IBM® Netezza® Analytics
Release 11.x

Java Analytic Executables API Reference

Part Number 00X6336-00 Rev. 1

Note: Before using this information and the product that it supports, read the information in "Notices and Trademarks" on page 83.

Contents

Notices and Trademarks

Index

Preface

This guide provides an API reference for Java AE programmers.

Audience for This Guide

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

Purpose of This Guide

This guide describes the Java AE API, which is a language adapter provided as part of IBM Netezza Analytics. The Java AE API provides programmatic access to the AE interface for Java 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 com-ponents:

1. Retry the action, carefully following the instructions in the documentation.
2. 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.
3. 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

http://www14.software.ibm.com/webapp/set2/sas/f/handbook/contacts.html#phone.

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

Record and Data Type Support

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

Classes

- class NzaeDataTypes
  This class contains constant int values corresponding to the Netezza database column data types.
- interface NzaeFieldInfo
  This class contains information for a specific field (column) including NZ data type, Java SQL data type, and default Java lanaguage data types.
- class NzaeInterval
  This class is for the Interval data type.
- interface NzaeRecord
  A record corresponds to an input or output database row.
- class NzaeTimeTz
  This class is for the TimeTz data type.

Detailed Description

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

Support APIs

This API family provides support functions for date and time conversions, numeric conversions, and getting runtime environment information.

Classes

- class NzaeException
  Unchecked Exception used thoughout the Java AE API.
- class NzaeUtil
  This class contains static utility methods that are not dependent on the main NzaeFactory in-terface.

Modules

- Runtime and Environment Information.
  Runtime, Environment, and Shared Library Information.

Detailed Description

This API family provides support functions for date and time conversions, numeric conversions, and getting runtime environment information.

Runtime and Environment Information.

Runtime, Environment, and Shared Library Information.

Classes

- interface NzaeEnvironment
  This is the Java AE Environment API interface.
- interface NzaeLibrary
  This is the Java AE Library API interface.
- class NzaeRuntime
  Constant runtime information.

Detailed Description

Runtime, Environment, and Shared Library Information.

Data Connection APIs.

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

Classes

- interface NzaeBase
  The base interface for data connection APIs.
- interface NzaeMessageHandler
  This class handles Nzae messages.
- class NzaeMetadata
  This class contains metadata about the AE, including input and output column attributes.

Modules

- Function
  Function AEs are called from Scalar or Table SQL Functions.
- Aggregate
  Aggregate AEs are called from Aggregate SQL 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 Scalar or Table SQL Functions.

Classes

- interface Nzae
  This is the main Java AE Function API interface.

Detailed Description

Function AEs are called from Scalar or Table SQL Functions.

Aggregate

Aggregate AEs are called from Aggregate SQL Functions.

Classes

- interface NzaeAgg
  This is the main Java AE Aggregation API interface.
- interface NzaeAggMessageHandler
  This class handles NzaeAggregation messages.

Detailed Description

Aggregate AEs are called from Aggregate SQL Functions.

Shaper and Sizer

Shapers are optionally called for Table Function AEs.

Classes

- interface NzaeShaper
  This is the main Java AE Shaper API interface.
- interface NzaeShaperMessageHandler

This class handles NzaeShaper messages.

Detailed Description

Shapers are optionally called for Table Function AEs.

Sizers are optionally called for Scalar Function AEs.

Initialization APIs.

This API family is used to get an open data connection.

Classes

- class NzaeAggInitialization
  Not implemented.
- class NzaeApi
  This class is the result of an AE connection request.
- class NzaeApiGenerator
  This class helps manage local and remote connections.
- interface NzaeFactory
  Used to create all objects and provide all information that is dependent on the underlying C implementation of AE.
- class NzaeInitialization
  Not implemented.
- class NzaeShaperInitialization
  Not implemented.

Modules

- Remote Initialization.
  Initialization functions related to Remote AEs.

Detailed Description

This API family is used to get an open data connection.

To get a data connection using system defaults use class NzaeApiGenerator .

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

For a greater range of options use class NzaeFactory and the Remote Initialization classes de-scribed below.

Remote Initialization.

Initialization functions related to Remote AEs.

Classes

- interface NzaeConnectionPoint
  Netezza Analytic Executables.
- interface NzaeRemoteProtocol
  Netezza Analytic Executables.
- interface NzaeRemoteProtocolCallback
  Netezza Analytic Executables.

Detailed Description

Initialization functions related to Remote AEs.

1) Create a connection point. 2) Listen using that connection point. 3) Accept a Data Connection Instance.

CHAPTER 2
Class Documentation

Nzae Interface Reference

This is the main Java AE Function API interface.

Inherits NzaeBase

Public Member Functions

- void close()
  Closes the AE and releases its resources.
- NzaeRecord createOutputRecord()
  Creates a new NzaeRecord object compatible for output.
- void done()
  Indicates the AE is finishing successfully and does not get any more rows or output any more res-ults.
- int getApiType()
  Returns the API Constant for this interface.
- NzaeDataTypes getDataTypes()
  Gets the NZ Data Type constants.
- NzaeEnvironment getEnvironment() Gets
  environment information for the AE.
- boolean getInterpretCharAsBinary(int index)
  Gets the TRUE / FALSE state of the interpretation mode of a char or varchar column as a byte ar-ray.
- NzaeLibrary getLibrary()
  Gets the library information for the AE.
- NzaeMessageHandler getMessageHandler()

Returns the message handler class object.

- NzaeMetadata getMetadata()
  Gets metadata about the AE, including the input and output columns.
- NzaeRuntime getRuntime()
  Gets runtime information the AE, including information about the Netezza system.
- void log(int logLevel, String message)
  Logs the specified message at the specified log level.
- NzaeRecord next() Gets
  the next input row.
- boolean nextPartition()
  Gets the next partition.
- void outputResult(NzaeRecord
  rec) Outputs the record.
- void ping()
  Indicates that the AE is still active and not hanging.
- void run(NzaeMessageHandler messageHandler)
  Begins higher level function message processing.
- void setInterpretCharAsBinary(int index, boolean arg)
  Interprets a char or varchar column as a byte array.
- int size()
  Gets the field size.
- void userError(String message)
  Indicates the AE has encountered an error condition.

Static Public Attributes

- LOG_DEBUG
  Log level debug.
- LOG_TRACE
  Log level trace.

Detailed Description

This is the main Java AE Function API interface.

A reference to this interface can be obtained from static methods in the NzaeFactory class.

- See Also
  - NzaeFactory

Public Member Function Documentation

- **void close()**

Closes the AE and releases its resources.

- **NzaeRecord createOutputRecord()**
  Creates a new NzaeRecord object compatible for output.
  - Returns
    NzaeRecord

    An instance of NzaeRecord with NULL fields.

  An object is compatible if it has the correct number of fields of the correct database type in the correct order.

- **void done()**
  Indicates the AE is finishing successfully and does not get any more rows or output any more results.

- **int getApiType()**
  Returns the API Constant for this interface.
  - Returns
    The values defined in NzaeApi .

- **NzaeDataTypes getDataTypes()**
  Gets the NZ Data Type constants.
  - Returns
    NzaeDataTypes

    The instance of NzaeDataTypes .

- **NzaeEnvironment getEnvironment()** Gets environment information for the AE.
  - Returns
    NzaeEnvironment

    The instance of NzaeEnvironment .

- **boolean getInterpretCharAsBinary(int index)**
  Gets the TRUE / FALSE state of the interpretation mode of a char or varchar column as a byte array.
  - Parameters
    - **index**
      The column index, which is 0 based.
  - Returns
    TRUE = byte array; FALSE = string.

  The default value is FALSE.

- **NzaeLibrary getLibrary()**
  Gets the library information for the AE.
  - Returns
    NzaeLibrary

    The instance of NzaeLibrary .

- **NzaeMessageHandler getMessageHandler()**
  Returns the message handler class object.
  - Returns
    NzaeMessageHandler

    The instance of NzaeFunctionMessageHandler.

- **NzaeMetadata getMetadata()**
  Gets metadata about the AE, including the input and output columns.
  - Returns
    NzaeMetadata

    The instance of NzaeMetadata .

- **NzaeRuntime getRuntime()**
  Gets runtime information the AE, including information about the Netezza system.
  - Returns
    NzaeRuntime

    The instance of NzaeRuntime .

- **void log(int logLevel, String message)**
  Logs the specified message at the specified log level.
  - Parameters
    - **logLevel**
      The log level constant.
    - **message**
      The message to log.

- **NzaeRecord next()** Gets
  the next input row.
  - Returns
    NzaeRecord

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

- **boolean nextPartition()**
  Gets the next partition.
  - Returns
    TRUE if there is another partition; otherwise FALSE.

