value to assign.

Returns
NzaeDateField

NzaeDateField& operator=(const NzaeTimestampField &field) Assigns the value of the argument to a field object.

Parameters
NzaeTimestampField
field The field to assign.

Returns
NzaeDateField

See Also
NzaeTimestampField

NzaeDateField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters
NzaeField field
The field to assign.

Returns
NzaeDateField

The field argument may be a different type, as long as it is compatible.

NzaeDateField& operator=(const NzaeDateField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeDateField field

The field to assign.

Returns

NzaeDateField

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

Static Public Member Function Documentation

static int32_t epochEnd()

Gets the encoded epoch end.

Returns

The encoded epoch end.

static int32_t epochStart()

Gets the encoded epoch start.

Returns

The encoded epoch start.

static uint32_t getYearDay(uint32_t month, uint32_t day, uint32_t year, bool *errorFlag=NULL) Given a m/d/y format date, returns the day number of the year.

Parameters

month

The month number, 1 to 12 inclusive.

year

The year of the date, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

day

The day of month, 1 to 31 inclusive.

errorFlag

Optional. If not NULL, set to TRUE if isValidDate(month,day,year) is FALSE. set to FALSE otherwise.

Returns

Day value [0,364] for non-leap years and [0,365] for leap years, 0 if isValidDate(month,day,year) is FALSE and errorFlag is not NULL.

Exceptions

NzaeException

static bool isValidDate(uint32_t month, uint32_t day, uint32_t year)

Specifies whether a decoded m/d/y Date value is valid and within the Netezza Date range.

Parameters

month

The month, 1 to 12 inclusive.

day

The day, 1 to 31 inclusive.

year

The year of the date, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

Returns

FALSE if (month>12 or month<1) or (day<1 or day>31) or (year<SQL_YEAR_MIN or year>SQL_YEAR_MAX) or (month is in (4, 6, 9, 11) and day>30) or (isLeapYear(year) and month=2 and day>29) or (!isLeapYear(year) and month=2 and day>28). TRUE otherwise.

static int32_t max()

Gets the encoded max.

Returns

The encoded max.

static int32_t min()

Gets the encoded min.

Returns

The encoded min.

static uint8_t numDaysInMonth(uint32_t month, uint32_t year, bool *errorFlag=NULL) Determine the total number of days in a given month.

Parameters

month

The month number, 1 to 12 inclusive.

year

The year number, 1 to 9999. Used to determine the correct number of days if month is Febru-ary.

errorFlag

Optional. If not NULL, set to TRUE if isValidSqlMonth(month) is FALSE or isValidSqlYear(year) is FALSE; set to FALSE otherwise.

Returns

30 if month is (4, 6, 9, 11), 31 if month is (1, 3, 5, 7, 8, 10, 12), 28 if month is 2 and isLeapYear(year), 29 if month is 2 and !isLeapYear(year), 0 if errorFlag is not NULL, and (isValidYearNumber(year) is FALSE or isValidMonthNumber(month) is FALSE)

Exceptions

NzaeException

static int16_t yearMax()

Gets the decoded year max.

Returns

The decoded year max.

static int16_t yearMin()

Gets the decoded year min.

Returns

The decoded year min.

Helpers that return information about the possible legal value ranges for decoded information.

NzaeDoubleField Class Reference

This class provides field access for type double.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

NzaeDoubleField()

Constructs a NULL double field.

NzaeDoubleField(NzaeDoubleField &field)

Constructs a double field with value field.

NzaeDoubleField(double val)

Constructs a double field with value val.

operator double()

Returns the double field value.

NzaeDoubleField& operator=(NzaeDoubleField &field)

Assigns the value of the argument to a field object.

NzaeDoubleField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

NzaeDoubleField& operator=(double val)

Assigns the value of the argument to a field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Detailed Description

This class provides field access for type double.

See Also

▲ [NzaeField](#)

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

NzaeDoubleField()

Constructs a NULL double field.

NzaeDoubleField(NzaeDoubleField &field)

Constructs a double field with value field.

Parameters

NzaeDoubleField field The

NzaeDoubleField value.

NzaeDoubleField(double val) Constructs

a double field with value val.

Parameters

val

The double value.

operator double()

Returns the double field value.

Returns
The double value.

NzaeDoubleField& operator=(NzaeDoubleField &field)

Assigns the value of the argument to a field object.

Parameters
NzaeDoubleField field
The field to assign.

Returns
NzaeDoubleField

NzaeDoubleField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters
NzaeField field
The field to assign.

Returns
NzaeDoubleField

The field argument may be a different type, as long as it is compatible.

NzaeDoubleField& operator=(double val) Assigns
the value of the argument to a field object.

Parameters
val
The value to assign.

Returns
NzaeDoubleField

std::string toString() const

Returns the string representation of the field.

Returns
The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns
Types
The field type.

NzaeEnvironment Class Reference

This class provides the AE Environment and lookup access to the AE environment.

Public Member Functions

```
virtual void addEntry(std::string name, std::string value)=0
virtual const char* getFirstKey() const =0
Returns the first key in the environment.

virtual const char* getNextKey() const =0
Returns the next key in the environment.

virtual const char* getValue(std::string name) const
=0 Returns the value for the key in the environment.

virtual bool hasKey(std::string name) const =0 Returns
TRUE if the key is defined in the environment.

virtual void setReadOnly()=0
virtual int size() const =0
Returns the number of entries in the environment.

virtual ~NzaeEnvironment()
```

Static Public Member Functions

```
static NzaeEnvironment* create()
```

Detailed Description

This class provides the AE Environment and lookup access to the AE environment.

See Also

▲ [NzaeException](#)

Public Member Function Documentation

```
virtual void addEntry(std::string name, std::string value)=0
```

```
virtual const char* getFirstKey() const =0
```

Returns the first key in the environment.

Returns

The key or NULL if none.

```
virtual const char* getNextKey() const =0
```

Returns the next key in the environment.

Returns
The key or NULL if none.

virtual const char* getValue(std::string name) const =0 Returns the value for the key in the environment.

Parameters
name
The environment name.

Returns
The value.

Exceptions
NzaeException

virtual bool hasKey(std::string name) const =0 Returns TRUE if the key is defined in the environment.

Parameters
name
The environment name.

Returns
TRUE if defined.

virtual void setReadOnly()=0

virtual int size() const =0
Returns the number of entries in the environment.

Returns
The size.

virtual ~NzaeEnvironment()

Static Public Member Function Documentation

static NzaeEnvironment* create()
Returns
NzaeEnvironment

NzaeException Class Reference

This class is used for all C++ AE Exceptions.

Public Member Functions

NzaeException(const std::string &what)

Creates an exception with error text.

virtual ~NzaeException()

Static Public Member Functions

static std::string format(const std::string &msg,...)

Format a string using printf style formatting.

Detailed Description

This class is used for all C++ AE Exceptions.

Public Member Function Documentation

NzaeException(const std::string &what)

Creates an exception with error text.

Parameters

what

The error text.

virtual ~NzaeException()

Static Public Member Function Documentation

static std::string format(const std::string &msg,...)

Format a string using printf style formatting.

Parameters

msg

The format string.

Returns

The formatted string.

NzaeFactory Class Reference

This class is used to get an API object.

Public Member Functions

virtual NzaeRemoteProtocol* createListener(NzaeConnectionPoint &connectionPoint)

Creates a new listener for remote AE connections.

virtual NzaeAggregate* getLocalAggregationApi(NzaeAggregateInitialization &arg)=0 Creates and returns the local instance of the Aggregation object.

virtual NzaeApi* getLocalApi()

Return the local API object.

virtual NzaeFunction* getLocalFunctionApi(NzaeFunctionInitialization &arg)=0 Creates and returns the local instance of the Function object.

virtual NzaeShaper* getLocalShaperApi(NzaeShaperInitialization &arg)=0 Creates and returns the local instance of the Shaper object.

virtual bool isLocal()

Returns TRUE if the process is a local AE.

virtual bool isRemote()

Returns true if this is a remote AE process.

virtual NzaeConnectionPoint* newConnectionPoint()

Returns a new instance of a connection point object.

virtual ~NzaeFactory()

Static Public Member Functions

static NzaeFactory& getFactory()

Returns the singleton Factory.

static pid_t getParentProcessId()

The parent ID of this process, which can be useful for debugging.

static pid_t getProcessId()

The ID of this process, which can be useful for debugging.

Detailed Description

This class is used to get an API object.

This class can be used to for both local and remote modes. In local mode, it can be used to get an API object, or function, aggregation and shaper objects. In remote mode, it can be used to create a connection point and a listener, which can then be used to get the API or other objects in remote mode.

Users may prefer to use the NzaeApiGenerator object, which may be easier to use.

See Also

NzaeApiGenerator

NzaeApi

NzaeFunction

NzaeAggregate

NzaeShaper

NzaeConnectionPoint

Public Member Function Documentation

virtual NzaeRemoteProtocol* createListener(NzaeConnectionPoint &connectionPoint)

Creates a new listener for remote AE connections.

Parameters

NzaeConnectionPoint connectionPoint

The connection point object.

Returns

NzaeRemoteProtocol

A Remote Protocol object.

Exceptions

NzaeException

A Listener is used for a remote AE. One listener per unique connection name may be created.

An AE may have multiple listeners.

This object must be deleted when complete.

See Also

NzaeRemoteProtocol

NzaeConnectionPoint

virtual NzaeAggregate* getLocalAggregationApi(NzaeAggregateInitialization &arg)=0 Creates and returns the local instance of the Aggregation object.

Parameters

NzaeAggregateInitialization arg

An aggregate initialization object.

Returns

NzaeAggregate

An Aggregate API object.

Exceptions

NzaeException

This object must be deleted when complete.

See Also

NzaeAggregate

NzaeAggregateInitialization

virtual NzaeApi* getLocalApi()

Return the local API object.

Returns

NzaeApi

An API object.

Determined by how the AE was launched (UDF,UDTF = function, or UDA = Aggregation, or function shaper and sizer) This method is only valid for local AEs. This object must be deleted when complete.

See Also
 NzaeApi

virtual NzaeFunction* getLocalFunctionApi(NzaeFunctionInitialization &arg)=0 Creates and returns the local instance of the Function object.

Parameters
NzaeFunctionInitialization arg
 A Function initialization object.

Returns
NzaeFunction
 A Function API object.

Exceptions
 NzaeException
 This object must be deleted when complete.

See Also
 NzaeFunction
 NzaeFunctionInitialization

virtual NzaeShaper* getLocalShaperApi(NzaeShaperInitialization &arg)=0 Creates and returns the local instance of the Shaper object.

Parameters
NzaeShaperInitialization arg
 A Shaper initialization object.

Returns
NzaeShaper
 A Shaper API object.

Exceptions
 NzaeException
 This object must be deleted when complete.

See Also
 NzaeShaper
 NzaeShaperInitialization

virtual bool isLocal()
 Returns TRUE if the process is a local AE.

Returns
 TRUE if the AE is local.

virtual bool isRemote()
 Returns true if this is a remote AE process.

Returns

True if remote AE

virtual NzaeConnectionPoint* newConnectionPoint()

Returns a new instance of a connection point object.

Returns

NzaeConnectionPoint

Connection point object.

Exceptions

NzaeException

A connection point object is used for a remote AE. The object must be deleted when complete.

▲ See Also

► NzaeConnectionPoint

virtual ~NzaeFactory()

Static Public Member Function Documentation

static NzaeFactory& getFactory()

Returns the singleton Factory.

Returns

NzaeFactory

The singleton Factory.

static pid_t getParentProcessId()

The parent ID of this process, which can be useful for debugging.

Returns

The parent ID of this process.

static pid_t getProcessId()

The ID of this process, which can be useful for debugging.

Returns

Process ID.

NzaeField Interface Reference

Provides the field interface.

Public Member Functions

`void assign(NzaeField &field)`

Assigns the value of the argument to the field object.

`virtual void fromString(std::string str)=0`

Constructs the field from the string.

`bool isNull() const`

Determines whether the field is NULL.

`NzaeField()`

Constructs a NULL field.

`NzaeField& operator=(NzaeField &field)`

Assigns the value of the argument to the field object.

`void setNull(bool null)`

Sets the NULL state of the field to specified value.

`virtual std::string toString() const =0`

Returns a string representation of the field.

`virtual NzaeDataTypes::Types type() const`

=0 Returns the type of the field.

`virtual ~NzaeField()`

Detailed Description

Provides the field interface.

See Also

`NzaeBoolField`

`NzaeInt8Field`

`NzaeInt16Field`

`NzaeInt32Field`

`NzaeInt64Field`

`NzaeFloatField`

`NzaeDoubleField`

`NzaeNumericField`

`NzaeNumeric32Field`

`NzaeNumeric64Field`

`NzaeNumeric128Field`

`NzaeStringField`

`NzaeFixedStringField`

`NzaeVariableStringField`

`NzaeNationalFixedStringField`

`NzaeNationalVariableStringField`

`NzaeGeometryStringField`

`NzaeVarbinaryStringField`

`NzaeDateField`

`NzaeTimeField`

`NzaeTimestampField`

NzaeTimeTzField
NzaeIntervalField

Public Member Function Documentation

void assign(NzaeField &field)

Assigns the value of the argument to the field object.

Parameters

NzaeField field

The field to assign.

The field argument may be a different type, as long as it is compatible.

virtual void fromString(std::string str)=0

Constructs the field from the string.

Parameters

str

The string to set value from.

bool isNull() const

Determines whether the field is NULL.

Returns

TRUE if the field is NULL.

NzaeField()

Constructs a NULL field.

NzaeField& operator=(NzaeField &field)

Assigns the value of the argument to the field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeField

The field argument may be a different type, as long as it is compatible.

void setNull(bool null)

Sets the NULL state of the field to specified

value. ▲ Parameters

null

TRUE if the field should be NULL.

virtual std::string toString() const =0

Returns a string representation of the field.

Returns

The string representation of the field.

virtual NzaeDataTypes::Types type() const**=0** Returns the type of the field.

Returns

Types

The field type.

virtual ~NzaeField()

NzaeFixedStringField Class Reference

This class provides field access for type fixed string.

Inherits NzaeStringField

Public Member Functions

int length() const Gets

the string length.

virtual NzaeDataTypes::Types type()**const** Returns the type of the field.

Detailed Description

This class provides field access for type fixed string.

See Also

▲ NzaeStringField

Public Member Function Documentation

int length() const Gets

the string length.

Returns

The string length in bytes.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeFloatField Class Reference

This class provides field access for type float.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

NzaeFloatField()

Constructs a NULL float field.

NzaeFloatField(NzaeFloatField &field)

Constructs a float field with value field.

NzaeFloatField(float val)

Constructs a float field with value val.

operator float()

Returns the float field value.

NzaeFloatField& operator=(NzaeFloatField &field)

Assigns the value of the argument to a field object.

NzaeFloatField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

NzaeFloatField& operator=(float val)

Assigns the value of the argument to a field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Detailed Description

This class provides field access for type float.

See Also

▲ NzaeField

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

NzeFloatField()

Constructs a NULL float field.

NzeFloatField(NzeFloatField &field)

Constructs a float field with value field.

Parameters

NzeFloatField field The
NzeFloatField value.

NzeFloatField(float val)

Constructs a float field with value val.

Parameters

val

The float value.

operator float()

Returns the float field value.

Returns

The float value.

NzeFloatField& operator=(NzeFloatField &field)

Assigns the value of the argument to a field object.

Parameters

NzeFloatField field

The field to assign.

Returns

NzeFloatField

NzeFloatField& operator=(NzeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeFloatField

The field argument may be a different type, as long as it is compatible.

NzaeFloatField& operator=(float val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeFloatField

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeFunction Class Reference

This class provides Function functionality and is used to implement Function AEs.

Public Types

```
enum LogLevel {  
    LOG_TRACE=1, LOG_DEBUG=2  
} Log Level.
```

Public Member Functions

virtual void close()=0
Closes the AE and releases its resources.

virtual NzaeRecord* createOutputRecord() const
=0 Create a new output record.

virtual void done() const
=0 Indicates done.

virtual const NzaeEnvironment& getEnvironment() const
=0 Gets environment information for the AE.

virtual const NzaeLibrary& getLibrary() const
=0 Gets library information for the AE.

virtual NzaeFunctionMessageHandler& getMessageHandler() const
=0 Returns the message handler class object.

virtual const NzaeMetadata& getMetadata() const =0
Gets metadata about the AE including the input and output columns.

virtual const NzaeParameters& getParameters() const
=0 Gets parameter information for the AE.

virtual const NzaeRuntime& getRuntime() const =0
Gets runtime information for the AE, including information about the Netezza software.

virtual void log(LogLevel logLevel, const char *message) const
=0 Logs the specified message at the given log level.

virtual std::string logFileName() const
=0 Returns the log file name.

virtual NzaeRecord* next()=0
Gets the next input row.

virtual bool nextPartition()=0
Returns TRUE if there is another partition.

virtual void outputResult(NzaeRecord
&rec)=0 Outputs the record.

virtual void ping() const =0
Indicates that the AE is still active and not hanging.

virtual void run(NzaeFunctionMessageHandler
*messageHandler)=0 Runs the function handler.

virtual void userError(const char *message) const =0
Indicates the AE has encountered an error condition.

virtual ~NzaeFunction()

Static Public Member Functions

static NzaeFunction* newInstance(NzaeFunctionInitialization &arg, NZAE_HANDLE handle)

Detailed Description

This class provides Function functionality and is used to implement Function AEs.

See Also

▲ NzaeFunctionMessageHandler

NzaeFactory
NzaeApi
NzaeLibrary
NzaeParameters
NzaeEnvironment
NzaeMetadata
NzaeRecord

Enumeration Type Documentation

enum LogLevel
Log Level.
LOG_TRACE
LOG_DEBUG

Public Member Function Documentation

virtual void close()=0
Closes the AE and releases its resources.
Release all resources associated with the function.

virtual NzaeRecord* createOutputRecord() const
=0 Create a new output record.

Returns
NzaeRecord
An instance of NzaeRecord with NULL fields.

Creates a new NzaeRecord object compatible for output. To be compatible, the object has the correct number of fields of the correct database type in the correct order.

See Also
► NzaeRecord

virtual void done() const
=0 Indicates done.
Indicates the AE is finishing successfully, getting no more rows and outputting no more results.

virtual const NzaeEnvironment& getEnvironment() const
=0 Gets environment information for the AE.

Returns
NzaeEnvironment

The instance of NzaeEnvironment .

See Also

NzaeEnvironment

virtual const NzaeLibrary& getLibrary() const

=0 Gets library information for the AE.

Returns

NzaeLibrary

The instance of NzaeLibrary .

See Also

NzaeLibrary

virtual NzaeFunctionMessageHandler& getMessageHandler() const

=0 Returns the message handler class object.

Returns

NzaeFunctionMessageHandler

The instance of NzaeFunctionMessageHandler .

The message handler is where custom function logic is implemented.

See Also

NzaeFunctionMessageHandler

virtual const NzaeMetadata& getMetadata() const =0

Gets metadata about the AE including the input and output columns.

Returns

NzaeMetadata

The instance of NzaeMetadata .

See Also

NzaeMetadata

virtual const NzaeParameters& getParameters() const

=0 Gets parameter information for the AE.

Returns

NzaeParameters

The instance of NzaeParameters .

See Also

NzaeParameters

virtual const NzaeRuntime& getRuntime() const =0

Gets runtime information for the AE, including information about the Netezza software.

Returns

NzaeRuntime

The instance of NzaeRuntime .

See Also

NzaeRuntime

virtual void log(LogLevel logLevel, const char *message) const
=0 Logs the specified message at the given log level.

Parameters

LogLevel logLevel The
log level constant.

message
The message to log.

virtual std::string logFileName() const
=0 Returns the log file name.

Returns

The log file name.

virtual NzaeRecord* next()=0
Gets the next input row.

Returns

NzaeRecord

An instance of NzaeRecord or NULL when there is no more data.

