IBM® Netezza® Analytics
Release 3.3.5.0

Fortran Analytic Executables API Reference

Part Number 00X6335-00 Rev. 1

Note: Before using this information and the product that it supports, read the information in "Notices and Trademarks" on page 57.

Contents

Preface

1 Module Documentation

Notices and Trademarks

Index

Preface

This guide provides an API reference for Fortran AE programmers.

Audience for This Guide

The *Fortran Analytic Executables API Reference* is written for programmers who intend to create Analytic Executables for IBM Netezza Analytics using Fortran. This guide does not provide a tutorial 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 Fortran AE API, which is a language adapter provided as part of IBM Netezza Analytics. The Fortran AE API provides programmatic access to the AE interface for Fortran 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 backslash 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:

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 request, 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 documentation. Please send us an e-mail message at netezza-doc@wwpdl.vnet.ibm.com and include the following 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

Error Handling Functions

Functions/Subroutines

- subroutine nzaeGetLastErrorCode(handle, errorCode)
  Returns the error code of the last AE error.
- subroutine nzaeGetLastErrorText(handle, errorText)
  Returns the error message text of the last AE error.
- subroutine nzaeUserError(handle, errorText)
  Notifies the Netezza system that the AE encountered and error and it should not return any results.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetLastErrorCode(handle, errorCode)**
  Returns the error code of the last AE error.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **errorCode**
      (integer) The return value of this function.

  The error code that is returned is defined in the C header file, /nz/export/ae/adapters/system/2/sys/include/nzaeusercodes.h.

- **subroutine nzaeGetLastErrorText(handle, errorText)**
  Returns the error message text of the last AE error.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **errorText**
      (character*) The return value of this function.

- **subroutine nzaeUserError(handle, errorText)**
  Notifies the Netezza system that the AE encountered and error and it should not return any results.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **errorText**
      (character*) The error to be displayed in nzsql.

  The maximum length of the error text is approximately 250 characters. Only the first call to this function is displayed.

Shared Library Functions

Functions/Subroutines

- subroutine nzaeGetLibraryFullPath(handle, name, caseSensitive, path)
  Returns the path to the shared library associated with the specified name.
- subroutine nzaeGetNumberOfSharedLibraries(handle, number)
  Returns the number of shared libraries registered in the Netezza system and available to this run of the AE.
- subroutine nzaeGetNumberOfSharedLibrariesForProcess(handle, number)
  Returns the number of shared libraries registered in the Netezza system and available to the AE process.
- subroutine nzaeGetSharedLibraryInfo(handle, index, name, path, autoload)
  Returns the shared library information for the running AE.
- subroutine nzaeGetSharedLibraryInfoForProcess(handle, index, name, path, autoload)
  Returns the shared library information for the process of the running AE.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetLibraryFullPath(handle, name, caseSensitive, path)**
  Returns the path to the shared library associated with the specified name.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **name**
      (character*) The name of the shared library.
    - **caseSensitive**
      (integer) The value is 1 for a case sensitive look up; 0 for a case-insensitive look up.
    - **path**
      (character*) The file path of the shared library.

  The path is an empty string if the shared library is not found.

- **subroutine nzaeGetNumberOfSharedLibraries(handle, number)**
  Returns the number of shared libraries registered in the Netezza system and available to this run of the AE.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **number**
      (integer) The number of available shared libraries.

  This function is used while running an AE. Use the for-process version to handle the shared libraries while setting up a remote AE.

- **subroutine nzaeGetNumberOfSharedLibrariesForProcess(handle, number)**
  Returns the number of shared libraries registered in the Netezza system and available to the AE process.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **number**
      (integer) The number of available shared libraries.

  This routine is generally used while setting up remote AEs. Use the non-for-process version to handle shared libraries while setting up a remote AE.

- **subroutine nzaeGetSharedLibraryInfo(handle, index, name, path, autoload)**
  Returns the shared library information for the running AE.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **index**
      (integer) The index of the shared library.

- **name**
  (character*) The registered name of the shared library.
- **path**
  (character*) The full path to the shared library.
- **autoload**
  (integer) The value is 1 if the library is set to auto-load; 0 otherwise.