- **void outputResult(NzaeRecord rec)** Outputs the record.
  - Parameters
    - **NzaeRecord rec**
      An output compatible instance of NzaeRecord .

- **void ping()**
  Indicates that the AE is still active and not hanging.

- **void run(NzaeMessageHandler messageHandler)**
  Begins higher level function message processing.
  - Parameters
    - **NzaeMessageHandler messageHandler** The message handler.

  Processes one row of input and produces one row of output. Used for all scalar functions and some table functions. Scalar functions use a single field in the result.

  This is an alternative to writing a custom for loop using next and outputResult.

- **void setInterpretCharAsBinary(int index, boolean arg)**
  Interprets a char or varchar column as a byte array.
  - Parameters
    - **index**
      The column index, which is 0 based.
    - **arg**
      TRUE = byte array; FALSE = string.

  The default value is FALSE.

- **int size()**
  Gets the field size.
  - Returns
    The field size.

- **void userError(String message)**

Indicates the AE has encountered an error condition.

- Parameters
  - **message**
    The message to send back to the Netezza system.

Implies NzaeDone.

Static Member Data Documentation

- final int LOG_DEBUG=
  2 Log level debug.

- final int LOG_TRACE=
  1 Log level trace.

NzaeAgg Interface Reference

This is the main Java AE Aggregation API interface.

Inherits NzaeBase

Public Member Functions

- void close()
  Close the AE and release resources.
- int getApiType()
  Returns the API Constant for this interface.
- NzaeDataTypes getDataTypes()
  Gets the NZ Data Type constants.
- NzaeEnvironment getEnvironment()
  Gets environment information the AE.
- boolean getInputInterpretCharAsBinary(int index)
  Get the TRUE / FALSE state of the interpretation mode of a char or varchar column as a byte array.
- NzaeLibrary getLibrary()
  Gets library information for the AE.
- NzaeAggMessageHandler getMessageHandler()
  Returns the message handler class object.
- NzaeRuntime getRuntime()
  Gets runtime information for the AE, including information about the Netezza system.

- boolean getStateInterpretCharAsBinary(int index)
- void log(int logLevel, String message)
  Logs the specified message at the specified log level.
- void ping()
  Indicates that the AE is still active and not hanging.
- void runAggregation(NzaeAggMessageHandler messageHandler) Begin the Aggregation Message Processing.
- void setInputInterpretCharAsBinary(int index, boolean arg) Interprets a char or varchar column as a byte array.
- void setStateInterpretCharAsBinary(int index, boolean arg)
- int type()
  Returns the Aggregate type.
- void userError(String message)
  Indicates this AE has encountered an error condition.

Static Public Attributes

- ANALYTIC
  Aggregate type analytic.
- GROUPED
  Aggregate type grouped.
- LOG_DEBUG
  Log level debug.
- LOG_TRACE
  Log level trace.
- UNKNOWN
  Aggregate type unknown.

Detailed Description

This is the main Java AE Aggregation API interface.

A reference to this interface can be obtained from static methods in the NzaeFactory class.

- See Also
  - NzaeFactory

Public Member Function Documentation

- **void close()**
  Close the AE and release resources.

- **int getApiType()**
  Returns the API Constant for this interface. ▲ Returns

The values defined in NzaeApi .

- **NzaeDataTypes getDataTypes()**
  Gets the NZ Data Type constants.
  - Returns
    NzaeDataTypes

    The instance of NzaeDataTypes .

- **NzaeEnvironment getEnvironment()**
  Gets environment information the AE.
  - Returns
    NzaeEnvironment

    The instance of NzaeEnvironment .

- **boolean getInputInterpretCharAsBinary(int index)**
  Get the TRUE / FALSE state of the interpretation mode of a char or varchar column as a byte array.
  - Parameters
    - **index**
      The column index, which is 0 based.
    - **TRUE**
      = byte array; FALSE = string.

  The default value is FALSE.

- **NzaeLibrary getLibrary()**
  Gets library information for the AE.
  - Returns
    NzaeLibrary

    The instance of NzaeLibrary .

- **NzaeAggMessageHandler getMessageHandler()**
  Returns the message handler class object.
  - Returns
    NzaeAggMessageHandler

    The Message Handler.

- **NzaeRuntime getRuntime()**

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

- Returns
  NzaeRuntime

  The instance of NzaeRuntime .

- **boolean getStateInterpretCharAsBinary(int index)**

- **void log(int logLevel, String message)**
  Logs the specified message at the specified log level.
  - Parameters
    - **logLevel**
      The log level constant.
    - **message**
      The message to log.

- **void ping()**
  Indicates that the AE is still active and not hanging.

- **void runAggregation(NzaeAggMessageHandler messageHandler)** Begin the Aggregation Message Processing.
  - Parameters
    - **NzaeAggMessageHandler**
      messageHandler Message Handler.

- **void setInputInterpretCharAsBinary(int index, boolean arg)** Interprets a char or varchar column as a byte array.
  - Parameters
    - **index**
      The column index, which is 0 based.
    - **arg**
      TRUE = byte array; FALSE = string.

  The default value is FALSE.

- **void setStateInterpretCharAsBinary(int index, boolean arg)**

- **int type()**
  Returns the Aggregate type.
  - Returns
    The Aggregate type.

- **void userError(String message)**
  Indicates this AE has encountered an error condition.
  - Parameters
    - **message**
      The message to send back to the Netezza system.

  Implies NzaeDone.

Static Member Data Documentation

- final int ANALYTIC= 2
  Aggregate type analytic.

- final int GROUPED= 1
  Aggregate type grouped.

- final int LOG_DEBUG=
  2 Log level debug.

- final int LOG_TRACE=
  1 Log level trace.

- final int UNKNOWN= 0
  Aggregate type unknown.

NzaeAggInitialization Class Reference

Not implemented.

Detailed Description

Not implemented.

This class is a placeholder for future parameters for AE Aggregation initialization.

- See Also
  - NzaeFactory

NzaeAggMessageHandler Interface Reference

This class handles NzaeAggregation messages.

Public Member Functions

- void accumulate(NzaeAgg api, NzaeRecord input, NzaeRecord state) Modifies the state depending on input.
- void finalResult(NzaeAgg api, NzaeRecord inputState, NzaeRecord result) Sets the final result, depending on the input state.
- void initializeState(NzaeAgg api, NzaeRecord state) The initialize state.
- void merge(NzaeAgg api, NzaeRecord inputState, NzaeRecord state) Merges the input state into state.

Detailed Description

This class handles NzaeAggregation messages.

- See Also
  - runAggregation

Public Member Function Documentation

- **void accumulate(NzaeAgg api, NzaeRecord input, NzaeRecord state)** Modifies the state depending on input.
  - Parameters
    - **NzaeAgg api**
      The Aggregate instance.
    - **NzaeRecord input**
      The input record.
    - **NzaeRecord state**
      The state record.

- **void finalResult(NzaeAgg api, NzaeRecord inputState, NzaeRecord result)** Sets the final result, depending on the input state.
  - Parameters
    - **NzaeAgg api**
      The Aggregate instance.
    - **NzaeRecord inputState**
      The input state record.
    - **NzaeRecord result**
      The output result record.

  The final result record may contain only a single field.

- **void initializeState(NzaeAgg api, NzaeRecord state)** The initialize state.
  - Parameters
    - **NzaeAgg api**
      The Aggregate instance.
    - **NzaeRecord state**
      The state record.

- **void merge(NzaeAgg api, NzaeRecord inputState, NzaeRecord state)** Merges the input state into state.
  - Parameters
    - **NzaeAgg api**
      The Aggregate instance.
    - **NzaeRecord inputState**
      The input state record.
    - **NzaeRecord state**
      Output state record.

NzaeApi Class Reference

This class is the result of an AE connection request.

Public Member Functions

- void close()
  Closes api API objects.

Public Attributes

- aeAggregate
  Aggregate.
- aeFunction
  Function.
- aeShaper
  Shaper.
- apiType
  Type of API in object.

Static Public Attributes

- AGGREGATION
  Aggregate API type.
- ANY
  Any API type.
- FUNCTION
  Function API type.
- SHAPER
  Shaper API type.
- UNKNOWN
  Unknown API type.

Detailed Description

This class is the result of an AE connection request.

Public Member Function Documentation

- **void close()**
  Closes api API objects.

Member Data Documentation

- NzaeAgg aeAggregate
  Aggregate.

- Nzae aeFunction
  Function.

- NzaeShaper
  aeShaper Shaper.

- int apiType
  Type of API in object.

Static Member Data Documentation

- final int AGGREGATION=
  2 Aggregate API type.