See Also

nzaeRecord

virtual bool nextPartition()=0
Returns TRUE if there is another partition.

Returns

TRUE if there is another partition.

In non-partition mode, the function returns TRUE once at the start of input.

In partition mode, if nextPartition has been called, the function returns TRUE at the start of a partition. At the end of a partition, the next function returns NULL, and nextPartition must be called before the next function can return data for the following partition.

If nextPartition has never been called, then next returns data for all the partitions.

**virtual void outputResult(NzaeRecord
&rec)=0** Outputs the record.

Parameters

NzaeRecord rec

An output compatible instance of NzaeRecord .

See Also

NzaeRecord

virtual void ping() const =0

Indicates that the AE is still active and not hanging.

**virtual void run(NzaeFunctionMessageHandler
*messageHandler)=0** Runs the function handler.

Parameters

NzaeFunctionMessageHandler

messageHandler The message handler.

Begins the Function Message Processing. Processes one row of input and produces one row of output. Used for scalar functions and some table functions. Scalar functions use only one field in the result.

This function can be used as an alternative to writing a for loop with next and outputResult.

The message handler is where custom logic is implemented.

See Also

NzaeFunctionMessageHandler

virtual void userError(const char *message) const =0

Indicates the AE has encountered an error condition.

Parameters

message

The message to send back to the Netezza software.

Implies NzaeDone.

virtual ~NzaeFunction()

Static Public Member Function Documentation

static NzaeFunction* newInstance(NzaeFunctionInitialization &arg, NZAE_HANDLE handle)

Returns

NzaeFunction

NzaeFunctionInitialization Class Reference

Not implemented. This class is a placeholder for future functionality.

Detailed Description

Not implemented. This class is a placeholder for future functionality.

See Also

NzaeFactory

NzaeApi

NzaeFunctionMessageHandler Interface Reference

This class allows implementation of higher level functions.

Public Member Functions

virtual void evaluate(NzaeFunction &api, NzaeRecord &input, NzaeRecord &result)=0 Processes one row of input and produces one row of output.

virtual ~NzaeFunctionMessageHandler()

Detailed Description

This class allows implementation of higher level functions.

Implement this class to handle NzaeFunction messages.

See Also

run

NzaeRecord

Public Member Function Documentation

virtual void evaluate(NzaeFunction &api, NzaeRecord &input, NzaeRecord &result)=0 Processes one row of input and produces one row of output.

Parameters

NzaeFunction api

The function object.

NzaeRecord input

The input record.

NzaeRecord result

The result record.

Used for scalar functions and some table functions that output only one column and one row of output per input.

Scalar functions only use one field in the result.

See Also

► NzaeFunction

NzaeRecord

virtual ~NzaeFunctionMessageHandler()

NzaeGeometryStringField Class Reference

This class provides field access for type geometry string.

Inherits NzaeStringField

Public Member Functions

int length() const Gets
the string length.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Detailed Description

This class provides field access for type geometry string.

See Also

▲ NzaeStringField

Public Member Function Documentation

int length() const Gets
the string length.

Returns
The string length in bytes.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Returns
Types
The field type.

NzaeInt16Field Class Reference

This class provides field access for type int16.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

NzaeInt16Field()

Constructs a NULL int16 field.

NzaeInt16Field(NzaeInt16Field &field)

Constructs an int16 field with value field.

NzaeInt16Field(int16_t val)

Constructs an int16 field with value val.

operator int16_t()

Returns an int16 field value.

NzaeInt16Field& operator=(NzaeInt16Field &field)

Assigns the value of the argument to the field object.

NzaeInt16Field& operator=(NzaeField &field) Assigns the value of the argument to the field object.

NzaeInt16Field& operator=(int16_t val)

Assigns the value of the argument to the field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Detailed Description

This class provides field access for type int16.

See Also

▲ NzaeField

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

NzaeInt16Field()

Constructs a NULL int16 field.

NzeInt16Field(NzeInt16Field &field)

Constructs an int16 field with value field.

Parameters

NzeInt16Field field

The NzeInt16Field value.

NzeInt16Field(int16_t val) Constructs an int16 field with value val.

Parameters

val

The int16 value.

operator int16_t()

Returns an int16 field value.

Returns

int16 The value.

NzeInt16Field& operator=(NzeInt16Field &field)

Assigns the value of the argument to the field object.

Parameters

NzeInt16Field field

The field to assign.

Returns

NzeInt16Field

NzeInt16Field& operator=(NzeField &field) Assigns the value of the argument to the field object.

Parameters

NzeField field

The field to assign.

Returns

NzeInt16Field

The field argument may be a different type, as long as it is compatible.

NzeInt16Field& operator=(int16_t val)

Assigns the value of the argument to the field object.

Parameters

val

The value to assign.

Returns

NzaeInt16Field

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeInt32Field Class Reference

This class provides field access for type int32.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

NzaeInt32Field()

Constructs a NULL int32 field.

NzaeInt32Field(NzaeInt32Field &field)

Constructs an int32 field with value field.

NzaeInt32Field(int32_t val)

Constructs an int32 field with value val.

operator int32_t()

Returns an int32 field value.

NzaeInt32Field& operator=(NzaeInt32Field &field)

Assigns the value of the argument to the field object.

NzaeInt32Field& operator=(NzaeField &field) Assigns the value of the argument to the field object.

NzaeInt32Field& operator=(int32_t val)

Assigns the value of the argument to the field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()
 const Returns the type of the field.

Detailed Description

This class provides field access for type int32.

See Also

▲ NzaeField

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

NzaeInt32Field()

Constructs a NULL int32 field.

NzaeInt32Field(NzaeInt32Field &field)

Constructs an int32 field with value field.

Parameters

NzaeInt32Field field

The NzaeInt32Field value.

NzaeInt32Field(int32_t val) Constructs
 an int32 field with value val.

Parameters

val

The int32 value.

operator int32_t()

Returns an int32 field value.

Returns

The int32 value.

NzaeInt32Field& operator=(NzaeInt32Field &field)

Assigns the value of the argument to the field object.

Parameters

NzaeInt32Field field

The field to assign.

Returns

NzaeInt32Field

NzaeInt32Field& operator=(NzaeField &field) Assigns the value of the argument to the field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeInt32Field

The field argument may be a different type, as long as it is compatible.

NzaeInt32Field& operator=(int32_t val)

Assigns the value of the argument to the field object.

Parameters

val

The value to assign.

Returns

NzaeInt32Field

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeInt64Field Class Reference

This class provides field access for type int64.

Inherits NzaeField

Public Member Functions

`void fromString(std::string str)`

Constructs the field from the string.

`NzeInt64Field()`

Constructs a NULL int64 field.

`NzeInt64Field(NzeInt64Field &field)`

Constructs an int64 field with value field.

`NzeInt64Field(int64_t val)`

Constructs an int64 field with value val.

`operator int64_t()`

Returns an int64 field value.

`NzeInt64Field& operator=(NzeInt64Field &field)`

Assigns the value of the argument to the field object.

`NzeInt64Field& operator=(NzeField &field)` Assigns the value of the argument to the field object.

`NzeInt64Field& operator=(int64_t val)`

Assigns the value of the argument to the field object.

`std::string toString() const`

Returns the string representation of the field.

`virtual NzeDataTypes::Types type()`

const Returns the type of the field.

Detailed Description

This class provides field access for type int64.

See Also

▲ [NzeField](#)

Public Member Function Documentation

`void fromString(std::string str)`

Constructs the field from the string.

Parameters

str

The string to assign from.

`NzeInt64Field()`

Constructs a NULL int64 field.

`NzeInt64Field(NzeInt64Field &field)`

Constructs an int64 field with value field.

Parameters

NzaeInt64Field field

The NzaeInt64Field value.

NzaeInt64Field(int64_t val) Constructs an int64 field with value val.

Parameters

val

The int64 value.

operator int64_t()

Returns an int64 field value.

Returns

The int64 value.

NzaeInt64Field& operator=(NzaeInt64Field &field)

Assigns the value of the argument to the field object.

Parameters

NzaeInt64Field field

The field to assign.

Returns

NzaeInt64Field

NzaeInt64Field& operator=(NzaeField &field) Assigns the value of the argument to the field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeInt64Field

The field argument may be a different type, as long as it is compatible.

NzaeInt64Field& operator=(int64_t val)

Assigns the value of the argument to the field object.

Parameters

val

The value to assign.

Returns

NzaeInt64Field**std::string toString() const**

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeInt8Field Class Reference

This class provides field access for type int8.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

NzaeInt8Field()

Constructs a NULL int8 field.

NzaeInt8Field(NzaeInt8Field &field)

Constructs an int8 field with value field.

NzaeInt8Field(int8_t val)

Constructs an int8 field with value val.

operator int8_t()

Returns an int8 field value.

NzaeInt8Field& operator=(NzaeInt8Field &field)

Assigns the value of the argument to the field object.

NzaeInt8Field& operator=(NzaeField &field)

Assigns the value of the argument to the field object.

NzaeInt8Field& operator=(int8_t val)

Assigns the value of the argument to the field object.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Detailed Description

This class provides field access for type int8.

See Also

▲ NzaeField

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

NzaeInt8Field()

Constructs a NULL int8 field.

NzaeInt8Field(NzaeInt8Field &field)

Constructs an int8 field with value field.

Parameters

NzaeInt8Field field

The NzaeInt8Field value.

NzaeInt8Field(int8_t val)

Constructs an int8 field with value val.

Parameters

val

The int8 value.

operator int8_t()

Returns an int8 field value.

Returns

The int8 value.

NzaeInt8Field& operator=(NzaeInt8Field &field)

Assigns the value of the argument to the field object.

Parameters

NzaeInt8Field field

The field to assign.

Returns
NzeInt8Field

NzeInt8Field& operator=(NzeField &field) Assigns the value of the argument to the field object.

Parameters
NzeField field
 The field to assign.

Returns
NzeInt8Field

The field argument may be a different type, as long as it is compatible.

NzeInt8Field& operator=(int8_t val)
 Assigns the value of the argument to the field object.

Parameters
val
 The value to assign.

Returns
NzeInt8Field

std::string toString() const
 Returns the string representation of the field.

Returns
 The string representation.

virtual NzeDataTypes::Types type()
const Returns the type of the field.

Returns
Types
 The field type.

NzeIntervalField Class Reference

This class provides field access for type interval.

Inherits NzeField

Public Member Functions

void fromString(std::string str)
 Construct the field from the string.

`bool isValidInterval() const`
Determines whether a Netezza-encoded Interval value is valid and within range.

`NzIntervalField()`
Constructs a NULL interval field.

`NzIntervalField(const NzIntervalField &field)`
Constructs an interval field with value field.

`NzIntervalField(NzInterval val)`
Constructs an interval field with value val.

`operator const NzTimeField()`
Returns the time field value.

`operator const NzInterval &() const`
Returns the encoded field value.

`operator NzInterval &()`
Returns the encoded field value.

`bool operator!=(const NzIntervalField &x)`
Returns Not Equal.

`bool operator<(const NzIntervalField &x)`
Returns Less than.

`bool operator<=(const NzIntervalField &x)`
Returns Less than or equal.

`NzIntervalField& operator=(NzField &field)`
Assigns the value of the argument to a field object.

`NzIntervalField& operator=(const NzIntervalField &field)`
Assigns the value of the argument to a field object.

`NzIntervalField& operator=(NzInterval val)`
Assigns the value of the argument to a field object.

`bool operator==(const NzIntervalField &x)`
Returns Equal to.

`bool operator>(const NzIntervalField &x)`
Returns Greater than.

`bool operator>=(const NzIntervalField &x)`
Returns Greater than or equal.

`std::string toString() const`
Returns the string representation of the field.

`virtual NzDataTypes::Types type()`
Returns the type of the field.

Detailed Description

This class provides field access for type interval.

See Also

▲ NzaeField

Public Member Function Documentation

void fromString(std::string str)

Construct the field from the string.

Parameters

str

The string to assign from.

bool isValidInterval() const

Determines whether a Netezza-encoded Interval value is valid and within range.

Returns

FALSE if intervalMonth < ENC_INTERVAL_MONTH_MIN or
intervalMonth > ENC_INTERVAL_MONTH_MAX. TRUE otherwise.

NzaeIntervalField()

Constructs a NULL interval field.

NzaeIntervalField(const NzaeIntervalField &field)

Constructs an interval field with value field.

Parameters

NzaeIntervalField field The

NzaeIntervalField value.

NzaeIntervalField(NzudsInterval val)

Constructs an interval field with value val.

Parameters

val

The encoded interval value.

operator const NzaeTimeField()

const Returns the time field value.

Returns

The time value converted from the interval.

See Also

NzaeTimeField

operator const NzudsInterval &()

const Returns the encoded field value.

Returns
The encoded value.

operator NzudsInterval &()

Returns the encoded field value.

Returns
The encoded value.

bool operator!=(const NzaeIntervalField &x)

const Not Equal.

Parameters
NzaeIntervalField
x Field to compare.

Returns
true if field is not equal to x

Exceptions
NzaeException

bool operator<(const NzaeIntervalField &x)

const Less than.

Parameters
NzaeIntervalField
x Field to compare.

Returns
True if the field is less than x.

Exceptions
NzaeException

bool operator<=(const NzaeIntervalField &x)

const Less than or equal.

Parameters
NzaeIntervalField
x Field to compare.

Returns
TRUE if the field is less than or equal to x.

Exceptions
NzaeException

NzaeIntervalField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzeField field

The field to assign.

Returns

NzeIntervalField

The field argument may be a different type, as long as it is compatible.

**NzeIntervalField& operator=(const NzeIntervalField
&field)** Assigns the value of the argument to a field object.

Parameters

NzeIntervalField field

The field to assign.

Returns

NzeIntervalField

NzeIntervalField& operator=(NzudsInterval val)

Assigns the value of the argument to a field object.

Parameters

val

The encoded value to assign.

Returns

NzeIntervalField

bool operator==(const NzeIntervalField &x)
const Equal to.

Parameters

NzeIntervalField

x Field to compare.

Returns

TRUE if the field is equal to x.

Exceptions

NzeException

bool operator>(const NzeIntervalField &x)
const Greater than.

Parameters

NzeIntervalField

x Field to compare.

Returns

TRUE if the field is greater than x.

Exceptions

NzaeException

bool operator>=(const NzaeIntervalField &x)

const Greater than or equal.

Parameters

NzaeIntervalField

x Field to compare.

Returns

TRUE if the field is greater than or equal to x.

Exceptions

NzaeException

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeLibrary Class Reference

This class provides access to the AE shared library information.

Public Types

```
enum NzaeLibrarySearchType {  
    NzaeLibrarySearchBoth, NzaeLibrarySearchLocal, NzaeLibrarySearchParent }
```

Specifies whether to search parent or child information.

NzaeLibrarySearchType

Public Member Functions

```
virtual void addEntry(std::string name, std::string path, bool autoLoad, bool local)=0  
virtual const NzaeLibraryInfo* const getLibraryInfo(std::string name, bool caseSensitive,  
    Nza-eLibrarySearchType type) const =0
```

Gets Library information by name.

```
virtual const NzaeLibraryInfo* const getLocalLibraryInfo(int idx) const
=0 Gets the parent library information by index.
```

```
virtual const NzaeLibraryInfo* const getParentLibraryInfo(int idx) const
=0 Gets the local library information by index.
```

```
virtual void setReadOnly()=0
virtual int sizeLocalEntries() const =0
Gets the number of local entries.
```

```
virtual int sizeParentEntries() const =0
Gets the number of parent entries.
```

```
virtual ~NzaeLibrary()
```

Static Public Member Functions

```
static NzaeLibrary* create()
```

Detailed Description

This class provides access to the AE shared library information.

See Also

- NzaeFunction
- NzaeAggregate
- NzaeShaper

Enumeration Type Documentation

```
enum NzaeLibrarySearchType
Specifies whether to search parent or child information.
```

NzaeLibrarySearchBoth

NzaeLibrarySearchLocal

NzaeLibrarySearchParent

Typedef Documentation

```
typedef enum nz::ae::NzaeLibrary::NzaeLibrarySearchType
NzaeLibrarySearchTypeNzaeLib-rarySearchType
```

Public Member Function Documentation

```
virtual void addEntry(std::string name, std::string path, bool autoLoad, bool local)=0
```

```
virtual const NzaeLibraryInfo* const getLibraryInfo(std::string name, bool caseSensitive,
NzaeLib-rarySearchType type) const =0
```

Gets Library information by name.

Parameters

name

The name of library.

caseSensitive

If FALSE, performs a case-insensitive search.

NzaeLibrarySearchType type

Search Local, Parent or Both.

Returns

NzaeLibraryInfo

The library information or NULL. Does not need to be deleted.

In remote mode, there is a parent and local context that may be different. "Parent" refers to the libraries used by the AE launcher of the remote AE service process. "Local" refers to the shared libraries specified for the current remote AE instance that is connected to the remote AE service.

See Also

NzaeLibraryInfo

virtual const NzaeLibraryInfo* const getLocalLibraryInfo(int idx) const

=0 Gets the parent library information by index.

Parameters

idx

The index to look up.

Returns

NzaeLibraryInfo

Library information or NULL. Does not need to be deleted.

Exceptions

NzaeException

See Also

NzaeLibraryInfo

virtual const NzaeLibraryInfo* const getParentLibraryInfo(int idx) const

=0 Gets the local library information by index.

Parameters

idx

Index to look up.

Returns

NzaeLibraryInfo

Library information or NULL. Does not need to be deleted.

Exceptions
 NzaeException
 See Also
 NzaeLibraryInfo

virtual void setReadOnly()=0

virtual int sizeLocalEntries() const
=0 Gets the number of local entries.

Returns
 The number of local entries.

Local entries are those associated with the AE.

virtual int sizeParentEntries() const
=0 Gets the number of parent entries.

Returns
 The number of parent entries.

Parent entries are those associated with the parent process in the case of a remote AE.

virtual ~NzaeLibrary()

Static Public Member Function Documentation

static NzaeLibrary* create()
 Returns
 NzaeLibrary

NzaeLibraryInfo Class Reference

This class provides information about an AE shared library.

Public Attributes

autoLoad
 The library autoloader status.
 libraryFullPath
 The library path.
 libraryName
 The library name.

Detailed Description

This class provides information about an AE shared library.

See Also

▲ [NzaeLibrary](#)

Member Data Documentation

bool autoLoad

The library autoloading status.

std::string libraryFullPath

The library path.

std::string libraryName

The library name.

NzaeMetadata Class Reference

This class provides AE Metadata information, containing data about the AE, including input and output column attributes. Column indexes are zero-based.

Public Types

```
enum NzaeCorrelationType {
    NzaeUnknownCorrelationType= 0, NzaeUncorrelated= 1, NzaeInnerCorrelation= 2,
    NzaeLeft-Correlation= 3 }
```

Correlation type for table Functions.

NzaeCorrelationType

Public Member Functions

```
NzaeCorrelationType getCorrelationType() const
Gets the correlation type.
```

```
int getInputColumnCount() const
Gets the number of input columns.
```

```
int getInputScale(int index) const
Gets the input column scale.
```

```
int getInputSize(int index) const
Gets the input column size.
```


NzaeDataTypes::Types getInputType(int index)

const Gets the input data type.

int getOutputColumnCount() const

Gets the number of output columns.

int getOutputScale(int index) const

Gets the output column scale.

int getOutputSize(int index) const

Gets the output column size.

NzaeDataTypes::Types getOutputType(int index)

const Gets the output data type.

bool hasFinal() const

Specifies if the function was invoked with a FINAL clause.

bool hasOrder() const

Specifies if the function was invoked with an ORDER BY clause.

bool hasOver() const

Specifies if the function invoked with an OVER clause.

bool hasPartition() const

Specifies if the function was invoked with a PARTITION BY clause.

bool inputsConstant(int index) const

Determines whether the input is constant.

bool isOneOutputRowRestriction() const

Determines if the function is scalar.