This function is used while running an AE. Use the for-proccess version to handle the shared libraries while setting up a remote AE.

- **subroutine nzaeGetSharedLibraryInfoForProcess(handle, index, name, path, autoload)**
  Returns the shared library information for the process of the running AE.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **index**
      (integer) The index of the shared library.
    - **name**
      (character*) The registered name of the shared library.
    - **path**
      (character*) The full path to the shared library.
    - **autoload**
      (integer) The value is 1 if the library is set to auto-load; 0 otherwise.

  This routine is generally used while setting up remote AEs. Use the non-for-process version to handle shared libraries while setting up a remote AE.

Metadata Functions

Functions/Subroutines

- subroutine nzaeGetInputScale(handle, columnIndex, result)
  Determines the scale of an input column.
- subroutine nzaeGetInputSize(handle, columnIndex, result)
  Determines the size of an input string column.
- subroutine nzaeGetInputType(handle, columnIndex, result)
  Determines the data type of a gven input column.
- subroutine nzaeGetNumberOfInputColumns(handle, result)
  Determines the number of columns in the input.
- subroutine nzaeGetNumberOfOutputColumns(handle, result)
  Determines the number of columns in the output.

- subroutine nzaeGetOutputScale(handle, columnIndex, result)
  Determines the scale of an output column.
- subroutine nzaeGetOutputSize(handle, columnIndex, result)
  Determines the size of an output string column.
- subroutine nzaeGetOutputType(handle, columnIndex, result)
  Determines the data type of a gven output column.
- subroutine nzaeIsDataInnerCorrelated(handle, result)
  Determines if the AE was invoked with inner-correlated data.
- subroutine nzaeIsDataLeftCorrelated(handle, result)
  Determines if the AE was invoked with left-correlated data.
- subroutine nzaeIsDataUncorrelated(handle, result)
  Determines if the AE was invoked with uncorrelated data.
- subroutine nzaeIsInvokedWithOrderByClause(handle, result)
  Determines if the AE was invoked with an ORDER BY clause.
- subroutine nzaeIsInvokedWithOverClause(handle, result)
  Determines if the AE was invoked with an OVER clause.
- subroutine nzaeIsInvokedWithPartitionByClause(handle, result)
  Determines if the AE was invoked with a PARTITION BY clause.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetInputScale(handle, columnIndex, result)**
  Determines the scale of an input column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**
      (integer) The scale of the input at the specified column index.

- **subroutine nzaeGetInputSize(handle, columnIndex, result)**
  Determines the size of an input string column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**

(integer) The size of the input at the specified column index.

- **subroutine nzaeGetInputType(handle, columnIndex, result)**
  Determines the data type of a gven input column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**
      (integer) The type of the input at the specified column index.

  The possible types can be found at /nz/kit/sys/include/nzudsudxtypes.h.

- **subroutine nzaeGetNumberOfInputColumns(handle, result)**
  Determines the number of columns in the input.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The number of columns in the input.

- **subroutine nzaeGetNumberOfOutputColumns(handle, result)**
  Determines the number of columns in the output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The number of columns in the output.

- **subroutine nzaeGetOutputScale(handle, columnIndex, result)**
  Determines the scale of an output column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**
      (integer) The scale of the output at the specified column index.

This function works for function AEs as well as during the "final-result" process state of aggregate AEs.

- **subroutine nzaeGetOutputSize(handle, columnIndex, result)**
  Determines the size of an output string column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**
      (integer) The size of the output at the specified column index.

  This function works for function AEs as well as during the "final-result" process state of aggregate AEs.

- **subroutine nzaeGetOutputType(handle, columnIndex, result)**
  Determines the data type of a gven output column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to interrogate.
    - **result**
      (integer) The type of the output at the specified column index.

  The possible types can be found at /nz/kit/sys/include/nzudsudxtypes.h. This function works for function AEs as well as during the "final-result" process state of aggregate AEs.

- **subroutine nzaeIsDataInnerCorrelated(handle, result)**
  Determines if the AE was invoked with inner-correlated data.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The value is 1 if the the AE was invoked such that the data is inner-correlated; 0 otherwise.