- final int ANY=
  4 Any API type.

- final int FUNCTION=
  1 Function API type.

- final int SHAPER=
  3 Shaper API type.

- final int UNKNOWN=
  0 Unknown API type.

NzaeApiGenerator Class Reference

This class helps manage local and remote connections.

Public Member Functions

- void close()
  Closes the API object if it is owned, has a connection point, and has a remote protocol.
- NzaeApi getApi(int
  type) Gets API.
- NzaeRemoteProtocolCallback getCallbackHandler()
  Gets the Remote Protocol Callback.
- boolean isLocal()
  Return true if this is a local AE process.
- boolean isRemote()
  Determines if the process is a remote AE.
- boolean ownsAPI()
  Determines if the helper owns the API.
- void setCallbackHandler(NzaeRemoteProtocolCallback
  handler) Sets the Remote Protocol Callback.
- void setDataSliceId(int dataSliceId) Sets
  the Connection Point dataslice ID.
- void setName(String name)
  Sets the Connection Point Name.
- void setOwnsAPI(boolean owns)
  Specifies if the object should manage the API.

- void setSessionId(int sessionId)
  Sets the Connection Point session ID.
- void setTransactionId(long transactionId)
  Set the Connection transaction ID.

Detailed Description

This class helps manage local and remote connections.

Public Member Function Documentation

- **void close()**
  Closes the API object if it is owned, has a connection point, and has a remote protocol.

- **NzaeApi getApi(int type)** Gets API.
  - Parameters
    - **type**
      The specified API type or ANY.
  - Returns
    NzaeApi

    API is NULL in parent; in fork mode if remote.
  - Exceptions
    - NzaeException

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

- **NzaeRemoteProtocolCallback getCallbackHandler()**
  Gets the Remote Protocol Callback.
  - Returns
    NzaeRemoteProtocolCallback

    The callback.

- **boolean isLocal()**
  Return true if this is a local AE process.
  - Returns
    true if local AE

- **boolean isRemote()**

Determines if the process is a remote AE.

- Returns
  TRUE if the process is a remote AE.

- **boolean ownsAPI()**
  Determines if the helper owns the API.
  - Returns
    TRUE if the helper owns the API

- **void setCallbackHandler(NzaeRemoteProtocolCallback handler)** Sets the Remote Protocol Callback.
  - Parameters
    - **NzaeRemoteProtocolCallback handler** The remote protocol handler.

- **void setDataSliceId(int dataSliceId)**
  Sets the Connection Point dataslice ID.
  - Parameters
    - **dataSliceId**
      The Connection Point dataslice ID.

  Does not override the remote values from the launcher available in NzaeConnectionPoint .

- **void setName(String name)** Sets the Connection Point Name.
  - Parameters
    - **name**
      The Connection Point name.

  Does not override the remote values from the launcher available in NzaeConnectionPoint .

- **void setOwnsAPI(boolean owns)**
  Specifies if the object should manage the API.
  - Parameters
    - **owns**
      TRUE if the helper owns the API.

  If TRUE, closes the API when a new API is accepted or the helper is closed.

- **void setSessionId(int sessionId)** Sets the Connection Point session ID.

- Parameters
  - **sessionId**
    The Connection Point session ID.

Does not override the remote values from the launcher available in NzaeConnectionPoint .

- **void setTransactionId(long transactionId)**
  Set the Connection transaction ID.
  - Parameters
    - **transactionId**
      The Connection Point transaction ID.

  Does not override the remote values from the launcher available in NzaeConnectionPoint .

NzaeBase Interface Reference

The base interface for data connection APIs.

Public Member Functions

- void close()
  Closes the AE and releases its resources.
- int getApiType()
  Returns the API Constant for this interface.
- NzaeEnvironment getEnvironment()
  Gets the environment information for the AE.
- NzaeLibrary getLibrary()
  Gets the library information for the AE.
- ArrayList<String> getParameters()
  Gets the SQL parameters used to invoke this AE.
- NzaeRuntime getRuntime()
  Gets runtime information for the AE, including information about the Netezza system.
- void log(int logLevel, String message)
  Logs the specified message at the specified log level.
- void userError(String message)
  Indicates this AE has encountered an error condition.

Detailed Description

The base interface for data connection APIs.

Public Member Function Documentation

- **void close()**
  Closes the AE and releases its resources.

- **int getApiType()**
  Returns the API Constant for this interface.
  - Returns
    The values defined in NzaeApi .

- **NzaeEnvironment getEnvironment()**
  Gets the environment information for the AE.
  - Returns
    NzaeEnvironment

    The instance of NzaeEnvironment .

- **NzaeLibrary getLibrary()**
  Gets the library information for the AE.
  - Returns
    NzaeLibrary

    The instance of NzaeLibrary .

- **ArrayList<String> getParameters()**
  Gets the SQL parameters used to invoke this

  AE. @ list of parameter strings.

- **NzaeRuntime getRuntime()**
  Gets runtime information for the AE, including information about the Netezza system.
  - Returns
    NzaeRuntime

    The instance of NzaeRuntime .

- **void log(int logLevel, String message)**
  Logs the specified message at the specified log level.
  - Parameters
    - **logLevel**
      The log level constant.
    - **message**
      The message to log.

- **void userError(String message)**
  Indicates this AE has encountered an error condition.
  - Parameters
    - **message**
      The message to send back to the Netezza system.

  Implies NzaeDone.

NzaeCallbackResult Class Reference

Public Attributes

- data
  The returned data.
- returnCode
  The return code.

Member Data Documentation

- byte [] data
  The returned data.

- int returnCode
  A value of 0 is normal

NzaeConnectionPoint Interface Reference

Netezza Analytic Executables.

Public Member Functions

- String buildFileTypeName()
  Gets the Connection Point file name.
- void close()
  Frees up the Connection Point.
- int getDataSliceId()
  Gets the Connection Point dataslice ID.
- String getName()
  Gets the Connection Point name.
- int getRemoteDataSliceId()

Gets the Remote Dataslice ID used in the launcher.

- String getRemoteName()
  Gets the Remote Name used in the launcher.
- int getRemoteSessionId()
  Gets the Remote Session ID used in the launcher.
- long getRemoteTransactionId()
  Gets the Remote Transaction ID used in the launcher.
- int getSessionId()
  Gets the Connection Point session ID.
- long getTransactionId()
  Gets the Connection Point Transaction ID.
- void setDataSliceId(int dataSliceId) Sets
  the Connection Point dataslice ID.
- void setName(String name) Sets
  the Connection Point name.
- void setSessionId(int sessionId)
  Sets the Connection Point session ID.
- void setTransactionId(long transactionId)
  Sets the Connection Point Transaction ID.

Detailed Description

Netezza Analytic Executables.

Connection Point This interface is used to build a fully qualified connection name based on the supplied properties.

Public Member Function Documentation

- **String buildFileTypeName()**
  Gets the Connection Point file name.
  - Returns
    The file name.

- **void close()**
  Frees up the Connection Point.

- **int getDataSliceId()**
  Gets the Connection Point dataslice ID.
  - Returns

The dataslice ID.

- **String getName()**
  Gets the Connection Point name.
  - Returns
    The name.

- **int getRemoteDataSliceId()**
  Gets the Remote Dataslice ID used in the launcher.
  - Returns
    The remote dataslice ID or -1 if not set.

- **String getRemoteName()**
  Gets the Remote Name used in the launcher.
  - Returns
    The remote name or an empty string if not set.

- **int getRemoteSessionId()**
  Gets the Remote Session ID used in the launcher.
  - Returns
    The remote session ID or -1 if not set.

- **long getRemoteTransactionId()**
  Gets the Remote Transaction ID used in the launcher.
  - Returns
    The remote transaction ID or -1 if not set.

- **int getSessionId()**
  Gets the Connection Point session ID.
  - Returns
    The session ID.

- **long getTransactionId()**
  Gets the Connection Point Transaction ID.
  - Returns
    The transaction ID

- **void setDataSliceId(int dataSliceId)**
  Sets the Connection Point dataslice ID.

- Parameters
  - **dataSliceId** The dataslice ID.

Controls whether the connection point uses the dataslice ID.

- **void setName(String name)** Sets the Connection Point name.
  - Parameters
    - **name**
      The connection point name.

A name must always be set.

- **void setSessionId(int sessionId)** Sets the Connection Point session ID.
  - Parameters
    - **sessionId**
      The session ID.

Controls whether the connection point uses the session ID.

- **void setTransactionId(long transactionId)**
  Sets the Connection Point Transaction ID.
  - Parameters
    - **transactionId**
      The transaction ID.

Controls whether the connection point uses the transaction ID.