NzaeMetadata(int inputColumnCount, NzaeDataTypes::Types *inputTypes, int *inputsConstant, int *inputSizes, int *inputScales, int outputColumnCount, NzaeDataTypes::Types *outputTypes, int *outputSizes, int *outputScales, bool oneRow, int correlationType, bool hasFinal, bool hasOver, bool hasSort, bool hasPartition)

~NzaeMetadata()

Detailed Description

This class provides AE Metadata information, containing data about the AE, including input and output column attributes. Column indexes are zero-based.

See Also

getMetadata

NzaeDataTypes

NzaeShaper

Enumeration Type Documentation

enum NzaeCorrelationType

Correlation type for table Functions.

NzaeUnknownCorrelationType

NzaeUncorrelated

NzaeInnerCorrelation

NzaeLeftCorrelation

Typedef Documentation

```
typedef enum nz::ae::NzaeMetadata::NzaeCorrelationType
NzaeCorrelationTypeNzaeCorrelationType
```

Public Member Function Documentation

NzaeCorrelationType getCorrelationType()

const Gets the correlation type.

Returns

NzaeCorrelationType

The correlation type.

int getInputColumnCount() const

Gets the number of input columns.

Returns

The number of input columns.

int getInputScale(int index) const

Gets the input column scale.

Parameters

index

The input index.

Returns

The scale of input column.

Exceptions

NzaeException

int getInputSize(int index) const

Gets the input column size.

Parameters

index

The input index.

Returns

The length for string type; precision for numeric type.

Exceptions

NzaeException

NzaeDataTypes::Types getInputType(int index)

const Gets the input data type.

Parameters

index

The input index.

Returns

Types

The input data type.

Exceptions

NzaeException

int getOutputColumnCount() const

Gets the number of output columns.

Returns

The number of output columns.

int getOutputScale(int index) const

Gets the output column scale.

Parameters

index

The output index.

Returns

The scale of output column.

Exceptions

NzaeException

int getOutputSize(int index) const

Gets the output column size.

Parameters

index

The output index.

Returns

The length for string type; precision for numeric type.

Exceptions

NzaeException

NzaeDataTypes::Types getOutputType(int index)

const Gets the output data type.

Parameters

index

The output index.

Returns

Types

The output data type.

Exceptions

NzaeException

bool hasFinal() const

Specifies if the function was invoked with a FINAL clause.

Returns

TRUE if the table function invoked with TABLE WITH FINAL.

bool hasOrder() const

Specifies if the function was invoked with an ORDER BY clause.

Returns

TRUE if the table function invoked with ORDER.

bool hasOver() const

Specifies if the function invoked with an OVER clause.

Returns

TRUE if the table function invoked with OVER.

bool hasPartition() const

Specifies if the function was invoked with a PARTITION BY clause.

Returns

TRUE if the table function invoked with PARTITION BY.

bool inputIsConstant(int index) const

Determines whether the input is constant.

Parameters

index

The input index.

Returns

TRUE if the value of this column is constant for all rows.

Exceptions

NzaeException

bool isOneOutputRowRestriction() const

Determines if the function is scalar.

Returns

TRUE if a scalar function.

NzaeMetadata(int inputColumnCount, NzaeDataTypes::Types *inputTypes, int *inputIsConstant, int *inputSizes, int *inputScales, int outputColumnCount, NzaeDataTypes::Types *outputTypes, int *outputSizes, int *outputScales, bool oneRow, int correlationType, bool hasFinal, bool hasOver, bool hasSort, bool hasPartition)

~NzaeMetadata()

NzaeNationalFixedStringField Class Reference

This class provides field access for type national fixed string.

Inherits NzaeStringField

Public Member Functions

bool isValidUTF8() const

Determines if the string is valid UTF8.

int length() const Gets

the string length.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Detailed Description

This class provides field access for type national fixed string.

See Also

▲ NzaeStringField

Public Member Function Documentation

bool isValidUTF8() const Determines if the string is valid UTF8.

Returns

TRUE if the string is valid UTF8.

int length() const Gets

the string length.

Returns
The string length in characters, not bytes.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Returns
Types
The field type.

NzaeNationalVariableStringField Class Reference

This class provides field access for type national variable string.

Inherits NzaeStringField

Public Member Functions


bool isValidUTF8() const
Determines if the string is valid UTF8.

int length() const Gets
the string length.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Detailed Description

This class provides field access for type national variable string.

See Also
 [NzaeStringField](#)

Public Member Function Documentation

bool isValidUTF8() const Determines
if the string is valid UTF8.

Returns
TRUE if the string is valid UTF8.

int length() const Gets
the string length.

Returns

The string length in characters, not bytes.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeNumeric128Field Class Reference

This class provides field access for type Numeric128.

Inherits NzaeNumericField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

void fromStringWithInfo(std::string str, int precision, int scale) Constructs the field from the string.

NzaeNumeric128Field(const NzaeNumericField &field)

Constructs a numeric128 field with value field.

NzaeNumeric128Field(int32_t val) Constructs

a numeric128 field with value val.

NzaeNumeric128Field()

Constructs a NULL numeric128.

NzaeNumeric128Field(const NzaeNumeric128Field &field) Constructs a numeric128 field with value field.

NzaeNumeric128Field(const NzudsNumeric128 val) Constructs a numeric128 field with value val.

NzaeNumeric128Field(double val) Constructs a numeric128 field with value val.

NzaeNumeric128Field(int64_t val) Constructs a numeric128 field with value val.

operator const NzudsNumeric128()

const Returns a numeric128 value.

operator double() const

Returns the value converted to a double.

operator NzudsNumeric128()

Returns a numeric128 value.

NzaeNumeric128Field& operator=(const NzaeNumeric128Field &field)

Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(const NzudsNumeric128 val)` Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(int32_t val)`
Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(const NzaeNumericField &val)` Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(double val)`
Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(NzaeField &field)`
Assigns the value of the argument to a field object.

`NzaeNumeric128Field& operator=(int64_t val)`
Assigns the value of the argument to a field object.

`std::string toString() const`
Returns the string representation of the field.

`virtual NzaeDataTypes::Types type()`
`const` Returns the type of the field.

Detailed Description

This class provides field access for type Numeric128.

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

void fromStringWithInfo(std::string str, int precision, int scale) Constructs the field from the string.

Parameters

str

The string to assign from.

precision

The precision to use.

scale

The scale to use.

Uses the specified precision scale, not the scale from the string.

**NzaeNumeric128Field(const NzaeNumericField
&field)** Constructs a numeric128 field with value field.

Parameters

**NzaeNumericField
field** The field.

The field argument may be a different type.

NzaeNumeric128Field(int32_t val)

Constructs a numeric128 field with value val.

Parameters

val
The int32_t value.

NzaeNumeric128Field()

Constructs a NULL numeric128.

**NzaeNumeric128Field(const NzaeNumeric128Field
&field)** Constructs a numeric128 field with value field.

Parameters

NzaeNumeric128Field field
The Numeric128 field.

**NzaeNumeric128Field(const NzudsNumeric128
val)** Constructs a numeric128 field with value val.

Parameters

val
The Numeric128 value.

This function reorders the digits. Use only with structures coming from serialization.

NzaeNumeric128Field(double val) Constructs
a numeric128 field with value val.

Parameters

val
The double value.

NzaeNumeric128Field(int64_t val)

Constructs a numeric128 field with value val.

Parameters

val

The int64_t value.

operator const NzudsNumeric128()

const Returns a numeric128 value.

Returns

The numeric128 value.

This function reorders the digits. Use only with structures going to serialization.

operator double() const

Returns the value converted to a double.

Returns

The converted double value.

operator NzudsNumeric128()

Returns a numeric128 value.

Returns

The numeric128 value.

This function reorders the digits. Use only with structures going to serialization.

**NzaeNumeric128Field& operator=(const NzaeNumeric128Field
&field)** Assigns the value of the argument to a field object.

Parameters

NzaeNumeric128Field field

The field to assign.

Returns

NzaeNumeric128Field

**NzaeNumeric128Field& operator=(const NzudsNumeric128
val)** Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric128Field

This function reorders the digits. Use only with structures coming from serialization.

NzaeNumeric128Field& operator=(int32_t val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric128Field

NzaeNumeric128Field& operator=(const NzaeNumericField &val) Assigns the value of the argument to a field object.

Parameters

NzaeNumericField val

The field to assign.

Returns

NzaeNumeric128Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric128Field& operator=(double val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric128Field

NzaeNumeric128Field& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeNumeric128Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric128Field& operator=(int64_t val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric128Field

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeNumeric32Field Class Reference

This class provides field access for type Numeric32.

Inherits NzaeNumericField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

void fromStringWithInfo(std::string str, int precision, int scale) Constructs the field from the string.

NzaeNumeric32Field(const NzaeNumericField &field)

Constructs a numeric32 field with value field.

NzaeNumeric32Field(int32_t val) Constructs

a numeric32 field with value val.

NzaeNumeric32Field()

Constructs a NULL numeric32.

NzaeNumeric32Field(const NzaeNumeric32Field

&field) Constructs a numeric32 field with value field.

NzaeNumeric32Field(const NzudsNumeric32 val)

Constructs a numeric32 field with value val.

NzaeNumeric32Field(double val) Constructs

a numeric32 field with value val.

NzaeNumeric32Field(int64_t val) Constructs

a numeric32 field with value val.

operator const NzudsNumeric32 &()
const Returns a numeric32 value.

operator double() const
Returns a value converted to a double.

operator NzudsNumeric32 &()
Returns a numeric32 value.

NzaeNumeric32Field& operator=(NzaeField &field)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(const NzudsNumeric32 val)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(const NzaeNumeric32Field &field)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(const NzaeNumericField &val)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(int32_t val)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(int64_t val)
Assigns the value of the argument to a field object.

NzaeNumeric32Field& operator=(double val)
Assigns the value of the argument to a field object.

std::string toString() const
Returns the string representation of the field.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Detailed Description

This class provides field access for type Numeric32.

See Also

▲ NzaNumericField

Public Member Function Documentation

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

void fromStringWithInfo(std::string str, int precision, int scale)
Constructs the field from the string.

Parameters

str

The string to assign from.

precision

The precision to use.

scale

The scale to use.

Uses the specified precision scale, not the scale from the string.

**NzaeNumeric32Field(const NzaeNumericField
&field)** Constructs a numeric32 field with value field.

Parameters

NzaeNumericField

field The field.

The field argument may be a different type, as long as it is compatible.

NzaeNumeric32Field(int32_t val)

Constructs a numeric32 field with value val.

Parameters

val

The int32_t value.

NzaeNumeric32Field()

Constructs a NULL numeric32.

**NzaeNumeric32Field(const NzaeNumeric32Field
&field)** Constructs a numeric32 field with value field.

Parameters

NzaeNumeric32Field

field The Numeric32 field.

**NzaeNumeric32Field(const NzudsNumeric32
val)** Constructs a numeric32 field with value val.

Parameters

val

The Numeric32 value.

NzaeNumeric32Field(double val) Constructs
a numeric32 field with value val.

Parameters

val

The double value.

NzaeNumeric32Field(int64_t val)

Constructs a numeric32 field with value val.

Parameters

val

The int64_t value.

operator const NzudsNumeric32 &()

const Returns a numeric32 value.

Returns

The numeric32 value.

operator double() const

Returns a value converted to a double.

Returns

The converted double value.

operator NzudsNumeric32 &()

Returns a numeric32 value.

Returns

The numeric32 value.

NzaeNumeric32Field& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeNumeric32Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric32Field& operator=(const NzudsNumeric32

val) Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric32Field

NzaeNumeric32Field& operator=(const NzaeNumeric32Field &field) Assigns the value of the argument to a field object.

Parameters

NzaeNumeric32Field

field The field to assign.

Returns

NzaeNumeric32Field

NzaeNumeric32Field& operator=(const NzaeNumericField &val) Assigns the value of the argument to a field object.

Parameters

NzaeNumericField val

The field to assign.

Returns

NzaeNumeric32Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric32Field& operator=(int32_t val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric32Field

NzaeNumeric32Field& operator=(int64_t val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric32Field

NzaeNumeric32Field& operator=(double val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric32Field**std::string toString() const**

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()**const** Returns the type of the field.

Returns

Types

The field type.

NzaeNumeric64Field Class Reference

This class provides field access for type Numeric64.

Inherits NzaeNumericField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

void fromStringWithInfo(std::string str, int precision, int scale) Constructs the field from the string.

NzaeNumeric64Field(const NzaeNumericField &field)

Constructs a numeric64 field with value field.

NzaeNumeric64Field(int32_t val) Constructs a numeric64 field with value val.

NzaeNumeric64Field()

Constructs a NULL numeric64.

NzaeNumeric64Field(const NzaeNumeric64Field &field) Constructs a numeric64 field with value field.

NzaeNumeric64Field(const NzudsNumeric64 val)

Constructs a numeric64 field with value val.

NzaeNumeric64Field(double val) Constructs a numeric64 field with value val.

NzaeNumeric64Field(int64_t val)

Constructs a numeric64 field with value val.

`operator const NzudsNumeric64()`

const Returns a numeric64 value.

`operator double() const`

Returns a value converted to a double.

`operator NzudsNumeric64()`

Returns a numeric64 value.

`NzaeNumeric64Field& operator=(NzaeField &field)`

Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(const NzaeNumeric64Field
&field)` Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(const NzaeNumericField
&val)` Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(const NzudsNumeric64
val)` Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(int32_t val)`

Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(int64_t val)`

Assigns the value of the argument to a field object.

`NzaeNumeric64Field& operator=(double val)`

Assigns the value of the argument to a field object.

`std::string toString() const`

Returns the string representation of the field.

`virtual NzaeDataTypes::Types type()`

const Returns the type of the field.

Detailed Description

This class provides field access for type Numeric64.

Public Member Function Documentation

`void fromString(std::string str)`

Constructs the field from the string.

Parameters

str

The string to assign from.

`void fromStringWithInfo(std::string str, int precision, int scale)`

Constructs the field from the string.

Parameters

str

The string to assign from.

precision

The precision to use.

scale

The scale to use.

Uses the specified precision scale, not the scale from the string.

**NzaeNumeric64Field(const NzaeNumericField
&field)** Constructs a numeric64 field with value field.

Parameters

NzaeNumericField

field The field.

The field argument may be a different type.

NzaeNumeric64Field(int32_t val)

Constructs a numeric64 field with value val.

Parameters

val

The int32_t value.

NzaeNumeric64Field()

Constructs a NULL numeric64.

**NzaeNumeric64Field(const NzaeNumeric64Field
&field)** Constructs a numeric64 field with value field.

Parameters

NzaeNumeric64Field

field The Numeric64 field.

**NzaeNumeric64Field(const NzudsNumeric64
val)** Constructs a numeric64 field with value val.

Parameters

val

The Numeric64 value.

This function reorders the digits. Use only with structures coming from serialization.

NzaeNumeric64Field(double val) Constructs a numeric64 field with value val.

Parameters

val

The double value.

NzaeNumeric64Field(int64_t val)

Constructs a numeric64 field with value val.

Parameters

val

The int64_t value.

operator const NzudsNumeric64()

const Returns a numeric64 value.

Returns

The numeric64 value.

This function reorders the digits. Use only with structures going to serialization.

operator double() const

Returns a value converted to a double.

Returns

The converted double value.

operator NzudsNumeric64()

Returns a numeric64 value.

Returns

The numeric64 value.

This function reorders the digits. Use only with structures going to serialization.

NzaeNumeric64Field& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeNumeric64Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric64Field& operator=(const NzaeNumeric64Field &field) Assigns the value of the argument to a field object.

Parameters

NzaeNumeric64Field
field The field to assign.

Returns

NzaeNumeric64Field

NzaeNumeric64Field& operator=(const NzaeNumericField &val) Assigns the value of the argument to a field object.

Parameters

NzaeNumericField val
 The field to assign.

Returns

NzaeNumeric64Field

The field argument may be a different type, as long as it is compatible.

NzaeNumeric64Field& operator=(const NzudsNumeric64 val) Assigns the value of the argument to a field object.

Parameters

val
 The value to assign.

Returns

NzaeNumeric64Field

This function reorders the digits. Use only with structures coming from serialization.

NzaeNumeric64Field& operator=(int32_t val)

Assigns the value of the argument to a field object.

Parameters

val
 The value to assign.

Returns

NzaeNumeric64Field

NzaeNumeric64Field& operator=(int64_t val)

Assigns the value of the argument to a field object.

Parameters

val
 The value to assign.

Returns

NzaeNumeric64Field

NzaeNumeric64Field& operator=(double val)

Assigns the value of the argument to a field object.

Parameters

val

The value to assign.

Returns

NzaeNumeric64Field

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

NzaeNumericField Class Reference

This class provides a common base class for the NzaeNumeric32Field , NzaeNumeric64Field , and NzaeNumeric128Field field classes.

Inherits NzaeField

Public Member Functions

virtual NzaeNumericField* abs()

const Gets the absolute value.

virtual NzaeNumericField* add(const NzaeNumericField &other)

const Add.

virtual NzaeNumericField* ceil()

const Gets the ceiling.

virtual int32_t cmp(const NzaeNumericField &other)

const Compare.

virtual NzaeNumericField* div(const NzaeNumericField &other) **const**

Divide.

virtual NzaeNumericField* exp()

const Gets the exponent.

virtual NzaeNumericField* floor()

const Gets the floor.

virtual int32_t getsign()

const Gets the sign.

virtual NzaeNumericField* ln()

const Gets the natural Log.

virtual NzaeNumericField* log()

const Get the base 10 log.

virtual NzaeNumericField* log(const NzaeNumericField &base)

const Gets the Log.

virtual NzaeNumericField* mod(const NzaeNumericField &other)

const Gets the modulus.

virtual NzaeNumericField* mul(const NzaeNumericField &other)

const Multiply.

NzaeNumericField()

Constructs a numeric field with precision and scale of 0.

operator double() const

Returns the value, converted to a double.

bool operator!=(const NzaeNumericField &x)

const Not Equal.

NzaeNumericField& operator%=(const NzaeNumericField &x) Assignment by modulo.

NzaeNumericField& operator*=(const NzaeNumericField &x) Assignment by multiplication.

NzaeNumericField&

operator++() Increment.

NzaeNumericField& operator+=(const NzaeNumericField &x) Assignment by addition.

NzaeNumericField& operator--

() Decrement.

NzaeNumericField& operator-=(const NzaeNumericField &x) Assignment by subtraction.

NzaeNumericField& operator/=(const NzaeNumericField &x) Assignment by division.

bool operator<(const NzaeNumericField &x)

const Less than.

bool operator<=(const NzaeNumericField &x) const

Less than or equal.

NzaeNumericField& operator=(int64_t val) Assigns the value of the argument to a field object.

NzaeNumericField& operator=(int32_t val) Assigns the value of the argument to a field object.

NzaeNumericField& operator=(const NzaeNumericField &val)
Assigns the value of the argument to a field object. The field argument may be a different type, as long as it is compatible.

NzaeNumericField& operator=(double val) Assigns the value of the argument to a field object.

bool operator==(const NzaeNumericField &x)
const Equal to.

bool operator>(const NzaeNumericField &x)
const Greater than.

bool operator>=(const NzaeNumericField &x)
const Greater than or equal.

virtual NzaeNumericField* power(const NzaeNumericField &exponent)
const Raise to a power.

int precision() const
Returns the precision.

virtual NzaeNumericField* round(int scale=0)
const Rounds the value.

int scale() const
Returns the scale.

void setPrecision(int
prec) Sets the precision.

void setScale(int
scale) Sets the scale.

virtual NzaeNumericField* sqrt()
const Gets the square root.

virtual NzaeNumericField* sub(const NzaeNumericField &other)
const Subtract.

virtual NzaeNumeric128Field* toNumeric128(int precision, int scale)
const Constructs a NzaeNumeric128Field from the current field.

virtual NzaeNumeric32Field* toNumeric32(int precision, int scale)
const Constructs a NzaeNumeric32Field from the current field.

virtual NzaeNumeric64Field* toNumeric64(int precision, int scale)
const Constructs a NzaeNumeric64Field from the current field.

virtual NzaeNumericField* trunc(int scale=0)
 const Truncates the value.

virtual NzaeNumericField* uminus()
 const Unary minus.

virtual NzaeNumericField* uplus()
 const Unary plus.

virtual ~NzaeNumericField()

Static Public Member Functions

static NzaeNumericField* newField(std::string str) Constructs a NumericField from string.

static NzaeNumericField* newField(int32_t val) Constructs a NumericField from int32_t.

static NzaeNumericField* newField(int64_t val) Constructs a NumericField from int64_t.

static NzaeNumericField* newField(double val) Constructs a NumericField from double.