- **subroutine nzaeIsDataLeftCorrelated(handle, result)**
  Determines if the AE was invoked with left-correlated data.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**

(integer) The value is 1 if the the AE was invoked such that the data is left-correlated; 0 otherwise.

- **subroutine nzaeIsDataUncorrelated(handle, result)**
  Determines if the AE was invoked with uncorrelated data.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The value is 1 if the the AE was invoked such that the data is correlated; 0 otherwise.

- **subroutine nzaeIsInvokedWithOrderByClause(handle, result)**
  Determines if the AE was invoked with an ORDER BY clause.
  - Parameters
    - **handle**
      (integer) The data type in question.
    - **result**
      (integer) The value is 1 if the the AE was invoked with a ORDER BY clause; 0 otherwise.

- **subroutine nzaeIsInvokedWithOverClause(handle, result)**
  Determines if the AE was invoked with an OVER clause.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The value is 1 if the the AE was invoked with an OVER clause; 0 otherwise.

- **subroutine nzaeIsInvokedWithPartitionByClause(handle, result)**
  Determines if the AE was invoked with a PARTITION BY clause.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The value is 1 if the the AE was invoked with a PARTITION BY clause; 0 otherwise.

Aggregate Functions

Functions/Subroutines

- subroutine nzaeGetNextAggregation(handle)
  Gets the next aggregation state.
- subroutine nzaeGetNumberOfStateColumns(handle, result)
  Gets the next aggregation state.
- subroutine nzaeGetStateBoolean(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateBoolean(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateDouble(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateDouble(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateFloat(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateFloat(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateInt8(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateInt8(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateInt16(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateInt16(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateInt32(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateInt32(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateString(handle, columnIndex, result, isNull)
  Gets the state value at the specified index.
- subroutine nzaeGetInputStateString(handle, columnIndex, result, isNull)
  Gets the input state value at the specified index.
- subroutine nzaeGetStateScale(handle, columnIndex, result)
  Gets the scale of the state at the specified index.
- subroutine nzaeGetStateSize(handle, columnIndex, result)
  Gets the size of the state at the specified index.
- subroutine nzaeGetStateType(handle, columnIndex, result)
  Gets the type of the state at the specified index.
- subroutine nzaeIsAggDone(handle, isDone)

Determines if the aggregation is done.

- subroutine nzaeIsAggError(handle, isError)
  Determines if the aggregation had an error.
- subroutine nzaeIsAggStateAccumulate(handle, isAccumulate)
  Determines if the aggregation state is "accumulate".
- subroutine nzaeIsAggStateInitializeState(handle, isInitializeState)
  Determines if the aggregation state is "initialize-state".
- subroutine nzaeIsAggStateAccumulate(handle, isFinalResult)
  Determines if the aggregation state is "final-result".
- subroutine nzaeIsAggStateMerge(handle, isMerge)
  Determines if the aggregation state is "merge".
- subroutine nzaeIsInputStateNull(handle, columnIndex, isNull)
  Determines if the input state value is NULL at the specified index.
- subroutine nzaeIsStateNull(handle, columnIndex, isNull)
  Determines if the state value is NULL at the specified index.
- subroutine nzaeSaveAggregateResult(handle)
  Saves the current aggregate result.
- subroutine nzaeSetAggregateBoolean(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateDouble(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateFloat(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateInt8(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateInt16(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateInt32(handle, columnIndex, value)
  Sets a state or output value for aggregation.
- subroutine nzaeSetAggregateNull(handle, columnIndex)
  Sets a NULL state or output value for aggregation.
- subroutine nzaeSetAggregateString(handle, columnIndex, value)
  Sets a state or output value for aggregation.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetNextAggregation(handle)**
  Gets the next aggregation state.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

- **subroutine nzaeGetNumberOfStateColumns(handle, result)**
  Gets the next aggregation state.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The number of columns in the state.