NzaeDataTypes Class Reference

This class contains constant int values corresponding to the Netezza database column data types.

Static Public Attributes

- NZUDSUDX_BOOL
  Boolean data type.
- NZUDSUDX_DATE
  Date data type.
- NZUDSUDX_DOUBLE
  Double data type.

- NZUDSUDX_FIXED
  Fixed string data type.
- NZUDSUDX_FLOAT
  Float data type.
- NZUDSUDX_GEOMETRY
  Geometry data type.
- NZUDSUDX_INT16
  Int16 data type.
- NZUDSUDX_INT32
  Int32 data type.
- NZUDSUDX_INT64
  Int64 data type.
- NZUDSUDX_INT8
  Int8 data type.
- NZUDSUDX_INTERVAL
  Interval data type.
- NZUDSUDX_MAX_TYPE
  Max data type.
- NZUDSUDX_NATIONAL_FIXED
  National fixed string data type.
- NZUDSUDX_NATIONAL_VARIABLE
  National variable data type.
- NZUDSUDX_NUMERIC128
  Numeric128 data type.
- NZUDSUDX_NUMERIC32
  Numeric32 data type.
- NZUDSUDX_NUMERIC64
  Numeric64 data type.
- NZUDSUDX_TIME
  Time data type.
- NZUDSUDX_TIMESTAMP
  Timestamp data type.
- NZUDSUDX_TIMETZ
  Timetz data type.
- NZUDSUDX_UNKNOWN
  Unknown data type.
- NZUDSUDX_VARBINARY
  VarBinary data type.
- NZUDSUDX_VARIABLE
  Variable string data type.

Static Public Member Functions

- static String javaTypeToString(int t)
  Returns a String representation of a Java data type from java.sql.Types.
- static String nzTypeToString(int t)
  Returns a String resprsentation of an NZ data type.

Detailed Description

This class contains constant int values corresponding to the Netezza database column data types.

These values are identical to data type values used in the Netezza UDX API.

- See Also
  - getDataTypes
  - getDataTypes

Static Member Data Documentation

- final int NZUDSUDX_BOOL=
  4 Boolean data type.
- final int NZUDSUDX_DATE=
  5 Date data type.
- final int NZUDSUDX_DOUBLE=
  12 Double data type.
- final int NZUDSUDX_FIXED=
  0 Fixed string data type.
- final int NZUDSUDX_FLOAT=
  11 Float data type.
- final int NZUDSUDX_GEOMETRY=
  19 Geometry data type.
- final int NZUDSUDX_INT16=
  15 Int16 data type.
- final int NZUDSUDX_INT32= 16

Int32 data type.

- final int NZUDSUDX_INT64=
  17 Int64 data type.

- final int NZUDSUDX_INT8=
  14 Int8 data type.

- final int NZUDSUDX_INTERVAL=
  13 Interval data type.

- final int NZUDSUDX_MAX_TYPE=
  21 Max data type.

- final int NZUDSUDX_NATIONAL_FIXED=
  2 National fixed string data type.

- final int NZUDSUDX_NATIONAL_VARIABLE=
  3 National variable data type.

- final int NZUDSUDX_NUMERIC128=
  10 Numeric128 data type.

- final int NZUDSUDX_NUMERIC32=
  8 Numeric32 data type.

- final int NZUDSUDX_NUMERIC64=
  9 Numeric64 data type.

- final int NZUDSUDX_TIME=
  6 Time data type.

- final int NZUDSUDX_TIMESTAMP=
  18 Timestamp data type.

- final int NZUDSUDX_TIMETZ=
  7 Timetz data type.

- final int NZUDSUDX_UNKNOWN= -
  1 Unknown data type.

- final int NZUDSUDX_VARBINARY=
  20 VarBinary data type.

- final int NZUDSUDX_VARIABLE=
  1 Variable string data type.

Static Public Member Function Documentation

- **static String javaTypeToString(int t)**
  Returns a String representation of a Java data type from java.sql.Types.
  - Parameters
    - **t**
      The java.sql.Types type number.
  - Returns
    The string representation of the java type.

- **static String nzTypeToString(int t)**
  Returns a String resprsentation of an NZ data type.
  - Parameters
    - **t**
      The type number.
  - Returns
    The string representation of the type.

NzaeEnvironment Interface Reference

This is the Java AE Environment API interface.

Public Member Functions

- String getFirstKey()
  Gets the first key.
- String getNextKey()
  Gets the next key.
- String getValue(String name)
  Gets the value for the key.

- boolean hasKey(String name)
  Determines whether the key exists in the environment.
- int size() Gets
  the size.

Detailed Description

This is the Java AE Environment API interface.

A reference to this interface can be obtained from the shaper, aggregate or function objects

Public Member Function Documentation

- **String getFirstKey()**
  Gets the first key.
  - Returns
    The key or NULL if none.

- **String getNextKey()**
  Gets the next key.
  - Returns
    The key or NULL if none.

- **String getValue(String name)**
  Gets the value for the key.
  - Parameters
    - **name**
      The environment name.
  - Returns
    The value.

- **boolean hasKey(String name)**
  Determines whether the key exists in the environment.
  - Parameters
    - **name**
      The environment name.
  - Returns
    TRUE if defined.

- **int size()**
  Gets the size.
  - Returns

The size.

NzaeException Class Reference

Unchecked Exception used thoughout the Java AE API.

Public Member Functions

- NzaeException()
- NzaeException(Throwable cause)
- NzaeException(String message, Throwable cause)
- NzaeException(String message)

Detailed Description

Unchecked Exception used thoughout the Java AE API.

Public Member Function Documentation

- **NzaeException()**
- **NzaeException(Throwable cause)**
- **NzaeException(String message, Throwable cause)**
- **NzaeException(String message)**

NzaeFactory Interface Reference

Used to create all objects and provide all information that is dependent on the underlying C imple-mentation of AE.

Public Member Functions

- NzaeRemoteProtocol createListener(NzaeConnectionPoint connectionPoint) Creates a new listener for remote AE connections.
- NzaeAgg getLocalAggregationApi(NzaeAggInitialization arg)
  Creates and returns the local instance of Aggregation NzaeAgg .
- NzaeApi getLocalApi()
  Returns the local API determined by how the AE was launched.
- Nzae getLocalFunctionApi(NzaeInitialization arg)

Creates and returns the local instance of Function Nzae .

- NzaeShaper getLocalShaperApi(NzaeShaperInitialization arg)
  Creates and returns the local instance of Shaper NzaeShaper .
- int getParentProcessId()
  Returns, the parent ID of this process, which is useful for debugging.
- int getProcessId()
  Returns the process ID, which is useful for debugging.
- boolean isLocal()
  Determines is the process is a local AE.
- boolean isRemote()
  Returns true if this is a remote AE process.
- NzaeConnectionPoint newConnectionPoint()
  Returns a new instance of a connection point.

Detailed Description

Used to create all objects and provide all information that is dependent on the underlying C implementation of AE.

Public Member Function Documentation

- **NzaeRemoteProtocol createListener(NzaeConnectionPoint connectionPoint)** Creates a new listener for remote AE connections.
  - Parameters
    - **NzaeConnectionPoint connectionPoint**
      The Connection Point.
  - Returns
    NzaeRemoteProtocol

    The Remote Protocol listener.

  Since one listener per unique connection name may be created, an AE may have multiple listeners.

- **NzaeAgg getLocalAggregationApi(NzaeAggInitialization arg)**
  Creates and returns the local instance of Aggregation NzaeAgg .
  - Parameters
    - **NzaeAggInitialization arg**
      The Aggregate initialization argument.
  - Returns
    NzaeAgg

    The AE Aggregate object.

- **NzaeApi getLocalApi()**

Returns the local API determined by how the AE was launched.

- Returns
  NzaeApi

  The API object.

(UDF,UDTF = function, or UDA = Aggregation, or function shaper and sizer.) This method is only valid for local AEs.

- **Nzae getLocalFunctionApi(NzaeInitialization arg)**
  Creates and returns the local instance of Function Nzae .
  - Parameters
    - **NzaeInitialization arg**
      The function initialization argument.
  - Returns
    Nzae

    The AE function object.

- **NzaeShaper getLocalShaperApi(NzaeShaperInitialization arg)**
  Creates and returns the local instance of Shaper NzaeShaper .
  - Parameters
    - **NzaeShaperInitialization arg**
      The Shaper initialization argument.
  - Returns
    NzaeShaper

    The AE Shaper object.

- **int getParentProcessId()**
  Returns, the parent ID of this process, which is useful for debugging.
  - Returns
    The Parent process ID.

- **int getProcessId()**
  Returns the process ID, which is useful for debugging.
  - Returns
    The Process ID.