Detailed Description

This class provides a common base class for the NzaeNumeric32Field , NzaeNumeric64Field , and NzaeNumeric128Field field classes.

See Also

NzaeNumeric32Field
 NzaeNumeric64Field
 NzaeNumeric128Field
 NzaeField

Public Member Function Documentation

virtual NzaeNumericField* abs()

const Gets the absolute value.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* add(const NzaeNumericField &other)

const Add.

▲ Parameters

► **NzaeNumericField** *other*

The field to add by.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

► NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* ceil() const

Gets the ceiling.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

► NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual int32_t cmp(const NzaeNumericField &other)

const Compare.

Parameters

NzaeNumericField other

The field to compare.

Returns

Value of 0 if equal, -1 if one field is less than the other, 1 if one field is greater than the other.

Exceptions

NzaeException

virtual NzaeNumericField* div(const NzaeNumericField &other)

const Divide.

Parameters

NzaeNumericField other

The field to divide by.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* exp()

const Gets the exponent.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

► NzaeException

Returns the value of e (the base of natural logarithms) raised to the power of the value of object.
Re-returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* floor()

const Gets the floor.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual int32_t getsign()

const Gets the sign.

Returns

A value of 0 if the value is 0, -1 if it is negative, 1 it is if positive.

Returns the sign of the value.

virtual NzaeNumericField* ln()

const Gets the natural Log.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* log()

const Get the base 10 log.

Returns

NzaeNumericField

The new NzeNumeric128Field object.

Exceptions

NzeException

Returns one of the three NzeNumericField-derived classes based on the field size.

virtual NzeNumericField* log(const NzeNumericField &base)

const Gets the Log.

Parameters

NzeNumericField base

Numeric Field base of the log.

Returns

NzeNumericField

The new NzeNumericField object.

Exceptions

NzeException

Returns one of the three NzeNumericField-derived classes based on the field size.

virtual NzeNumericField* mod(const NzeNumericField &other)

const Gets the modulus.

Parameters

NzeNumericField other

Field to modulus by.

Returns

NzeNumericField

The new NzeNumericField object.

Exceptions

NzeException

Returns one of the three NzeNumericField-derived classes based on the field size.

virtual NzeNumericField* mul(const NzeNumericField &other)

const Multiply.

Parameters

NzeNumericField other

Field to multiply by.

Returns

NzeNumericField

The new NzeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

NzaeNumericField()

Constructs a numeric field with precision and scale of 0.

operator double() const

Returns the value, converted to a double.

Returns

The converted double value.

bool operator!=(const NzaeNumericField &x)

const Not Equal.

Parameters

NzaeNumericField x

The field to compare.

Returns

TRUE if the field is not equal to x.

Exceptions

NzaeException

NzaeNumericField& operator%=(const NzaeNumericField &x) Assignment by modulo.

Parameters

NzaeNumericField x

The field to modulus into the current field.

Returns

NzaeNumericField

Exceptions

NzaeException

NzaeNumericField& operator*=(const NzaeNumericField &x) Assignment by multiplication.

Parameters

NzaeNumericField x

The field to multiply into the current field.

Returns

NzaeNumericField

Exceptions

NzaeException

NzaeNumericField&
operator++() Increment.

Returns

NzaeNumericField

Exceptions

NzaeException

**NzaeNumericField& operator+=(const NzaeNumericField
&x)** Assignment by addition.

Parameters

NzaeNumericField x

The field to add into the current field.

Returns

NzaeNumericField

Exceptions

NzaeException

NzaeNumericField& operator--
() Decrement.

Returns

NzaeNumericField

Exceptions

NzaeException

**NzaeNumericField& operator-=(const NzaeNumericField
&x)** Assignment by subtraction.

Parameters

NzaeNumericField x

The field to subtract into the current field.

Returns

NzaeNumericField

Exceptions

NzaeException

**NzaeNumericField& operator/=(const NzaeNumericField
&x)** Assignment by division.

Parameters

NzaeNumericField x

The field to divide into the current field.

Returns

NzeNumericField

Exceptions

NzeException

bool operator<(const NzeNumericField &x)**const** Less than.

Parameters

NzeNumericField x

The field to compare.

Returns

TRUE if the field is less than x.

Exceptions

NzeException

bool operator<=(const NzeNumericField &x)**const** Less than or equal.

Parameters

NzeNumericField x

The field to compare.

Returns

TRUE if the field is less than or equal to x.

Exceptions

NzeException

NzeNumericField& operator=(int64_t val) Assigns

the value of the argument to a field object.

Parameters

val

The int64_t value to assign.

Returns

NzeNumericField**NzeNumericField& operator=(int32_t val)** Assigns

the value of the argument to a field object.

Parameters

val

The int32_t value to assign.

Returns

NzeNumericField**NzeNumericField& operator=(const NzeNumericField &val)**

Assigns the value of the argument to a field object. The field argument may be a different type, as long as it is compatible.

Parameters

NzaeNumericField val

The field to assign.

Returns

NzaeNumericField

NzaeNumericField& operator=(double val) Assigns the value of the argument to a field object.

Parameters

val

The double value to assign.

Returns

NzaeNumericField

bool operator==(const NzaeNumericField &x)
const Equal to.

Parameters

NzaeNumericField x

The field to compare.

Returns

TRUE if the field is equal to x.

Exceptions

NzaeException

bool operator>(const NzaeNumericField &x)
const Greater than.

Parameters

NzaeNumericField x

The field to compare.

Returns

TRUE if the field is greater than x.

Exceptions

NzaeException

bool operator>=(const NzaeNumericField &x)
const Greater than or equal.

Parameters

NzaeNumericField x

The field to compare with.

Returns

TRUE if the field is greater than or equal to x.

Exceptions

NzaeException

virtual NzaeNumericField* power(const NzaeNumericField &exponent)

const Raise to a power.

Parameters

NzaeNumericField exponent

The power to raise field by.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

int precision() const

Returns the precision.

Returns

The precision.

virtual NzaeNumericField* round(int scale=0)

const Rounds the value.

Parameters

scale

The number of integer places to the right of decimal point.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

int scale() const

Returns the scale.

Returns

The scale.

void setPrecision(int prec) Sets the precision.

Parameters

prec
The precision.

void setScale(int scale)
Sets the scale.

Parameters

scale The
scale.

virtual NzaeNumericField* sqrt()
const Gets the square root.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* sub(const NzaeNumericField &other)
const Subtract.

Parameters

NzaeNumericField other
The field to subtract by.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumeric128Field* toNumeric128(int precision, int scale)
const Constructs a NzaeNumeric128Field from the current field.

▲ Parameters

► **precision**

The desired precision.

scale

The desired scale.

Returns

NzaeNumeric128Field

The new NzaeNumeric128Field object.

Exceptions

NzaeException

Uses the specified precision and scale for the new field.

virtual NzaeNumeric32Field* toNumeric32(int precision, int scale)

const Constructs a NzaeNumeric32Field from the current field.

Parameters

precision

The desired precision.

scale

The desired scale.

Returns

NzaeNumeric32Field

The new NzaeNumeric32Field object.

Exceptions

NzaeException

Uses the specified precision and scale for the new field.

virtual NzaeNumeric64Field* toNumeric64(int precision, int scale)

const Constructs a NzaeNumeric64Field from the current field.

Parameters

precision

The desired precision.

scale

The desired scale.

Returns

NzaeNumeric64Field

The new NzaeNumeric64Field object.

Exceptions

NzaeException

Uses the specified precision and scale for the new field.

virtual NzaeNumericField* trunc(int scale=0)

const Truncates the value.

Parameters

scale

The number of decimal places to truncate to.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* uminus()

const Unary minus.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

► NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual NzaeNumericField* uplus()

const Unary plus.

Returns

NzaeNumericField

The new NzaeNumericField object.

Exceptions

► NzaeException

Returns one of the three NzaeNumericField-derived classes based on the field size.

virtual ~NzaeNumericField()

Static Public Member Function Documentation

static NzaeNumericField* newField(std::string

str) Constructs a NumericField from string.

Parameters

str

The string to construct from.

Returns

NzaeNumericField

The new NumericField object.

Returns one of the three NzaeNumericField-derived classes based on the string value.

static NzaeNumericField* newField(int32_t val) Constructs a NumericField from int32_t.

Parameters

val

The int32_t to construct from.

Returns

NzaeNumericField

The new NumericField object.

Returns one of the three NzaeNumericField-derived classes based on the int32_t value.

static NzaeNumericField* newField(int64_t val) Constructs a NumericField from int64_t.

Parameters

val

The int64_t to construct from.

Returns

NzaeNumericField

The new NumericField object.

Returns one of the three NzaeNumericField-derived classes based on the int64_t value.

static NzaeNumericField* newField(double val) Constructs a NumericField from double.

Parameters

val

The double to construct from.

Returns

NzaeNumericField

The new NumericField object.

Returns one of the three NzaeNumericField-derived classes based on the double value.

NzaeParameters Class Reference

This class provides access to AE Parameters.

Public Member Functions

```
virtual void addEntry(std::string name)=0
virtual const char* getParameter(int idx) const
=0 Gets the parameter value.

virtual void setReadOnly()=0
virtual int size() const =0
Gets the number of parameters.

virtual ~NzaeParameters()
```

Static Public Member Functions

```
static NzaeParameters* create()
```

Detailed Description

This class provides access to AE Parameters.

See Also

- NzaeFunction
- NzaeAggregate
- NzaeShaper

Public Member Function Documentation

virtual void addEntry(std::string name)=0

virtual const char* getParameter(int idx) const
=0 Gets the parameter value.

Parameters

idx

The index to look up.

Returns

The value or NULL. Does not need to be deleted.

Exceptions

NzaeException

virtual void setReadOnly()=0

virtual int size() const =0

Gets the number of parameters.

Returns

The number of parameters.

virtual ~NzaeParameters()

Static Public Member Function Documentation

static NzaeParameters* create()

Returns

NzaeParameters

NzaeRecord Class Reference

This class provides an AE record.

Public Member Functions

NzaeField* AddColumn(NzaeDataTypes::Types type)

**NzaeField& get(int
idx)** Gets the field.

int numFields() const Gets
the number of fields.

NzaeRecord()

void setShapeReadOnly()

virtual ~NzaeRecord()

Detailed Description

This class provides an AE record.

A record is a group of NzaeField objects

See Also

NzaeShaper

NzaeAggregate

NzaeFunction

NzaeField

Public Member Function Documentation

NzaeField* AddColumn(NzaeDataTypes::Types type)

Returns

NzaeField

**NzaeField& get(int
idx)** Gets the field.

Parameters

► **idx**

The index to look up.

Returns

NzaeField

The field.

Exceptions

► **NzaeException**

int numFields() const Gets the number of fields.

Returns

The number of fields.

NzaeRecord()

void setShapeReadOnly()

virtual ~NzaeRecord()

NzaeRemoteProtocol Class Reference

Class to get an API object in Remote Mode.

Public Member Functions

virtual NzaeApi* acceptConnection()=0

Accepts a new connection.

virtual NzaeApi* acceptConnectionFork()=0

Accepts a new connection and fork.

virtual NzaeApi* acceptConnectionWithTimeout(int timeoutMilliseconds)=0 Accepts a new connection with timeout.

virtual NzaeApi* acceptConnectionWithTimeoutFork(int timeoutMilliseconds)=0 Accepts a new connection and fork with timeout.

virtual void close()=0

Closes the listener.

virtual NzaeRemoteProtocolCallback* getCallbackHandler()=0

Gets the remote protocol callback handler.

virtual void setCallbackHandler(NzaeRemoteProtocolCallback *handler)=0 Sets the remote protocol callback handler.

virtual ~NzaeRemoteProtocol()

Detailed Description

Class to get an API object in Remote Mode.

See Also

NzaeApi

NzaeRemoteProtocolCallback

Public Member Function Documentation

virtual NzaeApi* acceptConnection()=0

Accepts a new connection.

Returns

NzaeApi

The new API object.

This object must be deleted when complete.

See Also

► NzaeApi

virtual NzaeApi* acceptConnectionFork()=0

Accepts a new connection and fork.

Returns

NzaeApi

The new API object or NULL.

This object must be deleted when complete. Returns NULL in the parent and non-NULL in the new child.

The new child is in a new process group.

See Also

NzaeApi

virtual NzaeApi* acceptConnectionWithTimeout(int timeoutMilliseconds)=0 Accepts a new connection with timeout.

Parameters

timeoutMilliseconds

The timeout value in milliseconds.

Returns

NzaeApi

The new API object or NULL if timeout.

This object must be deleted when complete.

▲ See Also ►

NzaeApi

virtual NzaeApi* acceptConnectionWithTimeoutFork(int timeoutMilliseconds)=0

Accepts a new connection and fork with timeout.

Parameters

timeoutMilliseconds

The timeout value in milliseconds

Returns

NzaeApi

new The API object or NULL.

This object must be deleted when complete. Returns NULL in the parent and non-NULL in the new child. The new child is in a new process group.

See Also

► [NzaeApi](#)

virtual void close()=0

Closes the listener.

virtual NzaeRemoteProtocolCallback* getCallbackHandler()=0

Gets the remote protocol callback handler.

Returns

NzaeRemoteProtocolCallback

The callback handler.

A remote protocol handler class is used to handle remote commands such as stop, status, and ping.

See Also

[NzaeRemoteProtocolCallback](#)

**virtual void setCallbackHandler(NzaeRemoteProtocolCallback
*handler)=0** Sets the remote protocol callback handler.

Parameters

NzaeRemoteProtocolCallback

handler The remote protocol handler.

A remote protocol handler class is used to handle remote commands such as stop, status, and ping.

▲ See Also

► [NzaeRemoteProtocolCallback](#)

virtual ~NzaeRemoteProtocol()

NzaeRemoteProtocolCallback Class Reference

Class to handle callbacks for remote protocol mode.

Public Types

```
enum NzaeCallbackType {
    CallbackRequest, CallbackPing, CallbackStatus, CallbackStop, CallbackControl,
    CallbackSignal } Specifies the callback type.
```

Public Member Functions

```
virtual void execute(NzaeCallbackType code, int dataLen, const char *data,
    NzaeCallbackResult *result)=0
    The callback executor method.
virtual ~NzaeRemoteProtocolCallback()
```

Detailed Description

Class to handle callbacks for remote protocol mode.

They can be used to get status, stop or ping remote AEs.

Enumeration Type Documentation

```
enum NzaeCallbackType
    Specifies the callback type.
```

CallbackRequest

CallbackPing

CallbackStatus

CallbackStop

CallbackControl

CallbackSignal

Public Member Function Documentation

```
virtual void execute(NzaeCallbackType code, int dataLen, const char *data,
    NzaeCallbackResult *result)=0
    The callback executor method.
```

Parameters

NzaeCallbackType

code The callback type.

dataLen

The data length.

data The data.

NzaeCallbackResult result

The callback result data structure.

This method handles the following types: CallbackStatus, CallbackStop, CallbackControl, Call-backSignal.

If this method throws an exception, it causes the remote protocol accept method to error out. The values of dataLen and data are likely to be empty for Stop and Status.

The executor should fill out the result structure with: returnCode equal 0 for normal completion; dataLength equal to length of returned data; data equal to the data which should have been allocated with malloc; bFreeData set to be true if data and dataLength are not empty.

virtual ~NzaeRemoteProtocolCallback()

NzaeRuntime Class Reference

This class provides Runtime functionality.

Public Types

```
enum AdapterType {  
    NZAE_ADAPTER_OTHER= 0, NZAE_ADAPTER_UDTF= 1, NZAE_ADAPTER_UDF= 2,  
    NZAE_ADAPTER_UDA= 3 }  
enum LocusType {  
    NZAE_LOCUS_POSTGRES= 0, NZAE_LOCUS_DBOS= 1, NZAE_LOCUS_SPU=  
    2 } Specifies which locus the AE is executing in.
```

Specifies the AE's function type.

```
enum LocusType {  
    NZAE_LOCUS_POSTGRES= 0, NZAE_LOCUS_DBOS= 1, NZAE_LOCUS_SPU=  
    2 } Specifies which locus the AE is executing in.
```

Public Member Functions

AdapterType getAdapterType()
const Gets the adapter type.

int64_t getAeCallId()
const Gets the call ID.

int64_t getAeQueryId()
const Gets the query ID.

int getDataSliceId() const
Gets the dataslice ID.

int getHardwareId() const
Gets the hardware ID.

LocusType getLocus()
 const Gets the locus.

 int getNumberDataSlices() const
 Gets the number of dataslices.

 int getNumberSpus() const
 Gets the number of SPUs.

 int getSessionId() const
 Gets the session ID.

 int64_t getSuggestedMemoryLimit()
 const Gets the memory limit.

 int64_t getTransactionId()
 const Gets the transaction ID.

 std::string getUsername() const
 Gets the database user name.

 bool getUserQuery() const
 Determines if this is a user query.

Public Attributes

adapterType
 aeCallId
 aeQueryId
 dataSliceId
 hardwareId
 locus
 numberDataSlices
 numberSpus
 sessionId
 suggestedMemoryLimit
 transactionId
 userName
 userQuery

Detailed Description

This class provides Runtime functionality.

This class provides access to information common to all AEs about the runtime in which it was invoked.

See Also

NzaeFunction
 NzaeAggregate
 NzaeShaper

Enumeration Type Documentation

enum AdapterType

Specifies the AE's function type.

NZAE_ADAPTER_OTHER

NZAE_ADAPTER_UDTF

NZAE_ADAPTER_UDF

NZAE_ADAPTER_UDA

enum LocusType

Specifies which locus the AE is executing in.

NZAE_LOCUS_POSTGRES

NZAE_LOCUS_DBOS

NZAE_LOCUS_SPU

Public Member Function Documentation

AdapterType getAdapterType()

const Gets the adapter type.

Returns

AdapterType The

adapter type.

int64_t getAeCallId()

const Gets the call ID.

Returns

The call ID.

int64_t getAeQueryId()

const Gets the query ID.

Returns

The query ID.

int getDataSliceId() const

Gets the dataslice ID.

Returns

The dataslice ID.

int getHardwareId() const

Gets the hardware ID.

Returns
The hardware ID.

LocusType getLocus()
const Gets the locus.

Returns
LocusType
The locus of execution.

int getNumberDataSlices() const
Gets the number of dataslices.

Returns
The number of dataslices.

int getNumberSpus() const
Gets the number of SPUs.

Returns
The number of SPUs.

int getSessionId() const
Gets the session ID.

Returns
The session ID.

int64_t getSuggestedMemoryLimit()
const Gets the memory limit.

Returns
The memory limit.

This is an advisory limit only.

int64_t getTransactionId()
const Gets the transaction ID.

Returns
The transaction ID.

std::string getUsername() const
Gets the database user name.

Returns

The database user name.

bool getUserQuery() const

Determines if this is a user query.

Returns

TRUE if a user query as opposed to a JIT state or other prep query.

Member Data Documentation

AdapterType adapterType

int64_t aeCallId

int64_t aeQueryId

int dataSliceId

int hardwareId

LocusType locus

int numberDataSlices

int numberSpus

int sessionId

int64_t suggestedMemoryLimit

int64_t transactionId

std::string userName

bool userQuery

NzaeShaper Class Reference

This class provides Shaper or Sizer functionality.