- **subroutine nzaeGetStateBoolean(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateBoolean(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateDouble(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (real*8) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateDouble(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (real*8) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateFloat(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (real) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateFloat(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (real) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateInt8(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateInt8(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateInt16(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateInt16(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateInt32(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateInt32(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateString(handle, columnIndex, result, isNull)**
  Gets the state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (character*) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputStateString(handle, columnIndex, result, isNull)**
  Gets the input state value at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (character*) The state.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeGetStateScale(handle, columnIndex, result)**
  Gets the scale of the state at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The scale of the state field.

- **subroutine nzaeGetStateSize(handle, columnIndex, result)**
  Gets the size of the state at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The size of the state field.

- **subroutine nzaeGetStateType(handle, columnIndex, result)**
  Gets the type of the state at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **result**
      (integer) The type of the state field.

- **subroutine nzaeIsAggDone(handle, isDone)**
  Determines if the aggregation is done.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isDone**
      (integer) The value is 1 if the aggregation is done; 0 otherwise.

- **subroutine nzaeIsAggError(handle, isError)**
  Determines if the aggregation had an error.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isError**
      (integer) The value is 1 if the aggregation had an error; 0 otherwise.

- **subroutine nzaeIsAggStateAccumulate(handle, isAccumulate)**
  Determines if the aggregation state is "accumulate".
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isAccumulate**
      (integer) The value is 1 if the aggregation state is "accumulate"; 0 otherwise.

- **subroutine nzaeIsAggStateInitializeState(handle, isInitializeState)**
  Determines if the aggregation state is "initialize-state".
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isInitializeState**
      (integer) The value is 1 if the aggregation state is "initialize-state"; 0 otherwise.

- **subroutine nzaeIsAggStateAccumulate(handle, isFinalResult)**
  Determines if the aggregation state is "final-result".
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isFinalResult**
      (integer) The value is 1 if the aggregation state is "final-result"; 0 otherwise.

- **subroutine nzaeIsAggStateMerge(handle, isMerge)**
  Determines if the aggregation state is "merge".
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isMerge**
      (integer) The value is 1 if the aggregation state is "merge"; 0 otherwise.

- **subroutine nzaeIsInputStateNull(handle, columnIndex, isNull)**
  Determines if the input state value is NULL at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **isNull**
      (integer) The value is 1 if the input state value is NULL; 0 otherwise.

  This function is only valid during the "merge" and "final-result" aggregation process state. Calls to this function at any other time sends a user error.

- **subroutine nzaeIsStateNull(handle, columnIndex, isNull)**
  Determines if the state value is NULL at the specified index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **isNull**
      (integer) The value is 1 if the state value is NULL; 0 otherwise.

- **subroutine nzaeSaveAggregateResult(handle)**
  Saves the current aggregate result.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

  After setting a state or an output result, before calling nzaeGetNextAggregation , this function must be called, otherwise the result is not be sent to the Netezza software.

- **subroutine nzaeSetAggregateBoolean(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The aggregate value to set.

- **subroutine nzaeSetAggregateDouble(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (real*8) The aggregate value to set.

- **subroutine nzaeSetAggregateFloat(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (real) The aggregate value to set.

- **subroutine nzaeSetAggregateInt8(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The aggregate value to set.

- **subroutine nzaeSetAggregateInt16(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The aggregate value to set.

- **subroutine nzaeSetAggregateInt32(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The aggregate value to set.

- **subroutine nzaeSetAggregateNull(handle, columnIndex)**
  Sets a NULL state or output value for aggregation.
  - Parameters
    - **columnIndex**
      (integer) The handle passed to nzaeHandleRequest .
    - **handle**
      (integer) The column index to set.

- **subroutine nzaeSetAggregateString(handle, columnIndex, value)**
  Sets a state or output value for aggregation.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (character*) The aggregate value to set.

Data Type Functions.

Functions/Subroutines

- subroutine nzaeIsABooleanDataType(type, result)
  Determines if the specified data type is a boolean type.
- subroutine nzaeIsADateDataType(type, result)
  Determines if the specified data type is a date type.
- subroutine nzaeIsADoubleDataType(type, result)
  Determines if the specified data type is a double type.