- **boolean isLocal()**
  Determines is the process is a local AE.

- Returns
  TRUE if this is a local AE Process.

- **boolean isRemote()**
  Returns true if this is a remote AE process.
  - Returns
    True if this is a remote AE Process

- **NzaeConnectionPoint newConnectionPoint()**
  Returns a new instance of a connection point.
  - Returns
    NzaeConnectionPoint

    The Connection Point.

NzaeFieldInfo Interface Reference

This class contains information for a specific field (column) including NZ data type, Java SQL data type, and default Java lanaguage data types.

Public Member Functions

- int getJavaType()
  Return the Java data type of this field (column).
- int getNzType()
  Return the NZ data type of this field (column).
- boolean isBoolean()
  Default representation is a Boolean.
- boolean isByte()
  Default representation is a Byte.
- boolean isDate()
  Default representation is a java.sql.Date.
- boolean isDouble()
  Default representation is a Double.
- boolean isFloat()
  Default representation is a Float.
- boolean isInt()
  Default representation is an Int.
- boolean isLong()
  Default representation is a Long.
- boolean isNumber()

Default representation is an instance of Number.

- boolean isNumberDecimalType()
  Default representation is an instance of Number, BigDecimal.
- boolean isNumberFloatingPointType()
  Default representation is an instance of Number, floating point (Double or Float).
- boolean isNumberPrimitiveType()
  Default representation is an instance of Number, primitive type.
- boolean isShort()
  Default representation is a Short.
- boolean isString()
  Default representation is a String.
- boolean isTime()
  Default representation is a java.sql.Time.
- boolean isTimestamp()
  Default representation is a java.sql.Timestamp.

Detailed Description

This class contains information for a specific field (column) including NZ data type, Java SQL data type, and default Java lanaguage data types.

- See Also
  - getFieldInfo

Public Member Function Documentation

- **int getJavaType()**
  Return the Java data type of this field (column).
  - Returns
    Java type
  - See Also
    - java.sql.Types

- **int getNzType()**
  Return the NZ data type of this field (column).
  - Returns
    NZ type
  - See Also
    - NzaeDataTypes

- **boolean isBoolean()**

Default representation is a Boolean.

- Returns
  True if default representation is a Boolean

- **boolean isByte()**
  Default representation is a Byte.
  - Returns
    True if default representation is a Byte

- **boolean isDate()**
  Default representation is a java.sql.Date.
  - Returns
    True if default representation is a java.sql.Date

- **boolean isDouble()**
  Default representation is a Double.
  - Returns
    True if default representation is a Double

- **boolean isFloat()**
  Default representation is a Float.
  - Returns
    True if default representation is a Float

- **boolean isInt()**
  Default representation is an Int.
  - Returns
    True if default representation is an Int

- **boolean isLong()**
  Default representation is a Long.
  - Returns
    True if default representation is a Long

- **boolean isNumber()**
  Default representation is an instance of Number.
  - Returns
    True if default representation is an instance of Number

- **boolean isNumberDecimalType()**
  Default representation is an instance of Number, BigDecimal.
  - Returns
    True if default representation is an instance of Number or BigDecimal

- **boolean isNumberFloatingPointType()**
  Default representation is an instance of Number, floating point (Double or Float).
  - Returns
    True if default representation is an instance of Number or floating point

- **boolean isNumberPrimitiveType()**
  Default representation is an instance of Number, primitive type.
  - Returns
    True if default representation is an instance of Number or primitive type

- **boolean isShort()**
  Default representation is a Short.
  - Returns
    True if default representation is a Short

- **boolean isString()**
  Default representation is a String.
  - Returns
    True if default representation is a String

- **boolean isTime()**
  Default representation is a java.sql.Time.
  - Returns
    True if default representation is a java.sql.Time

- **boolean isTimestamp()**
  Default representation is a java.sql.Timestamp.
  - Returns
    True if default representation is a java.sql.Timestamp

NzaeInitialization Class Reference

Not implemented.

Detailed Description

Not implemented.

This class is a placeholder for future parameters for AE initialization.

- See Also
  - NzaeFactory

NzaeInterval Class Reference

This class is for the Interval data type.

Public Member Functions

- int compareTo(Object o)
  Compares two NzaeInterval objects.
- String toString()
  Returns the string representation.

Public Attributes

- month
  The month portion of the interval.
- time
  The time portion of the interval.

Detailed Description

This class is for the Interval data type.

Public Member Function Documentation

- **int compareTo(Object o)** Compares two NzaeInterval objects.
  - Parameters
    - **o**
      The object to be compared to.
  - Returns
    A negative integer if the object is less than the specified object, 0 is the objects are equal, or a pos-itive integer is the object is greater than the specified object.

  Returns an integer describing whether the object is less than, equal to, or greater than the specified ob-ject.

- **String toString()**
  Returns the string representation.
  - Returns
    The string representation.

Member Data Documentation

- long month
  Represents months and years.

- long time
  In microseconds. Represents everything but months and years.

NzaeLibrary Interface Reference

This is the Java AE Library API interface.

Public Member Functions

- NzaeLibraryInfo getLibraryInfo(String name, boolean caseSensitive, int type) Gets the specified library information.
- NzaeLibraryInfo getLocalLibraryInfo(int idx)
  Gets the Local library information at the specified index.
- NzaeLibraryInfo getParentLibraryInfo(int idx)
  Gets the Parent Library information at the specified index.
- int sizeLocalEntries()
  Gets the number of local entries.
- int sizeParentEntries()
  Gets the number of parent entries.

Static Public Attributes

- SEARCH_BOTH
  Searches both the parent and local contexts.
- SEARCH_LOCAL
  Searches the local context only.
- SEARCH_PARENT
  Searches the parent contect only.

Detailed Description

This is the Java AE Library API interface.

A reference to this interface can be obtained from the shaper, aggregate or function objects.

Public Member Function Documentation

- **NzaeLibraryInfo getLibraryInfo(String name, boolean caseSensitive, int type)** Gets the specified library information.
  - Parameters
    - **name**
      The name of the library.
    - **caseSensitive**
      If FALSE, performs a case-insensitive search
    - **type**
      Searches the Local context, the Parent context or Both.
  - Returns
    NzaeLibraryInfo

    he Library Information or NULL.

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

- **NzaeLibraryInfo getLocalLibraryInfo(int idx)**
  Gets the Local library information at the specified index.
  - Parameters
    - **idx**
      The index to look up.
  - Returns
    NzaeLibraryInfo

    The Library Information or NULL.

- **NzaeLibraryInfo getParentLibraryInfo(int idx)**
  Gets the Parent Library information at the specified index.
  - Parameters
    - **idx**
      The index to look up.
  - Returns
    NzaeLibraryInfo

    The Library Information or NULL.

- **int sizeLocalEntries()**
  Gets the number of local entries.
  - Returns
    The number of local entries.

  The Local Entries are those associated with the AE.

- **int sizeParentEntries()**
  Gets the number of parent entries.
  - Returns
    The number of local entries.

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

Static Member Data Documentation

- final int SEARCH_BOTH= 0
  Searches both the parent and local contexts.

- final int SEARCH_LOCAL= 1
  Searches the local context only.

- final int SEARCH_PARENT= 2
  Searches the parent contect only.

NzaeLibraryInfo Class Reference

Provides AE Library Information.

Public Attributes

- autoLoad
  The library is autoloaded.
- libraryFullPath
  The library path.
- libraryName
  The library name.

Detailed Description

Provides AE Library Information.

Member Data Documentation

- boolean autoLoad
  The library is autoloaded.

- String libraryFullPath
  The library path.

- String libraryName
  The library name.

NzaeMessageHandler Interface Reference

This class handles Nzae messages.

Public Member Functions

- void evaluate(Nzae api, NzaeRecord input, NzaeRecord result)
  Processes one row of input and produces one row of output.

Detailed Description

This class handles Nzae messages.

- See Also
  - run

Public Member Function Documentation

- **void evaluate(Nzae api, NzaeRecord input, NzaeRecord result)**
  Processes one row of input and produces one row of output.
  - Parameters
    - **Nzae api**
      The function object.
    - **NzaeRecord input**
      The input record.
    - **NzaeRecord result**
      The result record.

  Used for scalar functions and some table functions. Scalar functions use only a single field in the result.

NzaeMetadata Class Reference

This class contains metadata about the AE, including input and output column attributes.