Public Types

```
enum LogLevel {
    LOG_TRACE=1, LOG_DEBUG=2
} Log Level.
```

Public Member Functions

```
virtual void addOutputColumn(NzaeDataTypes::Types type, const char
*columnName)=0 Adds a non-string and non-numeric column.

virtual void addOutputColumnNumeric(NzaeDataTypes::Types type, const char *columnName, int
precision, int scale)=0
Adds a numeric column.

virtual void addOutputColumnString(NzaeDataTypes::Types type, const char *columnName, int
size)=0 Adds a string column.

virtual bool catalogIsUpper() const =0
Determines if the catalog is in upper case.

virtual void close()=0
Closes the AE and releases its resources.

virtual const NzaeEnvironment& getEnvironment() const
=0 Gets the environment information for the AE.

virtual const NzaeLibrary& getLibrary() const
=0 Gets library information about the AE.

virtual NzaeShaperMessageHandler& getMessageHandler() const
=0 Returns the message handler class object.

virtual const NzaeMetadata& getMetadata() const =0
Gets metadata about the AE, including the input and output columns.

virtual int getNumOutputColumns() const
=0 Gets number of output columns.

virtual const NzaeShaperOutputColumnInfo& getOutputColumnInfo(int idx) const
=0 Gets output column information.

virtual const NzaeParameters& getParameters() const
=0 Gets parameter information for the AE.

virtual const NzaeRuntime& getRuntime() const =0
Gets runtime information for the AE, including information about the Netezza system.

virtual const NzaeRecord& inputRow() const
=0 Gets the input row.

virtual void log(LogLevel logLevel, const char *message) const
=0 Logs the specified message at the specified log level.

virtual std::string logFileName() const
=0 Returns the log file name.

virtual NzaeDataTypes::Types outputType() const =0
```

Returns the UDF return type.

virtual void ping() const =0

Indicates that the AE is still active and not hanging.

virtual void run(NzaeShaperMessageHandler
*messageHandler)=0 Runs the shaper handler.

virtual void update()=0

Indicates that the shaper is done.

virtual void userError(const char *message) const =0

Indicates the AE has encountered an error condition.

virtual ~NzaeShaper()

Static Public Member Functions

static NzaeShaper* newInstance(NzaeShaperInitialization &arg, NZAESHP_HANDLE handle)

Detailed Description

This class provides Shaper or Sizer functionality.

This class is used to implement Scalar or Table function Sizer or Shaper functionality.

See Also

NzaeShaperMessageHandler

NzaeFactory

NzaeApi

NzaeLibrary

NzaeParameters

NzaeEnvironment

NzaeMetadata

NzaeRecord

Enumeration Type Documentation

enum LogLevel

Log Level.

LOG_TRACE

LOG_DEBUG

Public Member Function Documentation

virtual void addOutputColumn(NzaeDataTypes::Types type, const char
*columnName)=0 Adds a non-string and non-numeric column.

▲ Parameters

► Types type

The column type, which cannot be a string or numeric type.

columnName The
column name.

virtual void addOutputColumnNumeric(NzaeDataTypes::Types type, const char *columnName, int precision, int scale)=0

Adds a numeric column.

Parameters

Types type

The column type, which must be a numeric type.

columnName The
column name.

precision

The column precision.

scale

The column scale.

virtual void addOutputColumnString(NzaeDataTypes::Types type, const char *columnName, int size)=0

Adds a string column.

Parameters

Types type

The column type which must be a string type.

columnName The
column name.

size

The column size.

virtual bool catalogIsUpper() const =0

Determines if the catalog is in upper case.

Returns

TRUE if catalog is upper case.

virtual void close()=0

Closes the AE and releases its resources. Releases

all resources associated with the shaper.

virtual const NzaeEnvironment& getEnvironment() const =0 Gets the environment information for the AE.

Returns

NzaeEnvironment

The instance of NzaeEnvironment .

See Also

NzaeEnvironment

virtual const NzaeLibrary& getLibrary() const

=0 Gets library information about the AE.

Returns

NzaeLibrary

The instance of NzaeLibrary .

See Also

NzaeLibrary

virtual NzaeShaperMessageHandler& getMessageHandler() const

=0 Returns the message handler class object.

Returns

NzaeShaperMessageHandler

The instance of NzaeShaperMessageHandler .

The message handler is where custom function logic is implemented.

See Also

NzaeShaperMessageHandler

virtual const NzaeMetadata& getMetadata() const =0

Gets metadata about the AE, including the input and output columns.

Returns

NzaeMetadata

The instance of NzaeMetadata .

See Also

NzaeMetadata

virtual int getNumOutputColumns() const

=0 Gets number of output columns.

Returns

The number of output columns.

virtual const NzaeShaperOutputColumnInfo& getOutputColumnInfo(int idx) const

=0 Gets output column information.

Parameters

idx

The index of the column to get.

Returns

NzaeShaperOutputColumnInfo

The column information.

Exceptions

NzaeException

virtual const NzaeParameters& getParameters() const**=0** Gets parameter information for the AE.

Returns

NzaeParameters

The instance of NzaeParameters .

See Also

NzaeParameters

virtual const NzaeRuntime& getRuntime() const =0

Gets runtime information for the AE, including information about the Netezza system.

Returns

NzaeRuntime

The instance of NzaeRuntime .

See Also

NzaeRuntime

virtual const NzaeRecord& inputRow() const**=0** Gets the input row.

Returns

NzaeRecord

An instance of NzaeRecord .

All non-literal fields are NULL.

See Also

NzaeRecord

virtual void log(LogLevel logLevel, const char *message) const**=0** Logs the specified message at the specified log level.

Parameters

LogLevel logLevel The log level constant.**message**

The message to log.

virtual std::string logFileName() const

=0 Returns the log file name.

Returns

The log file name.

virtual NzaeDataTypes::Types outputType() const

=0 Returns the UDF return type.

Returns

Types

The return type.

Gets the return type for a sizer (UDF). The value can only be one of the string types or NUMERIC128.

virtual void ping() const =0

Indicates that the AE is still active and not hanging.

virtual void run(NzaeShaperMessageHandler

***messageHandler)=0** Runs the shaper handler.

Parameters

NzaeShaperMessageHandler messageHandler

The message handler. The message handler is where custom function logic is implemented.

This function is an alternative to writing custom shaper code.

See Also

NzaeShaperMessageHandler

virtual void update()=0 Indicates

that the shaper is done.

virtual void userError(const char *message) const =0

Indicates the AE has encountered an error condition.

Parameters

message

The message to send back to the Netezza software.

Implies NzaeDone.

virtual ~NzaeShaper()

Static Public Member Function Documentation

static NzaeShaper* newInstance(NzaeShaperInitialization &arg, NZAESHP_HANDLE handle)

Returns

NzaeShaper

NzaeShaperInitialization Class Reference

Not implemented. This class is a placeholder for future functionality.

Detailed Description

Not implemented. This class is a placeholder for future functionality.

See Also

NzaeFactory

NzaeShaper

NzaeApi

NzaeShaperMessageHandler Interface Reference

This class provides higher level shaper implementation.

Public Member Functions

virtual void shaper(NzaeShaper
&api)=0 Sets up the output shape.

virtual ~NzaeShaperMessageHandler()

Detailed Description

This class provides higher level shaper implementation.

Implement this class to handle NzaeShaper messages.

See Also

▲ run

Public Member Function Documentation

**virtual void shaper(NzaeShaper
&api)=0** Sets up the output shape.

Parameters

NzaeShaper api

The shaper object.

When the handler style is used, the framework handles exceptions and calling updates.

- ▲ See Also
 - NzaeShaper

virtual ~NzaeShaperMessageHandler()

NzaeShaperOutputColumn Class Reference

This class provides Shaper output information.

Detailed Description

This class provides Shaper output information.

This class is used for filling in the output information for the shaper.

- See Also
- ▲ NzaeShaper

NzaeShaperOutputColumnInfo Class Reference

Public Attributes

m_columnName
The column name.

m_precision
The precision, if numeric.

m_scale
The scale, if numeric.

m_size
The size, if string.

m_type
Type.

Member Data Documentation

std::string m_columnName
The column name.

int m_precision

The precision, if numeric.

int m_scale

The scale, if numeric.

int m_size

The size, if string.

NzaeDataTypes::Types

m_type Type.

NzaeStringField Class Reference

This class provides a common base class for the NzaeFixedStringField , NzaeVariableStringField , NzaeNationalFixedStringField , NzaeNationalVariableStringField , NzaeGeometryStringField and NzaeVarbinaryStringField classes.

Inherits NzaeField

Public Member Functions

void fromString(std::string str)

Constructs the field from the string.

virtual int length() const

=0 Gets the string length.

NzaeStringField(std::string str)

Constructs a string field with value str.

NzaeStringField(NzaeStringField &field)

Constructs a string field with value field.

NzaeStringField()

Constructs a NULL string field.

operator std::string &()

Returns the string value.

NzaeStringField& operator=(NzaeField &field) Assigns the value of the argument to the field object.

NzaeStringField& operator=(NzaeStringField &field)

Assigns the value of the argument to the field object.

NzaeStringField& operator=(std::string str)

Assigns the value of the argument to the field object.

std::string toString() const

Returns the string representation of field.

`virtual NzaeDataTypes::Types type() const`
=0 Returns the type of the field.

Detailed Description

This class provides a common base class for the `NzaeFixedStringField` , `NzaeVariableStringField` , `NzaeNationalFixedStringField` , `NzaeNationalVariableStringField` , `NzaeGeometryStringField` and `NzaeVarbinaryStringField` classes.

See Also

- `NzaeField`
- `NzaeFixedStringField`
- `NzaeVariableStringField`
- `NzaeNationalFixedStringField`
- `NzaeNationalVariableStringField`
- `NzaeGeometryStringField`
- `NzaeVarbinaryStringField`

Public Member Function Documentation

`void fromString(std::string str)`

Constructs the field from the string.

Parameters

`str`

The string to assign from.

`virtual int length() const`

=0 Gets the string length.

Returns

The string length in bytes for non-national, char for national.

`NzaeStringField(std::string str)`

Constructs a string field with value str.

Parameters

`str`

The value.

`NzaeStringField(NzaeStringField &field)`

Constructs a string field with value field.

Parameters

`NzaeStringField`

`field` The field name.

The field argument may be a different type.

NzaeStringField()

Constructs a NULL string field.

operator std::string &()

Returns the string value.

Returns
The string value.

NzaeStringField& operator=(NzaeField &field)

Assigns the value of the argument to the field object.

Parameters
NzaeField field
The field to assign.

Returns
NzaeStringField

The field argument may be a different type.

NzaeStringField& operator=(NzaeStringField &field)

Assigns the value of the argument to the field object.

Parameters
NzaeStringField field
The field to assign.

Returns
NzaeStringField

The field argument may be a different type.

NzaeStringField& operator=(std::string str)

Assigns the value of the argument to the field object.

Parameters
str
The value to assign.

Returns
NzaeStringField

std::string toString() const

Returns the string representation of field.

Returns

The string representation.

virtual NzaeDataTypes::Types type() const

=0 Returns the type of the field.

Returns

Types

The field type.

NzaeTimeField Class Reference

This class provides field access for type time.

Inherits NzaeField

Public Member Functions

NzaeTimeField addInterval(const NzaeIntervalField &x)

const Constructs a TimeField by adding an interval.

void decodeTime(uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrcs, bool *errorFlag=NULL) const

Converts a Netezza-encoded Time value to h:m:s:micros.

void encodeTime(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrcs, bool *errorFlag=NULL)

Converts a h:m:s:micros Time value to a Netezza-encoded Time.

void fromString(std::string str)

Constructs the field from the string.

bool isValidTime() const

Specifies whether a Netezza-encoded Time value is valid and within range.

NzaeTimeField(const NzaeTimeField &field)

Constructs a time field with value field.

NzaeTimeField(const NzaeTimeTzField &field)

Constructs a time field with value field.

NzaeTimeField()

Constructs a NULL time field.

NzaeTimeField(const NzaeTimestampField &field) Constructs a time field with value field.

NzaeTimeField(int64_t val) Constructs a time field with value val.

void offsetTime(int32_t sqlOffset, bool *errorFlag=NULL)

Applies an offset to the Netezza Time. If nzTime with offset runs over 23:59:59.999999, it

'wraps around' back at zero. For example, applying '+120 minutes' to the encoded equivalent of '23:00:00' returns the encoded equivalent of '01:00:00'.

operator int64_t() const Returns
the encoded field value.

operator NzaeIntervalField() const
Returns the interval field value.

operator NzaeTimeTzField() const
Returns the timetz field value.

NzaeTimeField& operator=(const NzaeTimestampField
&field) Assigns the value of the argument to a field object.

NzaeTimeField& operator=(NzaeField &field)
Assigns the value of the argument to a field object.

NzaeTimeField& operator=(const NzaeTimeTzField &field)
Assigns the value of the argument to a field object.

NzaeTimeField& operator=(int64_t val)
Assigns the value of the argument to a field object.

NzaeTimeField& operator=(const NzaeTimeField &field)
Assigns the value of the argument to a field object.

NzaeTimeField subInterval(const NzaeIntervalField &x)
const Constructs a TimeField by subtracting interval.

NzaeIntervalField subTime(const NzaeTimeField &x)
const Constructs an IntervalField by subtracting time.

std::string toString() const
Returns the string representation of the field.

virtual NzaeDataTypes::Types type()
const Returns the type of the field.

Static Public Member Functions

static bool isValidTime(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs) Determines
whether a decoded h:m:s:micros Time value is valid and within the Netezza Time range.

static int64_t max()
Gets the encoded max.

static int64_t min()
Gets the encoded min.

Detailed Description

This class provides field access for type time.

See Also

▲ NzaeField

Public Member Function Documentation

NzaeTimeField addInterval(const NzaeIntervalField &x)

const Constructs a TimeField by adding an interval.

Parameters

NzaeIntervalField x

The NzaeIntervalField value.

Returns

NzaeTimeField

The TimeField consisting of Interval plus Time.

See Also

NzaeIntervalField

void decodeTime(uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrs, bool *errorFlag=NULL) const

Converts a Netezza-encoded Time value to h:m:s:micros.

Parameters

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTime(encodedTime) is FALSE; *set to FALSE other-wise.

Exceptions

NzaeException

void encodeTime(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs, bool *errorFlag=NULL)

Converts a h:m:s:micros Time value to a Netezza-encoded Time.

Parameters

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTime(hour,minute,second,mcrs) is FALSE; *set to FALSE other-wise.

Exceptions

NzaeException

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

bool isValidTime() const

Specifies whether a Netezza-encoded Time value is valid and within range.

Returns

FALSE if encodedTime<ENC_TIME_MIN, or encodedTime>ENC_TIME_MAX. TRUE otherwise.

NzaeTimeField(const NzaeTimeField &field)

Constructs a time field with value field.

Parameters

NzaeTimeField field

The NzaeTimeField value.

NzaeTimeField(const NzaeTimeTzField &field)

Constructs a time field with value field.

Parameters

NzaeTimeTzField field The

NzaeTimeTzField value.

NzaeTimeField()

Constructs a NULL time field.

NzaeTimeField(const NzaeTimestampField &field) Constructs a time field with value field.

Parameters

NzaeTimestampField field The

NzaeTimestampField value.

NzaeTimeField(int64_t val)

Constructs a time field with value val.

Parameters

val

The encoded time value.

void offsetTime(int32_t sqlOffset, bool *errorFlag=NULL)

Applies an offset to the Netezza Time. If nzTime with offset runs over 23:59:59.999999, it 'wraps around' back at zero. For example, applying '+120 minutes' to the encoded equivalent of '23:00:00' returns the encoded equivalent of '01:00:00'.

Parameters

sqlOffset

The time offset, in minutes, SQL_OFFSET_MIN to SQL_OFFSET_MAX inclusive.

errorFlag

If not NULL, *set to TRUE if isValidSqlOffset(sqlOffset) is FALSE or isValidTime(nzTime) is FALSE; FALSE otherwise.

Exceptions

NzaeException

operator int64_t() const Returns

the encoded field value.

Returns

The encoded value.

operator NzaeIntervalField() const

Returns the interval field value.

Returns

The timestamp value converted from time.

See Also

NzaeIntervalField

operator NzaeTimeTzField() const

Returns the timetz field value.

Returns

The timestamp value converted from time.

See Also

NzaeTimeTzField

NzaeTimeField& operator=(const NzaeTimestampField &field) Assigns the value of the argument to a field object.

Parameters

NzaeTimestampField
field The field to assign.

Returns

NzaeTimeField

See Also

NzaeTimestampField

NzaeTimeField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field
 The field to assign.

Returns

NzaeTimeField

The field argument may be a different type, so long as it is compatible.

NzaeTimeField& operator=(const NzaeTimeTzField

&field) Assigns the value of the argument to a field object.

Parameters

NzaeTimeTzField field
 The field to assign.

Returns

NzaeTimeField

See Also

NzaeTimeTzField

NzaeTimeField& operator=(int64_t val)

Assigns the value of the argument to a field object.

Parameters

val
 The encoded value to assign.

Returns

NzaeTimeField

NzaeTimeField& operator=(const NzaeTimeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeTimeField field

The field to assign.

Returns

NzaeTimeField

NzaeTimeField subInterval(const NzaeIntervalField &x)

const Constructs a TimeField by subtracting interval.

Parameters

NzaeIntervalField x

The NzaeIntervalField value.

Returns

NzaeTimeField

The TimeField, consisting of Time minus interval.

See Also

NzaeIntervalField

NzaeIntervalField subTime(const NzaeTimeField &x)

const Constructs an IntervalField by subtracting time.

Parameters

NzaeTimeField x

The NzaeTimeField value.

Returns

NzaeIntervalField

The IntervalField, consisting of Time minus Time.

See Also

NzaeIntervalField

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

Static Public Member Function Documentation

static bool isValidTime(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs) Determines whether a decoded h:m:s:micros Time value is valid and within the Netezza Time range.

Parameters

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

Returns

FALSE if hour>23 or minute>59 or second>59 or mcrs>999,999. TRUE otherwise.

static int64_t max()

Gets the encoded max.

Returns

The encoded max.

static int64_t min()

Gets the encoded min.

Returns

The encoded min.

NzaeTimestampField Class Reference

This class provides field access for type timestamp.

Inherits NzaeField

Public Member Functions

NzaeTimestampField addInterval(const NzaeIntervalField &interval)

const Constructs a TimestampField by adding an interval.

NzaeIntervalField age(const NzaeTimestampField &x) const

Constructs an IntervalField by subtracting a timestamp.

void decodeTimestamp(uint8_t *month, uint8_t *day, uint16_t *year, uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrs, bool *errorFlag=NULL) const

Converts a Netezza-encoded Timestamp value to m/d/y, h:m:s:micros.

`void decodeTimestamp(time_t *result, bool *errorFlag=NULL) const`
 Converts a Netezza-encoded Timestamp value to `time_t`. Drops the microseconds after the last whole minute of the timestamp value.

`void decodeTimestamp(struct timeval *result, bool *errorFlag=NULL)`
 const Converts a Netezza-encoded Timestamp value to `struct timeval`.

`void decodeTimestamp(struct tm *result, bool *errorFlag=NULL) const`
 Converts a Netezza-encoded Timestamp value to `struct tm`. Drops the microseconds after the last whole minute of the timestamp value.

`void encodeTimestamp(uint32_t month, uint32_t day, uint32_t year, uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs, bool *errorFlag=NULL)`
 Converts a m/d/y, h:m:s:micros Timestamp value to a Netezza-encoded Timestamp.

`void encodeTimestamp(time_t ts, bool *errorFlag=NULL)`
 Converts a `time_t` value to a Netezza-encoded Timestamp. Encodes the value in UTC and applies no offsets. Adds 0 microseconds to the encoded value.

`void encodeTimestamp(const struct timeval &ts, bool *errorFlag=NULL)`
 Converts a `struct timeval` value to a Netezza-encoded Timestamp.