- subroutine nzaeIsAFixedStringDataType(type, result)
  Determines if the specified data type is a fixed string type.
- subroutine nzaeIsAFloatDataType(type, result)
  Determines if the specified data type is a float type.
- subroutine nzaeIsAnInt8DataType(type, result)
  Determines if the specified data type is an 8-bit integer type.
- subroutine nzaeIsAnInt16DataType(type, result)
  Determines if the specified data type is a 16-bit integer type.
- subroutine nzaeIsAnInt32DataType(type, result)
  Determines if the specified data type is a 32-bit integer type.
- subroutine nzaeIsAnInt64DataType(type, result)
  Determines if the specified data type is a 64-bit integer type.
- subroutine nzaeIsAnIntervalDataType(type, result)
  Determines if the specified data type is an interval type.
- subroutine nzaeIsANationalFixedStringDataType(type, result)
  Determines if the specified data type is a national fixed string type.
- subroutine nzaeIsANationalVariableStringDataType(type, result)
  Determines if the specified data type is a national variable string type.
- subroutine nzaeIsAGeometryDataType(type, result)
  Determines if the specified data type is a geometry string type.
- subroutine nzaeIsAVarbinaryDataType(type, result)
  Determines if the specified data type is a varbinary string type.
- subroutine nzaeIsANumericDataType(type, result)
  Determines if the specified data type is a numeric type.
- subroutine nzaeIsANumeric32DataType(type, result)
  Determines if the specified data type is a numeric-32 type.
- subroutine nzaeIsANumeric64DataType(type, result)
  Determines if the specified data type is a numeric-64 type.
- subroutine nzaeIsANumeric128DataType(type, result)
  Determines if the specified data type is a numeric-128 type.
- subroutine nzaeIsAStringDataType(type, result)
  Determines if the specified data type is a string type.
- subroutine nzaeIsATimeDataType(type, result)
  Determines if the specified data type is a time type.
- subroutine nzaeIsATimeStampDataType(type, result)
  Determines if the specified data type is a time stamp type.
- subroutine nzaeIsATimeZoneDataType(type, result)
  Determines if the specified data type is a time zone type.
- subroutine nzaeIsAVariableStringDataType(type, result)
  Determines if the specified data type is a variable string type.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeIsABooleanDataType(type, result)**
  Determines if the specified data type is a boolean type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a boolean type; 0 otherwise.

- **subroutine nzaeIsADateDataType(type, result)**
  Determines if the specified data type is a date type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a date type; 0 otherwise.

- **subroutine nzaeIsADoubleDataType(type, result)**
  Determines if the specified data type is a double type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a double type; 0 otherwise.

- **subroutine nzaeIsAFixedStringDataType(type, result)**
  Determines if the specified data type is a fixed string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a fixed string data type; 0 otherwise.

- **subroutine nzaeIsAFloatDataType(type, result)**
  Determines if the specified data type is a float type.

- Parameters
  - **type**
    (integer) The data type of the column.
  - **result**
    (integer) The value is 1 if the data type is a float type; 0 otherwise.

- **subroutine nzaeIsAnInt8DataType(type, result)**
  Determines if the specified data type is an 8-bit integer type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is an 8-bit integer; 0 otherwise.

- **subroutine nzaeIsAnInt16DataType(type, result)**
  Determines if the specified data type is a 16-bit integer type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a 16-bit integer; 0 otherwise.

- **subroutine nzaeIsAnInt32DataType(type, result)**
  Determines if the specified data type is a 32-bit integer type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a 32-bit integer; 0 otherwise.

- **subroutine nzaeIsAnInt64DataType(type, result)**
  Determines if the specified data type is a 64-bit integer type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a 64-bit integer; 0 otherwise.

- **subroutine nzaeIsAnIntervalDataType(type, result)**
  Determines if the specified data type is an interval type.

- Parameters
  - **type**
    (integer) The data type of the column.
  - **result**
    (integer) The value is 1 if the data type is an interval type; 0 otherwise.

- **subroutine nzaeIsANationalFixedStringDataType(type, result)**
  Determines if the specified data type is a national fixed string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a national fixed string data type; 0 otherwise.

- **subroutine nzaeIsANationalVariableStringDataType(type, result)**
  Determines if the specified data type is a national variable string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a national variable string data type; 0 otherwise.