Public Member Functions

- int getCorrelationType()
  Determines the correlation type used to invoke the UDTF.
- int getInputColumnCount()
  Gets the number of input columns.
- int getInputNzType(int index)
  Gets the input data type from NzaeDataTypes .
- int getInputScale(int index)
  The scale of the input column.
- int getInputSize(int index)
  Gets the size of the input column.
- int getOutputColumnCount()
  Gets the number of output columns.
- int getOutputNzType(int index)
  Gets the output data types from NzaeDataTypes .
- int getOutputSize(int index)
  Gets the size of the output column.
- boolean hasFinal()
  Determines if the UDTF is invoked with TABLE WITH FINAL.
- boolean hasOrder()
  Determines if the UDTF is invoked with ORDER BY in OVER.
- boolean hasOver()
  Determines if the UDTF is invoked with OVER.
- boolean hasPartition()
  Determines if the UDTF is invoked with PARTITION BY in OVER.
- boolean inputIsConstant(int index)
  Specifies if the value of the specified column is constant for all rows.
- boolean isOneOutputRowRestriction()
  Determines if the function is UDTF or UDF.
- int outputScale(int index)
  Gets the scale of the output column.

Static Public Attributes

- NZAE_INNER_CORRELATION
  Inner correlation table function correlation type.
- NZAE_LEFT_CORRELATION
  Left correlation table function correlation type.
- NZAE_UNCORRELATED
  Uncorrelated table function correlation type.
- NZAE_UNKNOWN_CORRELATION_TYPE
  Unknown table function correlation type.

Detailed Description

This class contains metadata about the AE, including input and output column attributes.

Column indexes are zero-based.

- See Also
  - getMetadata
  - NzaeDataTypes

Public Member Function Documentation

- **int getCorrelationType()**
  Determines the correlation type used to invoke the UDTF.
  - Returns
    The correlation type.

- **int getInputColumnCount()**
  Gets the number of input columns.
  - Returns
    The number of input columns.

- **int getInputNzType(int index)**
  Gets the input data type from NzaeDataTypes .
  - Parameters
    - **index**
      The input index.
  - Returns
    The input data type.

- **int getInputScale(int index)**
  The scale of the input column.
  - Parameters

- **index**
  The input index.

- Returns
  The scale of the input column.

- **int getInputSize(int index)**
  Gets the size of the input column.
  - Parameters
    - **index**
      The input index.
  - Returns
    The length for string type, the precision for numeric type.

  Length for string type, precision for numeric type.

- **int getOutputColumnCount()**
  Gets the number of output columns.
  - Returns
    The number of output columns.

- **int getOutputNzType(int index)**
  Gets the output data types from NzaeDataTypes .
  - Parameters
    - **index**
      The output index.
  - Returns
    The output data type.

- **int getOutputSize(int index)**
  Gets the size of the output column.
  - Parameters
    - **index**
      The output index.
  - Returns
    The length for string type, the precision for numeric type.

  Length for string type, precision for numeric type.

- **boolean hasFinal()**
  Determines if the UDTF is invoked with TABLE WITH FINAL.

- Returns
  TRUE if the table function is invoked using TABLE WITH FINAL.

- **boolean hasOrder()**
  Determines if the UDTF is invoked with ORDER BY in OVER.
  - Returns
    TRUE if the table function is invoked using ORDER BY.

- **boolean hasOver()**
  Determines if the UDTF is invoked with OVER.
  - Returns
    TRUE if the table function is invoked using OVER.

- **boolean hasPartition()**
  Determines if the UDTF is invoked with PARTITION BY in OVER.
  - Returns
    TRUE if the table function is invoked using PARTITION BY.

- **boolean inputIsConstant(int index)**
  Specifies if the value of the specified column is constant for all rows.
  - Parameters
    - **index**
      The input index.
  - Returns
    TRUE if the value of the column is constant for all rows.

  If method returns TRUE then the value of this column is constant for all rows.

- **boolean isOneOutputRowRestriction()**
  Determines if the function is UDTF or UDF.
  - Returns
    TRUE if the function is scalar; FALSE if the function is table.

  Returns true if the function can only return one output row per input row.

- **int outputScale(int index)**
  Gets the scale of the output column.
  - Parameters
    - **index**
      The output index.
  - Returns

The scale of the output column.

Static Member Data Documentation

- final int NZAE_INNER_CORRELATION= 2
  Inner correlation table function correlation type.
- final int NZAE_LEFT_CORRELATION= 3
  Left correlation table function correlation type.
- final int NZAE_UNCORRELATED= 1
  Uncorrelated table function correlation type.
- final int NZAE_UNKNOWN_CORRELATION_TYPE=
  0 Unknown table function correlation type.

NzaeRecord Interface Reference

A record corresponds to an input or output database row.

Public Member Functions

- boolean compatibleWithOutput()
  Determines if the record can be used as an output result.
- NzaeRecord duplicate()
  Creates a deep copy duplicate.
- Object getField(int index)
  Returns the field value as an Object.
- byte [] getFieldAsBinary(int index)
  Returns the field value cast to a byte [], that is a byte array.
- Boolean getFieldAsBoolean(int index)
  Returns the field value cast to a Boolean.
- java.sql.Date getFieldAsDate(int index)
  Returns the field value cast to a java.sql.Date.
- BigDecimal getFieldAsDecimal(int index)
  Returns the field value cast to a BigDecimal.
- NzaeInterval getFieldAsInterval(int index)
  Returns the field value cast to a NzaeInterval .

- Number getFieldAsNumber(int index)
  Returns the field value cast to a Number.
- String getFieldAsString(int index)
  Returns the field value cast to a String.
- java.sql.Time getFieldAsTime(int index)
  Returns the field value cast to a java.sql.Time.
- java.sql.Timestamp getFieldAsTimestamp(int index)
  Returns the field value cast to a java.sql.Timestamp.
- NzaeTimeTz getFieldAsTimeTz(int index)
  Returns the field value cast to a NzaeTimeTz .
- NzaeFieldInfo getFieldInfo(int index)
  Returns instance of NzaeFieldInfo interface.
- boolean isReadOnly()
  Determines if the record is read-only.
- void setField(int index, Object value)
  Sets the field value as an Object.
- void setFields(NzaeRecord rec)
  Sets all the fields from a record consisting of the same number and type of fields.
- int size()
  Gets the number of fields.
- String toString()
  The string representation of the record.

Detailed Description

A record corresponds to an input or output database row.

Each record consists of one or more fields, which correspond to columns. Fields are addresed by an int index which is zero-based. GetFieldX methods may return null. Objects applied to get and set methods must match or be convertable to or from the data type of the field (column).

- See Also
  - next
  - createOutputRecord
  - NzaeAggMessageHandler

Public Member Function Documentation

- **boolean compatibleWithOutput()**
  Determines if the record can be used as an output result.
  - Returns
    TRUE if it can be used as an output result.

  Can be an output for a function, or a state or final result record for an aggregate.

- **NzaeRecord duplicate()**
  Creates a deep copy duplicate.
  - Returns
    NzaeRecord

    The duplicate record.

  The duplicate cannot be read-only.

- **Object getField(int index)**
  Returns the field value as an Object.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as an Object.

- **byte [] getFieldAsBinary(int index)**
  Returns the field value cast to a byte [], that is a byte array.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as an byte array.

  The field must have already been set to use a binary interpretation using Nzae setInterpretCharAsBinary or NzaeAgg setInputInterpretCharAsBinary / setStateInterpretCharAsBinary.
  - See Also
    - setInterpretCharAsBinary
    - setInputInterpretCharAsBinary
    - setStateInterpretCharAsBinary

- **Boolean getFieldAsBoolean(int index)**
  Returns the field value cast to a Boolean.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as a Boolean.

- **java.sql.Date getFieldAsDate(int index)**
  Returns the field value cast to a java.sql.Date.

- Parameters
  - **index**
    The field index.
- Returns
  The field as a java.sql.Date.

- **BigDecimal getFieldAsDecimal(int index)**
  Returns the field value cast to a BigDecimal.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as a BigDecimal.

- **NzaeInterval getFieldAsInterval(int index)**
  Returns the field value cast to a NzaeInterval .
  - Parameters
    - **index**
      The field index.
  - Returns
    NzaeInterval

    The field as an NzaeInterval .

- **Number getFieldAsNumber(int index)**
  Returns the field value cast to a Number.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as a Number.

- **String getFieldAsString(int index)**
  Returns the field value cast to a String.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as a String.

- **java.sql.Time getFieldAsTime(int index)**

Returns the field value cast to a java.sql.Time.

- Parameters
  - **index**
    The field index.
- Returns
  The field as a java.sql.Time.

- **java.sql.Timestamp getFieldAsTimestamp(int index)**
  Returns the field value cast to a java.sql.Timestamp.
  - Parameters
    - **index**
      The field index.
  - Returns
    The field as a java.sql.Timestamp.