`void encodeTimestamp(const struct tm &ts, bool *errorFlag=NULL)`
 Converts a `struct tm` value to a Netezza-encoded Timestamp. Uses only the `ts.tm_year`, `ts.tm_day`, `ts.tm_mon`, `ts.tm_hour`, `ts.tm_min` and `ts.tm_sec` fields of `ts`, ignoring the remaining fields. The value specified for `ts` must pass `isValidTimeStruct()`. Adds 0 microseconds to the encoded value.

`void fromString(std::string str)`
 Constructs a field from the string.

`bool isValidEpochTimestamp() const`
 Determines whether a Netezza-encoded Timestamp value is valid and within the `time_t` Epoch range.

`bool isValidTimestamp() const`
 Determines whether a Netezza-encoded Timestamp value is valid and within range.

`NzaeTimestampField(const NzaeTimestampField &field)` Constructs a timestamp field with value `field`.

`NzaeTimestampField(int64_t val)` Constructs a timestamp field with value `val`.

`NzaeTimestampField(const NzaeDateField &field)`
 Constructs a timestamp field with value `field`.

`NzaeTimestampField()`
 Constructs a NULL timestamp field.

`void offsetTimestamp(int32_t sqlOffset, bool *errorFlag=NULL)` Applies an offset to an NZ Timestamp.

`operator int64_t() const` Returns the encoded field value.

operator NzaeDateField() const

Returns the date field value.

operator NzaeTimeField() const

Returns the time field value.

operator NzaeTimeTzField() const

Returns the timetz field value.

NzaeTimestampField& operator=(const NzaeTimestampField
&field) Assigns the value of the argument to a field object.

NzaeTimestampField& operator=(const NzaeDateField
&field) Assigns the value of the argument to a field object.

NzaeTimestampField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

NzaeTimestampField& operator=(int64_t val)

Assigns the value of the argument to a field object.

NzaeTimestampField subInterval(const NzaeIntervalField &interval)
const Constructs a TimestampField by subtracting an interval.

NzaeIntervalField subTimestamp(const NzaeTimestampField &x)
const Constructs an IntervalField by subtracting a timestamp.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Static Public Member Functions

static int64_t epochEnd()

Gets the encoded epoch end.

static int64_t epochStart()

Gets the encoded epoch start.

static bool isValidTimestamp(uint32_t month, uint32_t day, uint32_t year, uint32_t hour,
uint32_t minute, uint32_t second, uint32_t mcrs)

Determines whether a decoded m/d/y, h:m:s:micros Timestamp value is valid and within the Netezza Timestamp range.

static int64_t max()

Gets the encoded max.

static int64_t min()

Gets the encoded min.

Detailed Description

This class provides field access for type timestamp.

See Also

▲ NzaeField

Public Member Function Documentation

NzaeTimestampField addInterval(const NzaeIntervalField &interval)

const Constructs a TimestampField by adding an interval.

Parameters

NzaeIntervalField interval

The NzaeIntervalField value.

Returns

NzaeTimestampField

The TimestampField, consisting of Interval plus Timestamp.

See Also

NzaeIntervalField

NzaeIntervalField age(const NzaeTimestampField &x) const

Constructs an IntervalField by subtracting a timestamp.

Parameters

NzaeTimestampField x

The NzaeTimestampField value.

Returns

NzaeIntervalField

The IntervalField, consisting of timestamp minus timestamp.

This function returns a more detailed answer than subTimestamp

See Also

NzaeIntervalField

void decodeTimestamp(uint8_t *month, uint8_t *day, uint16_t *year, uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrs, bool *errorFlag=NULL)

const Converts a Netezza-encoded Timestamp value to m/d/y, h:m:s:micros.

Parameters

day

The day count, 1 to 31 inclusive.

month

The month number, 1 to 12 inclusive.

year

The year number, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTimestamp(encodedTimestamp) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void decodeTimestamp(time_t *result, bool *errorFlag=NULL) const

Converts a Netezza-encoded Timestamp value to time_t. Drops the microseconds after the last whole minute of the timestamp value.

Parameters

result

The resulting time_t value. Forced to be signed int32.

errorFlag

If not NULL, *set to TRUE if isValidEpochTimestamp(encodedTimestamp) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void decodeTimestamp(struct timeval *result, bool *errorFlag=NULL)

const Converts a Netezza-encoded Timestamp value to struct timeval.

Parameters

result

The structure where the decoded Timestamp is written.

errorFlag

If not NULL, *set to TRUE if isValidEpochTimestamp(encodedTimestamp) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void decodeTimestamp(struct tm *result, bool *errorFlag=NULL) const

Converts a Netezza-encoded Timestamp value to struct tm. Drops the microseconds after the last whole minute of the timestamp value.

Parameters

result

The structure where the decoded Timestamp is written, such that result->tm_hour, result->tm_min, result->tm_sec, result->tm_year, result->tm_mon, result->tm_mday, result->tm_yday, and result->tm_wday contain the appropriate fields in tm format. Result->tm_isdst is set to -1;

if applicable, all other fields of result are set to 0.

errorFlag

If not NULL, *set to TRUE if isValidTimestamp(encodedTimestamp) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeTimestamp(uint32_t month, uint32_t day, uint32_t year, uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs, bool *errorFlag=NULL)

Converts a m/d/y, h:m:s:micros Timestamp value to a Netezza-encoded Timestamp.

Parameters

year

The year of the date, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

month

The month, 1 to 12 inclusive.

day

The day, 1 to 31 inclusive.

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTimestamp(month, day, year, hour, minute, second, mcrs) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeTimestamp(time_t ts, bool *errorFlag=NULL)

Converts a time_t value to a Netezza-encoded Timestamp. Encodes the value in UTC and applies no offsets. Adds 0 microseconds to the encoded value.

Parameters

ts

The time_t Timestamp value.

errorFlag

If not NULL, *set to TRUE if isValidEpoch(ts) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeTimestamp(const struct timeval &ts, bool *errorFlag=NULL)

Converts a struct timeval value to a Netezza-encoded Timestamp.

Parameters

ts

The struct timeval Timestamp value.

errorFlag

If not NULL, *set to TRUE if isValidTimeVal(ts) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeTimestamp(const struct tm &ts, bool *errorFlag=NULL)

Converts a struct tm value to a Netezza-encoded Timestamp. Uses only the ts.tm_year, ts.tm_day, ts.tm_mon, ts.tm_hour, ts.tm_min and ts.tm_sec fields of ts, ignoring the remaining fields. The value specified for ts must pass isValidTimeStruct(). Adds 0 microseconds to the encoded value.

Parameters

ts

The struct tm Timestamp value.

errorFlag

If not NULL, *set to TRUE if isValidTimeStruct(ts) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void fromString(std::string str)

Constructs a field from the string.

Parameters

str

The string to assign from.

bool isValidEpochTimestamp() const

Determines whether a Netezza-encoded Timestamp value is valid and within the time_t Epoch range.

Returns

FALSE if encodedTimestamp< EPOCH_START_AS_TIMESTAMP or
encodedTimestamp>EPOCH_END_AS_TIMESTAMP; TRUE otherwise.

bool isValidTimestamp() const

Determines whether a Netezza-encoded Timestamp value is valid and within range.

Returns

FALSE if encodedTimestamp< ENC_TIMESTAMP_MIN or
encodedTimestamp>ENC_TIMESTAMP_MAX; TRUE otherwise.

NzaeTimestampField(const NzaeTimestampField &field) Constructs a timestamp field with value field.

Parameters

NzaeTimestampField field The NzaeTimestampField value.

NzaeTimestampField(int64_t val)

Constructs a timestamp field with value val.

Parameters

val
The encoded timestamp value.

NzaeTimestampField(const NzaeDateField &field)

Constructs a timestamp field with value field.

Parameters

NzaeDateField field
The NzaeDateField value.

NzaeTimestampField() Constructs a NULL timestamp field.

void offsetTimestamp(int32_t sqlOffset, bool

***errorFlag=NULL)** Applies an offset to an NZ Timestamp.

Parameters

sqlOffset
The time offset in minutes, SQL_OFFSET_MIN to SQL_OFFSET_MAX inclusive.

errorFlag
If not NULL, *set to TRUE if isValidSqlOffset(sqlOffset) is FALSE, or isValidTimestamp(nzTimestamp) is FALSE or isValidTimestamp(nzTimestamp+sqlOffset*60*1,000,000) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

operator int64_t() const Returns

the encoded field value.

Returns

The encoded value.

operator NzaeDateField() const

Returns the date field value.

Returns

The date value converted from the timestamp

See Also

NzaeDateField

operator NzaeTimeField() const

Returns the time field value.

Returns

The time value converted from the timestamp.

See Also

NzaeTimeField

operator NzaeTimeTzField() const

Returns the timetz field value.

Returns

The timetz value converted from the timestamp.

See Also

NzaeTimeTzField

NzaeTimestampField& operator=(const NzaeTimestampField &field) Assigns the value of the argument to a field object.

Parameters

NzaeTimestampField

field The field to assign.

Returns

NzaeTimestampField

NzaeTimestampField& operator=(const NzaeDateField &field) Assigns the value of the argument to a field object.

Parameters

NzaeDateField field

The field to assign.

Returns

NzaeTimestampField

See Also

NzaeDateField

NzaeTimestampField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeTimestampField

The field argument may be a different type, as long as it is compatible.

NzaeTimestampField& operator=(int64_t val)

Assigns the value of the argument to a field object.

Parameters

val

The encoded value to assign.

Returns

NzaeTimestampField

NzaeTimestampField subInterval(const NzaeIntervalField &interval)

const Constructs a TimestampField by subtracting an interval.

Parameters

NzaeIntervalField interval

The NzaeIntervalField value.

Returns

NzaeTimestampField

The TimestampField, consisting of Timestamp minus interval.

See Also

NzaeIntervalField

NzaeIntervalField subTimestamp(const NzaeTimestampField &x)

const Constructs an IntervalField by subtracting a timestamp.

Parameters

NzaeTimestampField x

The NzaeTimestampField value.

Returns

NzaeIntervalField

The IntervalField, consisting of Timestamp minus Timestamp.

See Also

NzaeIntervalField

std::string toString() const

Returns the string representation of the field.

Returns
The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns
Types
The field type.

Static Public Member Function Documentation

static int64_t epochEnd()

Gets the encoded epoch end.

Returns
The encoded epoch end.

static int64_t epochStart()

Gets the encoded epoch start.

Returns
The encoded epoch start.

static bool isValidTimestamp(uint32_t month, uint32_t day, uint32_t year, uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs)

Determines whether a decoded m/d/y, h:m:s:micros Timestamp value is valid and within the Netezza Timestamp range.

Parameters

month

The month, 1 to 12 inclusive.

day

The day, 1 to 31 inclusive.

year

The year, SQL_YEAR_MIN to SQL_YEAR_MAX inclusive.

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

Returns

FALSE if isValidDate(month, day, year) is FALSE or isValidTime(hour, minute, second, mi-cros) is FALSE; TRUE otherwise.

static int64_t max()

Gets the encoded max.

Returns

The encoded max.

static int64_t min()

Gets the encoded min.

Returns

The encoded min.

NzaeTimeTzField Class Reference

This class provides field access for type timetz.

Inherits NzaeField

Public Member Functions

NzaeTimeTzField addInterval(const NzaeIntervalField &interval)

const Constructs a TimeTzField by adding an interval.

void decodeTimeTz(uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrcs, int16_t *sqlOffset, bool *errorFlag=NULL) const

Converts a Netezza-encoded TimeTz value to h:m:s:micros.

void encodeTimeTz(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrcs, int32_t sqlOffset, bool *errorFlag=NULL)

void fromString(std::string str)

Constructs the field from the string.

bool isValidTimeTz() const

Determines whether a Netezza-encoded TimeTZ value is valid and within range.

NzaeTimeTzField(const NzaeTimeTzField &field)

Constructs a timetz field with value field.

NzaeTimeTzField(const NzaeTimeField &field)

Constructs a timetz field with value field.

NzaeTimeTzField(const NzaeTimestampField &field) Constructs a timetz field with value field.

NzaeTimeTzField()

Constructs a NULL timetz field.

NzaeTimeTzField(NzudsTimeTz val)

Constructs a timetz field with value val.

operator const NzudsTimeTz &() const

Returns the encoded field value.

operator NzaeTimeField() const

Returns the time field value.

operator NzudsTimeTz &()

Returns the encoded field value.

bool operator!=(const NzaeTimeTzField &x)

const Not Equal.

bool operator<(const NzaeTimeTzField &x)

const Less than.

bool operator<=(const NzaeTimeTzField &x)

const Less than or equal.

NzaeTimeTzField& operator=(NzudsTimeTz val)

Assigns the value of the argument to a field object.

NzaeTimeTzField& operator=(const NzaeTimestampField
&field) Assigns the value of the argument to a field object.

NzaeTimeTzField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

NzaeTimeTzField& operator=(const NzaeTimeTzField
&field) Assigns the value of the argument to a field object.

NzaeTimeTzField& operator=(const NzaeTimeField &field)

Assigns the value of the argument to a field object.

bool operator==(const NzaeTimeTzField &x)

const Equal to.

bool operator>(const NzaeTimeTzField &x)

const Greater than.

bool operator>=(const NzaeTimeTzField &x)

const Greater than or equal.

NzaeTimeTzField subInterval(const NzaeIntervalField &interval)

const Constructs a TimeTzField by subtracting an interval.

std::string toString() const

Returns the string representation of the field.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Static Public Member Functions

static int32_t max()

Gets the encoded max.

```
static int32_t min()
```

Gets the encoded min.

```
static int16_t offsetMax()
```

Gets the decoded offset max.

```
static int16_t offsetMin()
```

Gets the decoded offset min.

Detailed Description

This class provides field access for type `timetz`.

See Also

▲ [NzaeField](#)

Public Member Function Documentation

NzaeTimeTzField addInterval(const NzaeIntervalField &interval)

const Constructs a `TimeTzField` by adding an interval.

Parameters

NzaeIntervalField interval

The `NzaeIntervalField` value.

Returns

NzaeTimeTzField

The `TimeTzField` consisting of `Interval` plus `TimeTz`.

See Also

[NzaeIntervalField](#)

void decodeTimeTz(uint8_t *hour, uint8_t *minute, uint8_t *second, uint32_t *mcrs, int16_t *sqlOffset, bool *errorFlag=NULL) const

Converts a Netezza-encoded `TimeTz` value to `h:m:s:micros`.

Parameters

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

sqlOffset

The parameter in which to record the offset in minutes, SQL_OFFSET_MIN to SQL_OFFSET_MAX inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTimeTz(encodedTime, encodedZone) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

void encodeTimeTz(uint32_t hour, uint32_t minute, uint32_t second, uint32_t mcrs, int32_t sqlOffset, bool *errorFlag=NULL)

Parameters

hour

The hour, 0 to 23 inclusive.

minute

The minute, 0 to 59 inclusive.

second

The second, 0 to 59 inclusive.

mcrs

The microsecond, 0 to 999,999 inclusive.

sqlOffset

Offset in minutes, SQL_OFFSET_MIN to SQL_OFFSET_MAX inclusive.

errorFlag

If not NULL, *set to TRUE if isValidTimeTz(hour,minute,second,mcrs) is FALSE; *set to FALSE otherwise.

Exceptions

NzaeException

Converts a h:m:s:micros TimeTZ value to a Netezza-encoded TimeTZ.

void fromString(std::string str)

Constructs the field from the string.

Parameters

str

The string to assign from.

bool isValidTimeTz() const

Determines whether a Netezza-encoded TimeTZ value is valid and within range.

Returns

FALSE if isValidTime(encodedTime) is FALSE, or isValidTimeTzOffset(encodedZone) is FALSE; TRUE otherwise.

NzaeTimeTzField(const NzaeTimeTzField &field)

Constructs a timetz field with value field.

Parameters

NzaeTimeTzField field The
NzaeTimeTzField value.

NzaeTimeTzField(const NzaeTimeField &field)

Constructs a timetz field with value field.

Parameters

NzaeTimeField field
The NzaeTimeField value.

See Also

NzaeTimeField

**NzaeTimeTzField(const NzaeTimestampField
&field)** Constructs a timetz field with value field.

Parameters

NzaeTimestampField field The
NzaeTimestampField value.

See Also

NzaeTimestampField

NzaeTimeTzField() Constructs
a NULL timetz field.

NzaeTimeTzField(NzudsTimeTz val)
Constructs a timetz field with value val.

Parameters

val
The encoded timetz value.

operator const NzudsTimeTz &()
const Returns the encoded field value.

Returns

The encoded value.

operator NzaeTimeField() const

Returns the time field value.

Returns

The timestamp value converted from timetz.

See Also

NzaeTimeField

operator NzudsTimeTz &()

Returns the encoded field value.

Returns

The encoded value.

bool operator!=(const NzaeTimeTzField &x)

const Not Equal.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is not equal to x.

Exceptions

NzaeException

bool operator<(const NzaeTimeTzField &x)

const Less than.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is less than x.

Exceptions

NzaeException

bool operator<=(const NzaeTimeTzField &x)

const Less than or equal.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is less than or equal to x.

Exceptions

NzaeException

NzaeTimeTzField& operator=(NzudsTimeTz val)

Assigns the value of the argument to a field object.

Parameters

val

The encoded value to assign.

Returns

NzaeTimeTzField

NzaeTimeTzField& operator=(const NzaeTimestampField &field) Assigns the value of the argument to a field object.

Parameters

NzaeTimestampField

field The field to assign.

Returns

NzaeTimeTzField

See Also

NzaeTimestampField

NzaeTimeTzField& operator=(NzaeField &field)

Assigns the value of the argument to a field object.

Parameters

NzaeField field

The field to assign.

Returns

NzaeTimeTzField

The field argument may be a different type, as long as it is compatible.

NzaeTimeTzField& operator=(const NzaeTimeTzField &field) Assigns the value of the argument to a field object.

Parameters

NzaeTimeTzField field

The field to assign.

Returns

NzaeTimeTzField

NzaeTimeTzField& operator=(const NzaeTimeField &field) Assigns the value of the argument to a field object.

Parameters

NzaeTimeField field

The field to assign.

Returns

NzaeTimeTzField

See Also

NzaeTimeField

bool operator==(const NzaeTimeTzField &x)

const Equal to.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is equal to x.

Exceptions

NzaeException

bool operator>(const NzaeTimeTzField &x)

const Greater than.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is greater than x.

Exceptions

NzaeException

bool operator>=(const NzaeTimeTzField &x)

const Greater than or equal.

Parameters

NzaeTimeTzField x

The field to compare.

Returns

TRUE if the field is greater than or equal to x.

Exceptions

NzaeException

NzaeTimeTzField subInterval(const NzaeIntervalField &interval)

const Constructs a TimeTzField by subtracting an interval.

Parameters

NzaeIntervalField interval

The NzaeIntevalField value.

Returns

NzaeTimeTzField

the TimeTzField consisting of TimeTz minus interval.

See Also

► [NzaeIntervalField](#)

std::string toString() const

Returns the string representation of the field.

Returns

The string representation.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

Static Public Member Function Documentation

static int32_t max()

Gets the encoded max.

Returns

The encoded max.

static int32_t min()

Gets the encoded min.

Returns

The encoded min.

static int16_t offsetMax()

Gets the decoded offset max.

Returns

The decoded offset max.

static int16_t offsetMin()

Gets the decoded offset min.

Returns

The decoded offset min.

Helpers that return information about the possible legal value ranges for decoded information.

NzaeVarbinaryStringField Class Reference

This class provides field access for type varbinary string.

Inherits NzaeStringField

Public Member Functions

`int length() const` Gets the string length.

`virtual NzaeDataTypes::Types type() const` Returns the type of the field.

Detailed Description

This class provides field access for type varbinary string.

See Also

▲ NzaeStringField

Public Member Function Documentation

`int length() const` Gets the string length.

Returns
The string length in characters, not bytes.

`virtual NzaeDataTypes::Types type() const` Returns the type of the field.

Returns
Types
The field type.

NzaeVariableStringField Class Reference

This class provides field access for type variable string.

Inherits NzaeStringField

Public Member Functions

`int length() const` Gets the string length.