- **subroutine nzaeIsAGeometryDataType(type, result)**
  Determines if the specified data type is a geometry string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a geometry string data type; 0 otherwise.

- **subroutine nzaeIsAVarbinaryDataType(type, result)**
  Determines if the specified data type is a varbinary string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a varbinary string data type; 0 otherwise.

- **subroutine nzaeIsANumericDataType(type, result)**
  Determines if the specified data type is a numeric type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a numeric data type; 0 otherwise.

- **subroutine nzaeIsANumeric32DataType(type, result)**
  Determines if the specified data type is a numeric-32 type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a numeric-32 type; 0 otherwise.

- **subroutine nzaeIsANumeric64DataType(type, result)**
  Determines if the specified data type is a numeric-64 type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a numeri64 type; 0 otherwise.

- **subroutine nzaeIsANumeric128DataType(type, result)**
  Determines if the specified data type is a numeric-128 type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a numeric-128 type; 0 otherwise.

- **subroutine nzaeIsAStringDataType(type, result)**
  Determines if the specified data type is a string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a string type; 0 otherwise.

- **subroutine nzaeIsATimeDataType(type, result)**
  Determines if the specified data type is a time type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a time type; 0 otherwise.

- **subroutine nzaeIsATimeStampDataType(type, result)**
  Determines if the specified data type is a time stamp type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a time stamp type; 0 otherwise.

- **subroutine nzaeIsATimeZoneDataType(type, result)**
  Determines if the specified data type is a time zone type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a time zone type; 0 otherwise.

- **subroutine nzaeIsAVariableStringDataType(type, result)**
  Determines if the specified data type is a variable string type.
  - Parameters
    - **type**
      (integer) The data type of the column.
    - **result**
      (integer) The value is 1 if the data type is a variable string type; 0 otherwise.

Environment Functions.

Functions/Subroutines

- subroutine nzaeGetEnvironmentVariable(handle, name, value)
  Returns the environment variable value associated with the passed name.
- subroutine nzaeGetFirstEnvironmentVariable(handle, name, value)

Returns the first name/value pair in the environment.

- subroutine nzaeGetNextEnvironmentVariable(handle, name, value, hasValue)
  Returns the next name/value pair in the environment.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetEnvironmentVariable(handle, name, value)**
  Returns the environment variable value associated with the passed name.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **name**
      (character*) The name to look up in the environment.
    - **value**
      (integer) The value of name in the environment.

  The result is an empty string if the environment value is not set. The name parameter is case insensitive.

- **subroutine nzaeGetFirstEnvironmentVariable(handle, name, value)**
  Returns the first name/value pair in the environment.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **name**
      (character*) The name of the first entry in the environment.
    - **value**
      (integer) The value of the first entry in the environment.

  This function should be used in conjunction with nzaeGetNextEnvironmentVariable to determine the entire environment.

- **subroutine nzaeGetNextEnvironmentVariable(handle, name, value, hasValue)**
  Returns the next name/value pair in the environment.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **name**
      (character*) The name of the first entry in the environment.
    - **value**
      (integer) The value of the first entry in the environment.

- **hasValue**
  (integer) The value is 1 if name and value were set. The value is 0 if there are no more entries in the environment.

This function should be used in conjunction with nzaeGetFirstEnvironmentVariable in order to determine the entire environment.

General AE Functions.

Functions/Subroutines

- subroutine nzaeClose(handle)
  Optionally called when no more interface functions are to be called to the Netezza system.
- subroutine nzaeDone(handle)
  Optionally called when there is no data left to be sent.
- subroutine nzaeIsLocal(isLocal)
  Specifies whether the AE is local or remote.
- subroutine nzaeIsRemote(isRemote)
  Specifies whether the AE is local or remote.
- subroutine nzaeIsShaper(handle, isShaper)
  Specifies whether the AE currently running is a Shaper.
- subroutine nzaeIsUda(handle, isUda)
  Specifies whether the AE currently running is a UDA.
- subroutine nzaeIsUdf(handle, isUdf)
  Specifies whether the AE currently running is a UDF.
- subroutine nzaeIsUdtf(handle, isUdtf)
  Specifies whether the AE currently running is a UDTF.
- subroutine nzaePing(handle)
  Notifies the Netezza system that work is still being performed and the AE should not be terminated.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeClose(handle)**
  Optionally called when no more interface functions are to be called to the Netezza system.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

  This functionality is performed upon returning from nzaeHandleRequest , if it has not already

been called. It frees up adapter memory in the Netezza system for AEs that are doing work after finishing other AE API calls.