- **NzaeTimeTz getFieldAsTimeTz(int index)**
  Returns the field value cast to a NzaeTimeTz .
  - Parameters
    - **index**
      The field index.
  - Returns
    NzaeTimeTz

    The field as an NzaeTimeTz .

- **NzaeFieldInfo getFieldInfo(int index)**
  Returns instance of NzaeFieldInfo interface.
  - Parameters
    - **index**
      The field index.
  - Returns
    NzaeFieldInfo

    The FieldInfo object.

- **boolean isReadOnly()**
  Determines if the record is read-only.
  - Returns
    TRUE if the record is read-only.

- **void setField(int index, Object value)**
  Sets the field value as an Object.
  - Parameters
    - **index**
      The field index.
    - **value**
      The field value.

  NULL is a valid value.

- **void setFields(NzaeRecord rec)**
  Sets all the fields from a record consisting of the same number and type of fields.
  - Parameters
    - **NzaeRecord**
      rec Record.

- **int size()**
  Gets the number of fields.
  - Returns
    The number of fields.

- **String toString()**
  The string representation of the record.
  - Returns
    The string representation.

NzaeRemoteProtocol Interface Reference

Netezza Analytic Executables.

Public Member Functions

- NzaeApi acceptConnection()
  Accepts a new connection.
- NzaeApi acceptConnectionWithTimeout(int timeoutMilliseconds) Accepts a new connection with timeout.
- void close() Closes the listener.
- NzaeRemoteProtocolCallback getCallbackHandler()
  Gets the Remote Protocol Callback.
- void setCallbackHandler(NzaeRemoteProtocolCallback handler)

Sets the Remote Protocol Callback.

Detailed Description

Netezza Analytic Executables.

AE Remote Protocol. This class receives new AE connection requests.

Public Member Function Documentation

- **NzaeApi acceptConnection()**
  Accepts a new connection.
  - Returns
    NzaeApi The
    new API.

- **NzaeApi acceptConnectionWithTimeout(int timeoutMilliseconds)** Accepts a new connection with timeout.
  - Parameters
    - **timeoutMilliseconds**
      The timeout.
  - Returns
    NzaeApi The
    new API.

- **void close()** Closes
  the listener.

- **NzaeRemoteProtocolCallback getCallbackHandler()**
  Gets the Remote Protocol Callback.
  - Returns
    NzaeRemoteProtocolCallback
    The callback.

- **void setCallbackHandler(NzaeRemoteProtocolCallback handler)** Sets the Remote Protocol Callback.
  - Parameters
    - **NzaeRemoteProtocolCallback**
      handler The remote protocol handler.

The handler is owned by NzaeRemoteProtocol .

NzaeRemoteProtocolCallback Interface Reference

Netezza Analytic Executables.

Public Member Functions

- NzaeCallbackResult execute(int code, byte[] data) The callback executor.

Static Public Attributes

- CALLBACK_CONTROL
  Control Data.
- CALLBACK_PING
  Ping.
- CALLBACK_REQUEST
  Request.
- CALLBACK_SIGNAL
  Signal.
- CALLBACK_STATUS
  Get Status.
- CALLBACK_STOP
  Stop.

Detailed Description

Netezza Analytic Executables.

AE Remote Protocol Callback. This class handles the remote protocol callback requests.

Public Member Function Documentation

- **NzaeCallbackResult execute(int code, byte[] data)** The callback executor.
  - Parameters
    - **code**
      The callback type.
    - **data** The
      data.
  - Returns
    NzaeCallbackResult

The callback result.

This function can handle the following types: CALLBACK_STATUS, CALLBACK_STOP, CALLBACK_CONTROL

If this method throws an exeception, it causes the remote protocol accept method to error out; data is likely empty for Stop and Status.

The executor fills out the result structure with a returnCode equal to 0 for normal completion.

Static Member Data Documentation

- final int CALLBACK_CONTROL=
  4 Control Data.
- final int CALLBACK_PING=
  1 Ping.
- final int CALLBACK_REQUEST=
  0 Request.
- final int CALLBACK_SIGNAL=
  5 Signal.
- final int CALLBACK_STATUS=
  2 Get Status.
- final int CALLBACK_STOP=
  3 Stop.

NzaeRuntime Class Reference

Constant runtime information.

Public Member Functions

- int getAdapterType()
  Get Adapter type.
- long getAeCallId()
  Get SQL Function Call ID.
- long getAeQueryId()

Get Query ID.

- int getDataSliceId()
  Get Dataslice ID.
- int getHardwareId()
  Get hardware ID.
- int getLocus()
  Get Locus.
- int getNumberDataSlices()
  Get Number of data slices.
- int getNumberSpus()
  Get Number of SPUs.
- int getSessionId()
  Get Session ID.
- long getSuggestedMemoryLimit()
  Get Memory Limit.
- long getTransactionId()
  Get Transaction ID.
- String getUsername()
  Get User name.
- boolean getUserQuery()
  Is User Query.
- String toString()
  Return string representation.

Static Public Attributes

- NZAE_ADAPTER_OTHER UDX
  Adapter type unknown.
- NZAE_ADAPTER_UDA
  UDX Adapter type UDA.
- NZAE_ADAPTER_UDF
  UDX Adapter type UDF.
- NZAE_ADAPTER_UDTF
  UDX Adapter type UDTF.
- NZAE_LOCUS_DBOS Executing
  in locus dbos (host).
- NZAE_LOCUS_POSTGRES
  Executing in locus postgres (host).
- NZAE_LOCUS_SPU
  Executing in locus SPU (s-blade).

Detailed Description

Constant runtime information.

Public Member Function Documentation

- **int getAdapterType()**
  Get Adapter type.
  - Returns
    Adapter type
  - See Also
    - NZAE_ADAPTER_UDTF
    - NZAE_ADAPTER_UDF
    - NZAE_ADAPTER_UDA
    - NZAE_ADAPTER_OTHER

- **long getAeCallId()**
  Get SQL Function Call ID.
  - Returns
    SQL Function Call ID.

- **long getAeQueryId()**
  Get Query ID.
  - Returns
    Query ID.

- **int getDataSliceId()**
  Get Dataslice ID.
  - Returns
    Dataslice ID.

- **int getHardwareId()**
  Get hardware ID.
  - Returns
    Hardware ID.

- **int getLocus()**
  Get Locus.
  - Returns

Locus of execution

- See Also
  - NZAE_LOCUS_POSTGRES
  - NZAE_LOCUS_DBOS
  - NZAE_LOCUS_SPU

- **int getNumberDataSlices()**
  Get Number of data slices.
  - Returns
    Number of data slices

- **int getNumberSpus()**
  Get Number of SPUs.
  - Returns Number
    of SPUs

- **int getSessionId()**
  Get Session ID.
  - Returns
    Session ID

- **long getSuggestedMemoryLimit()**
  Get Memory Limit.
  - Returns
    Memory Limit.

  This is an advisory limit only.

- **long getTransactionId()**
  Get Transaction ID.
  - Returns
    Transaction ID.

- **String getUsername()**
  Get User name.
  - Returns
    User name

- **boolean getUserQuery()**
  Is User Query.

- Returns
  True if a user query as opposed to a JIT state or other prep query

- **String toString()**
  Return string representation.
  - Returns
    String representation

Static Member Data Documentation

- final int NZAE_ADAPTER_OTHER=
  0 UDX Adapter type unknown.

- final int NZAE_ADAPTER_UDA=
  3 UDX Adapter type UDA.

- final int NZAE_ADAPTER_UDF=
  2 UDX Adapter type UDF.

- final int NZAE_ADAPTER_UDTF=
  1 UDX Adapter type UDTF.

- final int NZAE_LOCUS_DBOS= 1
  Executing in locus dbos (host).

- final int NZAE_LOCUS_POSTGRES= 0
  Executing in locus postgres (host).

- final int NZAE_LOCUS_SPU= 2
  Executing in locus SPU (s-blade).

NzaeShaper Interface Reference

This is the main Java AE Shaper API interface.