`virtual NzaeDataTypes::Types type() const` Returns the type of the field.

Detailed Description

This class provides field access for type variable string.

See Also

▲ [NzaeStringField](#)

Public Member Function Documentation

int length() const Gets

the string length.

Returns

The string length in bytes.

virtual NzaeDataTypes::Types type()

const Returns the type of the field.

Returns

Types

The field type.

Notices and Trademarks

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
26 Forest Street
Marlborough, MA 01752 U.S.A.*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement

or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

Trademarks

IBM, the IBM logo, ibm.com and Netezza are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trade-mark information" at ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

NEC is a registered trademark of NEC Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and/or other countries.

D-CC, D-C++, Diab+, FastJ, pSOS+, SingleStep, Tornado, VxWorks, Wind River, and the Wind River logo are trademarks, registered trademarks, or service marks of Wind River Systems, Inc. Tornado patent pending.

APC and the APC logo are trademarks or registered trademarks of American Power Conversion Corporation.

Other company, product or service names may be trademarks or service marks of others.



Regulatory and Compliance

Regulatory Notices

Install the NPS system in a restricted-access location. Ensure that only those trained to operate or service the equipment have physical access to it. Install each AC power outlet near the NPS rack that plugs into it, and keep it freely accessible. Provide approved 30A circuit breakers on all power sources.

Product may be powered by redundant power sources. Disconnect ALL power sources before servicing. High leakage current. Earth connection essential before connecting supply. Courant de fuite élevé. Raccordement à la terre indispensable avant le raccordement au réseau.

Homologation Statement

This product may not be certified in your country for connection by any means whatsoever to interfaces of public telecommunications networks. Further certification may be required by law prior to making any such connection. Contact an IBM representative or reseller for any questions.

FCC - Industry Canada Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

CE Statement (Europe)

This product complies with the European Low Voltage Directive 73/23/EEC and EMC Directive 89/336/EEC as amended by European Directive 93/68/EEC.

Warning: This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

VCCI Statement

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づくクラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

Index

A

- abs
 - NzaeNumericField,161
- acceptConnection
 - NzaeRemoteProtocol,177
- acceptConnectionFork
 - NzaeRemoteProtocol,177
- acceptConnectionWithTimeout
 - NzaeRemoteProtocol,177
- acceptConnectionWithTimeoutFork
 - NzaeRemoteProtocol,177
- accumulate
 - NzaeAggregateMessageHandler,70
- adapterType
 - NzaeRuntime,184
- AdapterType
 - NzaeRuntime,181
- add
 - NzaeNumericField,161
- AddColumn
 - NzaeRecord,175
- addEntry
 - NzaeEnvironment,98
 - NzaeLibrary,133
 - NzaeParameters,174
- addInterval
 - NzaeTimeField,198
 - NzaeTimestampField,206
 - NzaeTimeTzField,216
- addOutputColumn
 - NzaeShaper,186
- addOutputColumnNumeric
 - NzaeShaper,187
- addOutputColumnString
 - NzaeShaper,187
- addTime
 - NzaeDateField,88
- addTimeTz
 - NzaeDateField,88
- aeAggregate
 - NzaeApi,73

- aeCallId
 - NzaeRuntime,184
- aeFunction
 - NzaeApi,73
- aeQueryId
 - NzaeRuntime,184
- aeShaper
 - NzaeApi,73
- age
 - NzaeDateField,88
 - NzaeTimestampField,206
- Aggregate,17
- apiType
 - NzaeApi,73
- ApiType
 - NzaeApi,72
- assign
 - NzaeField,106
- autoLoad
 - NzaeLibraryInfo,136

B

- bFreeData
 - NzaeCallbackResult,80
- buildFileName
 - NzaeConnectionPoint,82

C

- catalogIsUpper
 - NzaeShaper,187
- ceil
 - NzaeNumericField,162
- close
 - NzaeAggregate,67
 - NzaeConnectionPoint,82
 - NzaeFunction,112
 - NzaeRemoteProtocol,178
 - NzaeShaper,187
- cmp
 - NzaeNumericField,162
- create
 - NzaeEnvironment,99
 - NzaeLibrary,135

Index

- NzaeParameters,175
- createListener
 - NzaeFactory,101
- createOutputRecord
 - NzaeFunction,112

D

- data
 - NzaeCallbackResult,80
- Data Connection APIs,16
- dataLength
 - NzaeCallbackResult,81
- dataSliceld
 - NzaeRuntime,184
- decodeDate
 - NzaeDateField,89
- decodeTime
 - NzaeTimeField,198
- decodeTimestamp
 - NzaeTimestampField,206
- decodeTimeTz
 - NzaeTimeTzField,216
- div
 - NzaeNumericField,162
- done
 - NzaeFunction,112

E

- encodeDate
 - NzaeDateField,90
- encodeTime
 - NzaeTimeField,198
- encodeTimestamp
 - NzaeTimestampField,208
- encodeTimeTz
 - NzaeTimeTzField,217
- epochEnd
 - NzaeDateField,93
 - NzaeTimestampField,213
- epochStart
 - NzaeDateField,93
 - NzaeTimestampField,213
- evaluate

- NzaeFunctionMessageHandler,116
- execute
 - NzaeRemoteProtocolCallback,179
- exp
 - NzaeNumericField,163

F

- finalResult
 - NzaeAggregateMessageHandler,70
- floor
 - NzaeNumericField,163
- format
 - NzaeException,100
- fromString
 - NzaeBoolField,78
 - NzaeDateField,91
 - NzaeDoubleField,96
 - NzaeField,106
 - NzaeFloatField,109
 - NzaeInt16Field,118
 - NzaeInt32Field,121
 - NzaeInt64Field,123
 - NzaeInt8Field,126
 - NzaeIntervalField,129
 - NzaeNumeric128Field,144
 - NzaeNumeric32Field,149
 - NzaeNumeric64Field,154
 - NzaeStringField,194
 - NzaeTimeField,199
 - NzaeTimestampField,209
 - NzaeTimeTzField,217
- fromStringWithInfo
 - NzaeNumeric128Field,144
 - NzaeNumeric32Field,149
 - NzaeNumeric64Field,154
- Function,17

G

- get
 - NzaeRecord,175
- getAdapterType
 - NzaeRuntime,182
- getAeCallId

- NzaeRuntime,182
- getAeQueryId
 - NzaeRuntime,182
- getApi
 - NzaeApiGenerator,74
- getCallbackHandler
 - NzaeApiGenerator,75
 - NzaeRemoteProtocol,178
- getCorrelationType
 - NzaeMetadata,138
- getDataSliceId
 - NzaeConnectionPoint,82
 - NzaeRuntime,182
- getEnvironment
 - NzaeAggregate,67
 - NzaeFunction,113
 - NzaeShaper,187
- getFactory
 - NzaeFactory,104
- getFirstKey
 - NzaeEnvironment,98
- getHandle
 - NzaeConnectionPoint,82
- getHardwareId
 - NzaeRuntime,182
- getInputColumnCount
 - NzaeMetadata,138
- getInputScale
 - NzaeMetadata,138
- getInputSize
 - NzaeMetadata,138
- getInputType
 - NzaeMetadata,139
- getLibrary
 - NzaeAggregate,67
 - NzaeFunction,113
 - NzaeShaper,188
- getLibraryInfo
 - NzaeLibrary,133
- getLocalAggregationApi
 - NzaeFactory,102
- getLocalApi
 - NzaeFactory,102
- getLocalFunctionApi
 - NzaeFactory,103
- getLocalLibraryInfo
 - NzaeLibrary,134
- getLocalShaperApi
 - NzaeFactory,103
- getLocus
 - NzaeRuntime,183
- getMessageHandler
 - NzaeAggregate,67
 - NzaeFunction,113
 - NzaeShaper,188
- getMetadata
 - NzaeFunction,113
 - NzaeShaper,188
- getName
 - NzaeConnectionPoint,82
- getNextKey
 - NzaeEnvironment,98
- getNumberDataSlices
 - NzaeRuntime,183
- getNumberSpus
 - NzaeRuntime,183
- getNumOutputColumns
 - NzaeShaper,188
- getOutputColumnCount
 - NzaeMetadata,139
- getOutputColumnInfo
 - NzaeShaper,188
- getOutputScale
 - NzaeMetadata,139
- getOutputSize
 - NzaeMetadata,139
- getOutputType
 - NzaeMetadata,139
- getParameter
 - NzaeParameters,174
- getParameters
 - NzaeAggregate,67
 - NzaeFunction,113
 - NzaeShaper,189
- getParentLibraryInfo
 - NzaeLibrary,134
- getParentProcessId
 - NzaeFactory,104

Index

- getProcessId
 - NzaeFactory,104
- getRemoteDataSliceId
 - NzaeConnectionPoint,83
- getRemoteName
 - NzaeConnectionPoint,83
- getRemoteSessionId
 - NzaeConnectionPoint,83
- getRemoteTransactionId
 - NzaeConnectionPoint,83
- getRuntime
 - NzaeAggregate,68
 - NzaeFunction,113
 - NzaeShaper,189
- getSessionId
 - NzaeConnectionPoint,83
 - NzaeRuntime,183
- getsign
 - NzaeNumericField,163
- getSuggestedMemoryLimit
 - NzaeRuntime,183
- getTransactionId
 - NzaeConnectionPoint,83
 - NzaeRuntime,183
- getUserName
 - NzaeRuntime,183
- getUserQuery
 - NzaeRuntime,184
- getValue
 - NzaeEnvironment,99
- getYearDay
 - NzaeDateField,93

H

- hardwareId
 - NzaeRuntime,184
- hasFinal
 - NzaeMetadata,140
- hasKey
 - NzaeEnvironment,99
- hasOrder
 - NzaeMetadata,140
- hasOver
 - NzaeMetadata,140

- hasPartition
 - NzaeMetadata,140

I

- Initialization APIs,15
- initializeState
 - NzaeAggregateMessageHandler,71
- inputIsConstant
 - NzaeMetadata,140
- inputRow
 - NzaeShaper,189
- Integer Fields,19
- isLocal
 - NzaeApiGenerator,76
 - NzaeFactory,103
- isNull
 - NzaeField,106
- isOneOutputRowRestriction
 - NzaeMetadata,141
- isRemote
 - NzaeApiGenerator,76
 - NzaeFactory,103
- isValidDate
 - NzaeDateField,91
 - NzaeDateField,94
- isValidEpochDate
 - NzaeDateField,91
- isValidEpochTimestamp
 - NzaeTimestampField,209
- isValidInterval
 - NzaeIntervalField,129
- isValidTime
 - NzaeTimeField,199
 - NzaeTimeField,203
- isValidTimestamp
 - NzaeTimestampField,209
 - NzaeTimestampField,213
- isValidTimeTz
 - NzaeTimeTzField,217
- isValidUTF8
 - NzaeNationalFixedStringField,141
 - NzaeNationalVariableStringField,142

L

- length
 - NzaeFixedStringField,107
 - NzaeGeometryStringField,117
 - NzaeNationalFixedStringField,141
 - NzaeNationalVariableStringField,142
 - NzaeStringField,194
 - NzaeVarbinaryStringField,223
 - NzaeVariableStringField,224
- libraryFullPath
 - NzaeLibraryInfo,136
- libraryName
 - NzaeLibraryInfo,136
- In
 - NzaeNumericField,163
- locus
 - NzaeRuntime,184
- LocusType
 - NzaeRuntime,182
- log
 - NzaeAggregate,68
 - NzaeFunction,114
 - NzaeNumericField,163
 - NzaeShaper,189
- logFileName
 - NzaeAggregate,68
 - NzaeFunction,114
 - NzaeShaper,190
- LogLevel
 - NzaeAggregate,66
 - NzaeFunction,112
 - NzaeShaper,186

M

- m_columnName
 - NzaeShaperOutputColumnInfo,192
- m_precision
 - NzaeShaperOutputColumnInfo,192
- m_scale
 - NzaeShaperOutputColumnInfo,193
- m_size
 - NzaeShaperOutputColumnInfo,193
- m_type

- NzaeShaperOutputColumnInfo,193
- max
 - NzaeDateField,94
 - NzaeTimeField,203
 - NzaeTimestampField,214
 - NzaeTimeTzField,222
- merge
 - NzaeAggregateMessageHandler,71
- min
 - NzaeDateField,94
 - NzaeTimeField,203
 - NzaeTimestampField,214
 - NzaeTimeTzField,222
- mod
 - NzaeNumericField,164
- mul
 - NzaeNumericField,164

N

- newConnectionPoint
 - NzaeFactory,104
- newField
 - NzaeNumericField,172
- newInstance
 - NzaeAggregate,69
 - NzaeConnectionPoint,84
 - NzaeFunction,115
 - NzaeShaper,191
- next
 - NzaeFunction,114
- nextPartition
 - NzaeFunction,114
- numberDataSlices
 - NzaeRuntime,184
- numberSpus
 - NzaeRuntime,184
- numDaysInMonth
 - NzaeDateField,94
- Numeric Fields,20
- numFields
 - NzaeRecord,176
- nz,23
- nz::ae,23
- operator!,29

Index

operator%,29
operator%,29
operator%,30
operator%,30
operator%,30
operator%,31
operator%,31
operator%,31
operator%,32
operator%,32
operator%,32
operator%,33
operator%,33
operator%,33
operator%,34
operator%,34
operator*,34
operator*,35
operator*,35
operator*,35
operator*,36
operator*,36
operator*,36
operator*,37
operator*,37
operator*,37
operator*,38
operator*,38
operator*,38
operator*,39
operator*,39
operator*,39
operator*,39
operator+,40
operator+,40
operator+,40
operator+,41
operator+,41
operator+,41
operator+,42
operator+,42
operator+,42
operator+,42
operator+,43
operator+,43
operator+,43
operator+,44
operator+,44
operator+,44
operator+,44
operator+,45
operator+,45
operator+,45
operator+,46
operator+,46
operator+,46
operator+,47
operator+,47
operator+,47
operator+,48
operator+,48
operator+,48
operator+,48
operator+,49
operator++,49
operator-,49
operator-,50
operator-,50
operator-,50
operator-,51
operator-,51
operator-,51
operator-,52
operator-,52
operator-,52
operator-,53
operator-,53
operator-,53
operator-,54
operator-,54
operator-,54
operator-,55
operator-,55
operator-,55
operator-,56
operator-,56
operator-,56
operator-,57
operator-,57
operator--,57
operator/,58

- operator/,58
- operator/,58
- operator/,59
- operator/,59
- operator/,59
- operator/,60
- operator/,60
- operator/,60
- operator/,61
- operator/,61
- operator/,61
- operator/,62
- operator/,62
- operator/,62
- operator/,63
- NzaeAggregate,65
 - close,67
 - getEnvironment,67
 - getLibrary,67
 - getMessageHandler,67
 - getParameters,67
 - getRuntime,68
 - log,68
 - logFileName,68
 - LogLevel,66
 - newInstance,69
 - NzaeAggType,66
 - NzaeAggType,67
 - ping,68
 - runAggregation,68
 - type,69
 - userError,69
 - ~NzaeAggregate,69
- NzaeAggregateInitialization,69
- NzaeAggregateMessageHandler,69
 - accumulate,70
 - finalResult,70
 - initializeState,71
 - merge,71
 - ~NzaeAggregateMessageHandler,71
- NzaeAggType
 - NzaeAggregate,66
- NzaeApi,73
 - aeAggregate,73
 - aeFunction,73
 - aeShaper,73
 - ApiType,72
 - apiType,73
 - NzaeApi,73
 - NzaeApi,73
 - ~NzaeApi,73
- NzaeApiGenerator,76
 - getApi,74
 - getApi,75
 - getCallbackHandler,75
 - isLocal,76
 - isRemote,76
 - NzaeApiGenerator,76
 - NzaeApiGenerator,76
 - ownsAPI,76
 - setCallbackHandler,76
 - setDataSliceId,76
 - setName,77
 - setOwnsAPI,77
 - setSessionId,77
 - setTransactionId,77
 - ~NzaeApiGenerator,77
- NzaeBoolField,79
 - fromString,78
 - NzaeBoolField,79
 - NzaeBoolField,79
 - NzaeBoolField,79
 - NzaeBoolField,79
 - operator bool,79
 - operator=,79
 - operator=,79
 - operator=,80
 - toString,80
 - type,80
- NzaeCallbackResult,80
 - bFreeData,80
 - data,80
 - dataLength,81
 - returnCode,81
- NzaeCallbackType
 - NzaeRemoteProtocolCallback,179
- NzaeConnectionPoint,81
 - buildFileName,82

Index

- close,82
- getDataSlicId,82
- getHandle,82
- getName,82
- getRemoteDataSlicId,83
- getRemoteName,83
- getRemoteSessionId,83
- getRemoteTransactionId,83
- getSessionId,83
- getTransactionId,83
- newInstance,84
- setDataSlicId,83
- setName,84
- setSessionId,84
- setTransactionId,84
- ~NzeConnectionPoint,84
- NzeCorrelationType
 - NzeMetadata,137
- NzeDataTypes,84
 - Types,19
- NzeDateField,91
 - addTime,88
 - addTimeTz,88
 - age,88
 - decodeDate,89
 - decodeDate,89
 - decodeDate,89
 - encodeDate,90
 - encodeDate,90
 - encodeDate,90
 - epochEnd,93
 - epochStart,93
 - fromString,91
 - getYearDay,93
 - isValidDate,91
 - isValidDate,94
 - isValidEpochDate,91
 - max,94
 - min,94
 - numDaysInMonth,94
 - NzeDateField,91
 - NzeDateField,91
 - NzeDateField,91
 - NzeDateField,91
- NzeDateField,91
 - operator int32_t,92
 - operator NzeTimestampField,92
 - operator=,92
 - operator=,92
 - operator=,92
 - operator=,93
 - toString,93
 - type,93
 - yearMax,95
 - yearMin,95
- NzeDoubleField,96
 - fromString,96
 - NzeDoubleField,96
 - NzeDoubleField,96
 - NzeDoubleField,96
 - NzeDoubleField,96
 - operator double,97
 - operator=,97
 - operator=,97
 - operator=,97
 - toString,97
 - type,97
- NzeEnvironment,98
 - addEntry,98
 - create,99
 - getFirstKey,98
 - getNextKey,98
 - getValue,99
 - hasKey,99
 - setReadOnly,99
 - size,99
 - ~NzeEnvironment,99
- NzeException,100
 - format,100
 - NzeException,100
 - NzeException,100
 - ~NzeException,100
- NzeFactory,100
 - createListener,101
 - getFactory,104
 - getLocalAggregationApi,102
 - getLocalApi,102
 - getLocalFunctionApi,103

- getLocalShaperApi,103
- getParentProcessId,104
- getProcessId,104
- isLocal,103
- isRemote,103
- newConnectionPoint,104
- ~NzaeFactory,104
- NzaeField,106
 - assign,106
 - fromString,106
 - isNull,106
 - NzaeField,106
 - NzaeField,106
 - operator=,106
 - setNull,106
 - toString,107
 - type,107
 - ~NzaeField,107
- NzaeFixedStringField,107
 - length,107
 - type,108
- NzaeFloatField,109
 - fromString,109
 - NzaeFloatField,109
 - NzaeFloatField,109
 - NzaeFloatField,109
 - NzaeFloatField,109
 - operator float,109
 - operator=,109
 - operator=,109
 - operator=,110
 - toString,110
 - type,110
- NzaeFunction,110
 - close,112
 - createOutputRecord,112
 - done,112
 - getEnvironment,112
 - getLibrary,113
 - getMessageHandler,113
 - getMetadata,113
 - getParameters,113
 - getRuntime,113
 - log,114
 - logFileName,114
 - LogLevel,112
 - newInstance,115
 - next,114
 - nextPartition,114
 - outputResult,114
 - ping,115
 - run,115
 - userError,115
 - ~NzaeFunction,115
- NzaeFunctionInitialization,115
- NzaeFunctionMessageHandler,116
 - evaluate,116
 - ~NzaeFunctionMessageHandler,117
- NzaeGeometryStringField,117
 - length,117
 - type,117
- NzaeInt16Field,119
 - fromString,118
 - NzaeInt16Field,118
 - NzaeInt16Field,118
 - NzaeInt16Field,119
 - NzaeInt16Field,119
 - operator int16_t,119
 - operator=,119
 - operator=,119
 - operator=,119
 - toString,120
 - type,120
- NzaeInt32Field,121
 - fromString,121
 - NzaeInt32Field,121
 - NzaeInt32Field,121
 - NzaeInt32Field,121
 - NzaeInt32Field,121
 - operator int32_t,121
 - operator=,121
 - operator=,122
 - operator=,122
 - toString,122
 - type,122
- NzaeInt64Field,124
 - fromString,123
 - NzaeInt64Field,123

Index

NzaeInt64Field,123
NzaeInt64Field,123
NzaeInt64Field,124
operator int64_t,124
operator=,124
operator=,124
operator=,124
toString,125
type,125
NzaeInt8Field,126
fromString,126
NzaeInt8Field,126
NzaeInt8Field,126
NzaeInt8Field,126
NzaeInt8Field,126
operator int8_t,126
operator=,126
operator=,127
operator=,127
toString,127
type,127
NzaeIntervalField,129
fromString,129
isValidInterval,129
NzaeIntervalField,129
NzaeIntervalField,129
NzaeIntervalField,129
NzaeIntervalField,129
operator const NzaeTimeField,129
operator const NzudsInterval &,129
operator NzudsInterval &,130
operator!=,130
operator>,131
operator>=,132
operator<,130
operator<=,130
operator=,130
operator=,131
operator=,131
operator==,131
toString,132
type,132
NzaeLibrary,132
addEntry,133
create,135
getLibraryInfo,133
getLocalLibraryInfo,134
getParentLibraryInfo,134
NzaeLibrarySearchType,133
NzaeLibrarySearchType,133
setReadOnly,135
sizeLocalEntries,135
sizeParentEntries,135
~NzaeLibrary,135
NzaeLibraryInfo,135
autoLoad,136
libraryFullPath,136
libraryName,136
NzaeLibrarySearchType
NzaeLibrary,133
NzaeMetadata,141
getCorrelationType,138
getInputColumnCount,138
getInputScale,138
getInputSize,138
getInputType,139
getOutputColumnCount,139
getOutputScale,139
getOutputSize,139
getOutputType,139
hasFinal,140
hasOrder,140
hasOver,140
hasPartition,140
inputIsConstant,140
isOneOutputRowRestriction,141
NzaeCorrelationType,137
NzaeCorrelationType,138
NzaeMetadata,141
NzaeMetadata,141
~NzaeMetadata,141
NzaeNationalFixedStringField,141
isValidUTF8,141
length,141
type,142
NzaeNationalVariableStringField,142
isValidUTF8,142
length,142

- type,143
- NzaeNumeric128Field,145
 - fromString,144
 - fromStringWithInfo,144
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - NzaeNumeric128Field,145
 - operator const NzudsNumeric128,146
 - operator double,146
 - operator NzudsNumeric128,146
 - operator=,146
 - operator=,146
 - operator=,147
 - operator=,147
 - operator=,147
 - operator=,147
 - operator=,147
 - operator=,147
 - toString,148
 - type,148
- NzaeNumeric32Field,151
 - fromString,149
 - fromStringWithInfo,149
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,150
 - NzaeNumeric32Field,151
 - operator const NzudsNumeric32 &,151
 - operator double,151
 - operator NzudsNumeric32 &,151
 - operator=,151
 - operator=,151
 - operator=,152
 - operator=,152
 - operator=,152
 - operator=,152

- operator=,152
- toString,153
- type,153
- NzaeNumeric64Field,156
 - fromString,154
 - fromStringWithInfo,154
 - NzaeNumeric64Field,155
 - NzaeNumeric64Field,155
 - NzaeNumeric64Field,155
 - NzaeNumeric64Field,155
 - NzaeNumeric64Field,155
 - NzaeNumeric64Field,156
 - NzaeNumeric64Field,156
 - operator const NzudsNumeric64,156
 - operator double,156
 - operator NzudsNumeric64,156
 - operator=,156
 - operator=,157
 - operator=,157
 - operator=,157
 - operator=,157
 - operator=,157
 - operator=,158
 - toString,158
 - type,158
- NzaeNumericField,165
 - abs,161
 - add,161
 - ceil,162
 - cmp,162
 - div,162
 - exp,163
 - floor,163
 - getsign,163
 - ln,163
 - log,163
 - log,164
 - mod,164
 - mul,164
 - newField,172
 - newField,173
 - newField,173
 - newField,173

Index

NzaeNumericField,165
NzaeNumericField,165
operator double,165
operator!=,165
operator%=",165
operator>,168
operator>=,168
operator<,167
operator<=,167
operator*=",165
operator++,166
operator+=,166
operator--,166
operator-=,166
operator/=",166
operator=,167
operator=,167
operator=,167
operator=,168
operator==,168
power,169
precision,169
round,169
scale,169
setPrecision,170
setScale,170
sqrt,170
sub,170
toNumeric128,170
toNumeric32,171
toNumeric64,171
trunc,171
uminus,172
uplus,172
~NzaeNumericField,172
NzaeParameters,173
 addEntry,174
 create,175
 getParameter,174
 setReadOnly,174
 size,174
 ~NzaeParameters,175
NzaeRecord,176
 AddColumn,175
 get,175
 numFields,176
 NzaeRecord,176
 NzaeRecord,176
 setShapeReadOnly,176
 ~NzaeRecord,176
NzaeRemoteProtocol,176
 acceptConnection,177
 acceptConnectionFork,177
 acceptConnectionWithTimeout,177
 acceptConnectionWithTimeoutFork,177
 close,178
 getCallbackHandler,178
 setCallbackHandler,178
 ~NzaeRemoteProtocol,178
NzaeRemoteProtocolCallback,179
 execute,179
 NzaeCallbackType,179
 ~NzaeRemoteProtocolCallback,180
NzaeRuntime,180
 AdapterType,181
 adapterType,184
 aeCallId,184
 aeQueryId,184
 dataSliceId,184
 getAdapterType,182
 getAeCallId,182
 getAeQueryId,182
 getDataSliceId,182
 getHardwareId,182
 getLocus,183
 getNumberDataSlices,183
 getNumberSpus,183
 getSessionId,183
 getSuggestedMemoryLimit,183
 getTransactionId,183
 getUserName,183
 getUserQuery,184
 hardwareId,184
 locus,184
 LocusType,182
 numberDataSlices,184
 numberSpus,184
 sessionId,184

- suggestedMemoryLimit,184
- transactionId,184
- userName,184
- userQuery,184
- NzaeShaper,184
 - addOutputColumn,186
 - addOutputColumnNumeric,187
 - addOutputColumnString,187
 - catalogIsUpper,187
 - close,187
 - getEnvironment,187
 - getLibrary,188
 - getMessageHandler,188
 - getMetadata,188
 - getNumOutputColumns,188
 - getOutputColumnInfo,188
 - getParameters,189
 - getRuntime,189
 - inputRow,189
 - log,189
 - logFileName,190
 - LogLevel,186
 - newInstance,191
 - outputType,190
 - ping,190
 - run,190
 - update,190
 - userError,190
 - ~NzaeShaper,190
- NzaeShaperInitialization,191
- NzaeShaperMessageHandler,191
 - shaper,191
 - ~NzaeShaperMessageHandler,192
- NzaeShaperOutputColumn,192
- NzaeShaperOutputColumnInfo,192
 - m_columnName,192
 - m_precision,192
 - m_scale,193
 - m_size,193
 - m_type,193
- NzaeStringField,195
 - fromString,194
 - length,194
 - NzaeStringField,194
 - NzaeStringField,194
 - NzaeStringField,195
 - operator std::string &,195
 - operator=,195
 - operator=,195
 - operator=,195
 - toString,195
 - type,196
- NzaeTimeField,200
 - addInterval,198
 - decodeTime,198
 - encodeTime,198
 - fromString,199
 - isValidTime,199
 - isValidTime,203
 - max,203
 - min,203
 - NzaeTimeField,199
 - NzaeTimeField,199
 - NzaeTimeField,199
 - NzaeTimeField,199
 - NzaeTimeField,199
 - NzaeTimeField,200
 - offsetTime,200
 - operator int64_t,200
 - operator NzaeIntervalField,200
 - operator NzaeTimeTzField,200
 - operator=,201
 - operator=,201
 - operator=,201
 - operator=,201
 - operator=,201
 - subInterval,202
 - subTime,202
 - toString,202
 - type,202
- NzaeTimestampField,210
 - addInterval,206
 - age,206
 - decodeTimestamp,206
 - decodeTimestamp,207
 - decodeTimestamp,207
 - decodeTimestamp,207

Index

- encodeTimestamp,208
- encodeTimestamp,208
- encodeTimestamp,209
- encodeTimestamp,209
- epochEnd,213
- epochStart,213
- fromString,209
- isValidEpochTimestamp,209
- isValidTimestamp,209
- isValidTimestamp,213
- max,214
- min,214
- NzaeTimestampField,210
- NzaeTimestampField,210
- NzaeTimestampField,210
- NzaeTimestampField,210
- NzaeTimestampField,210
- offsetTimestamp,210
- operator int64_t,210
- operator NzaeDateField,211
- operator NzaeTimeField,211
- operator NzaeTimeTzField,211
- operator=,211
- operator=,211
- operator=,211
- operator=,212
- subInterval,212
- subTimestamp,212
- toString,212
- type,213
- NzaeTimeTzField,218
 - addInterval,216
 - decodeTimeTz,216
 - encodeTimeTz,217
 - fromString,217
 - isValidTimeTz,217
 - max,222
 - min,222
 - NzaeTimeTzField,217
 - NzaeTimeTzField,217
 - NzaeTimeTzField,218
 - NzaeTimeTzField,218
 - NzaeTimeTzField,218
 - NzaeTimeTzField,218

- offsetMax,222
- offsetMin,222
- operator const NzudsTimeTz &,218
- operator NzaeTimeField,218
- operator NzudsTimeTz &,219
- operator!=,219
- operator>,221
- operator>=,221
- operator<,219
- operator<=,219
- operator=,219
- operator=,220
- operator=,220
- operator=,220
- operator=,220
- operator==,221
- subInterval,221
- toString,222
- type,222
- NzaeVarbinaryStringField,223
 - length,223
 - type,223
- NzaeVariableStringField,223
 - length,224
 - type,224

O

- offsetMax
 - NzaeTimeTzField,222
- offsetMin
 - NzaeTimeTzField,222
- offsetTime
 - NzaeTimeField,200
- offsetTimestamp
 - NzaeTimestampField,210
- operator bool
 - NzaeBoolField,79
- operator const NzaeTimeField
 - NzaeIntervalField,129
- operator const NzudsInterval &
 - NzaeIntervalField,129
- operator const NzudsNumeric128
 - NzaeNumeric128Field,146
- operator const NzudsNumeric32 &

NzaeNumeric32Field,151
 operator const NzudsNumeric64
 NzaeNumeric64Field,156
 operator const NzudsTimeTz &
 NzaeTimeTzField,218
 operator double
 NzaeDoubleField,97
 NzaeNumeric128Field,146
 NzaeNumeric32Field,151
 NzaeNumeric64Field,156
 NzaeNumericField,165
 operator float
 NzaeFloatField,109
 operator int16_t
 NzaeInt16Field,119
 operator int32_t
 NzaeDateField,92
 NzaeInt32Field,121
 operator int64_t
 NzaeInt64Field,124
 NzaeTimeField,200
 NzaeTimestampField,210
 operator int8_t
 NzaeInt8Field,126
 operator NzaeDateField
 NzaeTimestampField,211
 operator NzaeIntervalField
 NzaeTimeField,200
 operator NzaeTimeField
 NzaeTimestampField,211
 NzaeTimeTzField,218
 operator NzaeTimestampField
 NzaeDateField,92
 operator NzaeTimeTzField
 NzaeTimeField,200
 NzaeTimestampField,211
 operator NzudsInterval &
 NzaeIntervalField,130
 operator NzudsNumeric128
 NzaeNumeric128Field,146
 operator NzudsNumeric32 &
 NzaeNumeric32Field,151
 operator NzudsNumeric64
 NzaeNumeric64Field,156
 operator NzudsTimeTz &
 NzaeTimeTzField,219
 operator std::string &
 NzaeStringField,195
 operator!
 nz::ae,29
 operator!=
 NzaeIntervalField,130
 NzaeNumericField,165
 NzaeTimeTzField,219
 operator%
 nz::ae,29
 operator%=
 NzaeNumericField,165
 operator>
 NzaeIntervalField,131
 NzaeNumericField,168
 NzaeTimeTzField,221
 operator>=
 NzaeIntervalField,132
 NzaeNumericField,168
 NzaeTimeTzField,221
 operator<
 NzaeIntervalField,130
 NzaeNumericField,167
 NzaeTimeTzField,219
 operator<=
 NzaeIntervalField,130
 NzaeNumericField,167
 NzaeTimeTzField,219
 operator*
 nz::ae,34
 operator*=
 NzaeNumericField,165
 operator+
 nz::ae,40
 operator++
 nz::ae,49
 NzaeNumericField,166
 operator+=
 NzaeNumericField,166
 operator-
 nz::ae,49
 operator--

Index

- nz::ae,57
- NzaeNumericField,166
- operator=
 - NzaeNumericField,166
- operator/
 - nz::ae,58
- operator/=
 - NzaeNumericField,166
- operator=
 - NzaeBoolField,79
 - NzaeDateField,92
 - NzaeDoubleField,97
 - NzaeField,106
 - NzaeFloatField,109
 - NzaeInt16Field,119
 - NzaeInt32Field,121
 - NzaeInt64Field,124
 - NzaeInt8Field,126
 - NzaeIntervalField,130
 - NzaeNumeric128Field,146
 - NzaeNumeric32Field,151
 - NzaeNumeric64Field,156
 - NzaeNumericField,167
 - NzaeStringField,195
 - NzaeTimeField,201
 - NzaeTimestampField,211
 - NzaeTimeTzField,219
- operator==
 - NzaeIntervalField,131
 - NzaeNumericField,168
 - NzaeTimeTzField,221
- outputResult
 - NzaeFunction,114
- outputType
 - NzaeShaper,190
- ownsAPI
 - NzaeApiGenerator,76

P

- ping
 - NzaeAggregate,68
 - NzaeFunction,115
 - NzaeShaper,190
- power

- NzaeNumericField,169
- precision
 - NzaeNumericField,169

R

- Record and Data Type Support,18
 - Types,19
- Remote Initialization,16
- returnCode
 - NzaeCallbackResult,81
- round
 - NzaeNumericField,169
- run
 - NzaeFunction,115
 - NzaeShaper,190
- runAggregation
 - NzaeAggregate,68
- Runtime and Environment Information,22

S

- scale
 - NzaeNumericField,169
- sessionId
 - NzaeRuntime,184
- setCallbackHandler
 - NzaeApiGenerator,76
 - NzaeRemoteProtocol,178
- setDataSliceId
 - NzaeApiGenerator,76
 - NzaeConnectionPoint,83
- setName
 - NzaeApiGenerator,77
 - NzaeConnectionPoint,84
- setNull
 - NzaeField,106
- setOwnsAPI
 - NzaeApiGenerator,77
- setPrecision
 - NzaeNumericField,170
- setReadOnly
 - NzaeEnvironment,99
 - NzaeLibrary,135
 - NzaeParameters,174

- setScale
 - NzaeNumericField,170
- setSessionId
 - NzaeApiGenerator,77
 - NzaeConnectionPoint,84
- setShapeReadOnly
 - NzaeRecord,176
- setTransactionId
 - NzaeApiGenerator,77
 - NzaeConnectionPoint,84
- shaper
 - NzaeShaperMessageHandler,191
- Shaper and Sizer,17
- size
 - NzaeEnvironment,99
 - NzaeParameters,174
- sizeLocalEntries
 - NzaeLibrary,135
- sizeParentEntries
 - NzaeLibrary,135
- sqrt
 - NzaeNumericField,170
- String Fields,20
- sub
 - NzaeNumericField,170
- subInterval
 - NzaeTimeField,202
 - NzaeTimestampField,212
 - NzaeTimeTzField,221
- subTime
 - NzaeTimeField,202
- subTimestamp
 - NzaeTimestampField,212
- suggestedMemoryLimit
 - NzaeRuntime,184
- Support APIs,21

T

- Temporal Fields,21
- toNumeric128
 - NzaeNumericField,170
- toNumeric32
 - NzaeNumericField,171
- toNumeric64

- NzaeNumericField,171
- toString
 - NzaeBoolField,80
 - NzaeDateField,93
 - NzaeDoubleField,97
 - NzaeField,107
 - NzaeFloatField,110
 - NzaeInt16Field,120
 - NzaeInt32Field,122
 - NzaeInt64Field,125
 - NzaeInt8Field,127
 - NzaeIntervalField,132
 - NzaeNumeric128Field,148
 - NzaeNumeric32Field,153
 - NzaeNumeric64Field,158
 - NzaeStringField,195
 - NzaeTimeField,202
 - NzaeTimestampField,212
 - NzaeTimeTzField,222
- transactionId
 - NzaeRuntime,184
- trunc
 - NzaeNumericField,171
- type
 - NzaeAggregate,69
 - NzaeBoolField,80
 - NzaeDateField,93
 - NzaeDoubleField,97
 - NzaeField,107
 - NzaeFixedStringField,108
 - NzaeFloatField,110
 - NzaeGeometryStringField,117
 - NzaeInt16Field,120
 - NzaeInt32Field,122
 - NzaeInt64Field,125
 - NzaeInt8Field,127
 - NzaeIntervalField,132
 - NzaeNationalFixedStringField,142
 - NzaeNationalVariableStringField,143
 - NzaeNumeric128Field,148
 - NzaeNumeric32Field,153
 - NzaeNumeric64Field,158
 - NzaeStringField,196
 - NzaeTimeField,202

Index

- NzaeTimestampField,213
- NzaeTimeTzField,222
- NzaeVarbinaryStringField,223
- NzaeVariableStringField,224
- Types
 - NzaeDataTypes,19
 - Record and Data Type Support,19

U

- uminus
 - NzaeNumericField,172
- update
 - NzaeShaper,190
- uplus
 - NzaeNumericField,172
- userError
 - NzaeAggregate,69
 - NzaeFunction,115
 - NzaeShaper,190
- userName
 - NzaeRuntime,184
- userQuery
 - NzaeRuntime,184

Y

- yearMax
 - NzaeDateField,95
- yearMin
 - NzaeDateField,95

Symbols

- ~NzaeAggregate
 - NzaeAggregate,69
- ~NzaeAggregateMessageHandler
 - NzaeAggregateMessageHandler,71
- ~NzaeApi
 - NzaeApi,73
- ~NzaeApiGenerator
 - NzaeApiGenerator,77
- ~NzaeConnectionPoint
 - NzaeConnectionPoint,84
- ~NzaeEnvironment

- NzaeEnvironment,99
- ~NzaeException
 - NzaeException,100
- ~NzaeFactory
 - NzaeFactory,104
- ~NzaeField
 - NzaeField,107
- ~NzaeFunction
 - NzaeFunction,115
- ~NzaeFunctionMessageHandler
 - NzaeFunctionMessageHandler,117
- ~NzaeLibrary
 - NzaeLibrary,135
- ~NzaeMetadata
 - NzaeMetadata,141
- ~NzaeNumericField
 - NzaeNumericField,172
- ~NzaeParameters
 - NzaeParameters,175
- ~NzaeRecord
 - NzaeRecord,176
- ~NzaeRemoteProtocol
 - NzaeRemoteProtocol,178
- ~NzaeRemoteProtocolCallback
 - NzaeRemoteProtocolCallback,180
- ~NzaeShaper
 - NzaeShaper,190
- ~NzaeShaperMessageHandler
 - NzaeShaperMessageHandler,192