Inherits NzaeBase

Public Member Functions

- void addOutputColumn(int type, String columnName)
  Adds an output column of a non-string or non-numeric type.
- void addOutputColumnNumeric(int type, String columnName, int precision, int scale) Adds an output column of a numeric type.
- void addOutputColumnString(int type, String columnName, int size) Adds an output column of a string type.
- boolean catalogIsUpper()
  Determines if the catalog is upper case.
- void close()
  Closes the AE and releases its resources.
- int getApiType()
  Returns the API Constant for this interface.
- NzaeDataTypes getDataTypes()
  Gets the NZ Data Type constants.
- NzaeEnvironment getEnvironment() Gets environment information for the AE.
- NzaeLibrary getLibrary()
  Gets library information for the AE.
- NzaeShaperMessageHandler getMessageHandler()
  Returns the message handler class object.
- NzaeMetadata getMetadata()
  Gets metadata information for the AE.
- int getNumOutputColumns()
  Gets the number of output columns.
- NzaeShaperOutputColumnInfo getOutputColumnInfo(int idx) Gets information about the output columns.
- NzaeRuntime getRuntime()
  Gets runtime information for the AE, including information about the Netezza system.
- NzaeRecord inputRow()
  Gets the input row.
- void log(int logLevel, String message)
  Log the specified message at the specified log level.
- int outputType()
  Gets the return type for a sizer (UDF).
- void ping()
  Indicates that the AE is still active and not hanging.
- void run(NzaeShaperMessageHandler messageHandler)
  Begins higher-level shaper processing.
- int size()

Gets the output field size.

- void update()
  Signals that the shaper is done.
- void userError(String message)
  Indicates the AE has encountered an error condition.

Static Public Attributes

- LOG_DEBUG
  Log level debug.
- LOG_TRACE
  Log level trace.

Detailed Description

This is the main Java AE Shaper API interface.

A reference to this interface can be obtained from static methods in the NzaeFactory class.

- See Also
  - NzaeFactory

Public Member Function Documentation

- **void addOutputColumn(int type, String columnName)** Adds an output column of a non-string or non-numeric type.
  - Parameters
    - **type**
      The column type; cannot be string or numeric.
    - **columnName** The column name.

- **void addOutputColumnNumeric(int type, String columnName, int precision, int scale)** Adds an output column of a numeric type.
  - Parameters
    - **type**
      The column type; must be numeric.
    - **columnName** The column name.
    - **precision**
      The column precision.
    - **scale**
      The column scale.

- **void addOutputColumnString(int type, String columnName, int size)** Adds an output column of a string type.
  - Parameters
    - **type**
      The column type; must be string.
    - **columnName** The
      column name.
    - **size**
      The column size.

- **boolean catalogIsUpper()**
  Determines if the catalog is upper case.
  - Returns
    TRUE if the catalog is upper case.

- **void close()**
  Closes the AE and releases its resources.

- **int getApiType()**
  Returns the API Constant for this interface.
  - Returns
    The values defined in NzaeApi .

- **NzaeDataTypes getDataTypes()**
  Gets the NZ Data Type constants.
  - Returns
    NzaeDataTypes

    The data types object.

- **NzaeEnvironment getEnvironment()** Gets
  environment information for the AE.
  - Returns
    NzaeEnvironment

    The instance of NzaeEnvironment .

- **NzaeLibrary getLibrary()**
  Gets library information for the AE.
  - Returns

NzaeLibrary

The instance of NzaeLibrary .

- **NzaeShaperMessageHandler getMessageHandler()**
  Returns the message handler class object.
  - Returns
    NzaeShaperMessageHandler

    The instance of the Shaper handler.

- **NzaeMetadata getMetadata()**
  Gets metadata information for the AE.
  - Returns
    NzaeMetadata

    The instance of NzaeMetadata .

- **int getNumOutputColumns()**
  Gets the number of output columns.
  - Returns
    The number of columns.

- **NzaeShaperOutputColumnInfo getOutputColumnInfo(int idx)** Gets information about the output columns.
  - Parameters
    - **idx**
      The index of the column to get.
  - Returns
    NzaeShaperOutputColumnInfo

    The column information.

- **NzaeRuntime getRuntime()**
  Gets runtime information for the AE, including information about the Netezza system.
  - Returns
    NzaeRuntime

    The instance of NzaeRuntime .

- **NzaeRecord inputRow()**
  Gets the input row.

- Returns
  NzaeRecord

  An instance of NzaeRecord .

- **void log(int logLevel, String message)**
  Log the specified message at the specified log level.
  - Parameters
    - **logLevel**
      The log level constant.
    - **message**
      The message to log.

- **int outputType()**
  Gets the return type for a sizer (UDF).
  - Returns
    The return type.

  Must either be one of the string types or NUMERIC128

- **void ping()**
  Indicates that the AE is still active and not hanging.

- **void run(NzaeShaperMessageHandler messageHandler)** Begins higher-level shaper processing.
  - Parameters
    - **NzaeShaperMessageHandler messageHandler** The message handler.

  This is an alternative to writing custom shaper code.

- **int size()**
  Gets the output field size.
  - Returns
    The output field size.

- **void update()**
  Signals that the shaper is done.

- **void userError(String message)**
  Indicates the AE has encountered an error condition.
  - Parameters

- **message**
  The message to send back to the Netezza system.

Implies NzaeDone.

Static Member Data Documentation

- final int LOG_DEBUG=
  2 Log level debug.

- final int LOG_TRACE=
  1 Log level trace.

NzaeShaperInitialization Class Reference

Not implemented.

Detailed Description

Not implemented.

This class is a placeholder for future parameters for AE shaper initialization.

- See Also
  - NzaeFactory

NzaeShaperMessageHandler Interface Reference

This class handles NzaeShaper messages.

Public Member Functions

- void shaper(NzaeShaper
  api) Runs the shaper logic.

Detailed Description

This class handles NzaeShaper messages.

- See Also
  - run

Public Member Function Documentation

- **void shaper(NzaeShaper api)** Runs the shaper logic.
  - Parameters
    - **NzaeShaper api**
      The Shaper object.

NzaeShaperOutputColumnInfo Class Reference

Provides Shaper Output Information.

Public Attributes

- columnName
  Column name.
- precision
  Column precision, if numeric.
- scale
  Column scale, if numeric.
- size
  Column size, if string .
- type Column
  type.

Detailed Description

Provides Shaper Output Information.

Member Data Documentation

- String columnName
  Column name.
- int precision
  Column precision, if numeric.
- int scale
  Column scale, if numeric.
- int size
  Column size, if string .

- int type
  Column type.

NzaeTimeTz Class Reference

This class is for the TimeTz data type.

Public Member Functions

- int compareTo(Object o) Compares two NzaeTimeTz objects.
- String toString()
  Returns the string representation.

Public Attributes

- offsetMinutes
  The Time Zone portion of TimeTz.
- time
  The time portion of TimeTz.

Detailed Description

This class is for the TimeTz data type.

Public Member Function Documentation

- **int compareTo(Object o)** Compares two NzaeTimeTz objects.
  - Parameters
    - **o**
      The object to be compared to.
  - Returns
    A negative integer if the object is less than the specified object, 0 is the objects are equal, or a positive integer is the object is greater than the specified object.

  Returns an integer describing whether the object is less than, equal to, or greater than the specified object.

- **String toString()**
  Returns the string

  representation. ▲ Returns

The string representation.

Member Data Documentation

- int offsetMinutes
  The Offset is in minutes.

- java.sql.Time time
  The time portion of TimeTz.

NzaeUtil Class Reference

This class contains static utility methods that are not dependent on the main NzaeFactory interface.

Static Public Member Functions

- static void logException(Throwable t)
  Logs the exception to the process-specific AE log file or standard error.
- static void logException(Throwable t, String appMessage)
  Logs the exception to the process-specific AE log file or standard error.
- static void logJavaSystemProperties()
  Dumps the JVM System Properties to the process-specific AE log file or standard error.

Detailed Description

This class contains static utility methods that are not dependent on the main NzaeFactory interface.

Static Public Member Function Documentation

- **static void logException(Throwable t)**
  Logs the exception to the process-specific AE log file or standard error.
  - Parameters
    - **t**
      The throwable object to log.

- **static void logException(Throwable t, String appMessage)**
  Logs the exception to the process-specific AE log file or standard error.
  - Parameters
    - **t**
      The throwable object to log.
    - **appMessage**

An application-specific message for the log.

- **static void logJavaSystemProperties()**
  Dumps the JVM System Properties to the process-specific AE log file or standard error.

Source Class Reference

Class used to instantiate a factory.

Static Public Member Functions

- static NzaeFactory getFactory()
  Gets the Factory instance.

Detailed Description

Class used to instantiate a factory.

Static Public Member Function Documentation

- **static NzaeFactory getFactory()**
  Gets the Factory instance.
  - Returns
    NzaeFactory

    The Factory instance.

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 docu-ment. 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 Intellec-tual 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 pro-visions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IM-PLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MER-CHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of ex-press 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 publica-tion. 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 es-timated through extrapolation. Actual results may vary. Users of this document should verify the ap-plicable 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 in-formation 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 program-ming 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 operat-ing 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 servi-cing. 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, pursu-ant 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 interfer-ence 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越すことがあります。この場合には使用者が適切な対策を講ずるう要求されることがあります。

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U