- **subroutine nzaeDone(handle)**
  Optionally called when there is no data left to be sent.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

  This functionality is performed upon returning from nzaeHandleRequest , if it has not already been called.

- **subroutine nzaeIsLocal(isLocal)**
  Specifies whether the AE is local or remote.
  - Parameters
    - **isLocal**
      (integer) The return value of this function. A value of 1 indicates the AE is local; 0 indicates remote.

- **subroutine nzaeIsRemote(isRemote)**
  Specifies whether the AE is local or remote.
  - Parameters
    - **isRemote**
      (integer) The return value of this function. A value of 1 indicates the AE is remote; 0 indicates local.

- **subroutine nzaeIsShaper(handle, isShaper)**
  Specifies whether the AE currently running is a Shaper.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isShaper**
      (integer) The return value of this function. A value of 1 indicates the currently-running AE is a shaper; 0 otherwise.

- **subroutine nzaeIsUda(handle, isUda)**
  Specifies whether the AE currently running is a UDA.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUda**
      (integer) The return value of this function. A value of 1 indicates the currently-running AE is a

UDA; 0 otherwise.

- **subroutine nzaeIsUdf(handle, isUdf)**
  Specifies whether the AE currently running is a UDF.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUdf**
      (integer) The return value of this function. A value of 1 indicates the currently-running AE is a UDF; 0 otherwise.

- **subroutine nzaeIsUdtf(handle, isUdtf)**
  Specifies whether the AE currently running is a UDTF.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUdtf**
      (integer) The return value of this function. A value of 1 indicates the currently-running AE is a UDTF; 0 otherwise.

- **subroutine nzaePing(handle)**
  Notifies the Netezza system that work is still being performed and the AE should not be terminated.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

  When there are long periods of inactivity with the AE system, this function should be called to notify the Netezza system that the AE is still working.

Logging Functions.

Functions/Subroutines

- subroutine nzaeGetLogFilePath(handle, path)
  Returns the full path to the AE log file.
- subroutine nzaeLog(handle, string)
  Logs the specified string to the AE log file.
- subroutine nzaeLogStderr(handle, string)
  Writes the specified string to stderr.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetLogFilePath(handle, path)**
  Returns the full path to the AE log file.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **path**
      (character*) The full path to the log file.

- **subroutine nzaeLog(handle, string)**
  Logs the specified string to the AE log file.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **string**
      (character*) The string to log.

- **subroutine nzaeLogStderr(handle, string)**
  Writes the specified string to stderr.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **string**
      (character*) The string to write to stderr.

  When the AE is registered with a particular log level, this function can be useful to help with debugging as the stdout and stderr of the AE is logged by the AE system to a specific location on the file system.

Primary Interface Functions.

Functions/Subroutines

- subroutine nzaeRun()
  This function is called by the AE when the executable is ready to turn control over to the Netezza system and the AE subsystem.
- subroutine nzaeHandleRequest(handle)
  This function is called by the AE subsystem when it is ready to let the AE handle a request from the Netezza system.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeRun()**
  This function is called by the AE when the executable is ready to turn control over to the Netezza system and the AE subsystem.

  The AE subsystem makes calls to nzaeHandleRequest as appropriate when requests come in.

- **subroutine nzaeHandleRequest(handle)**
  This function is called by the AE subsystem when it is ready to let the AE handle a request from the Netezza system.

  - Parameters
    - **handle**
      (integer) The handle for this run of the AE.

Remote AE Functions.

Functions/Subroutines

- subroutine nzaeDisableForking()
  Disables forking in the remote AE.
- subroutine nzaeEnableForking()
  Enables forking in the remote AE.
- subroutine nzaeGetConnectionPointName(name)
  Gets the remote AE connection point name.
- subroutine nzaeGetConnectionPointDatasliceId(id)
  Gets the connection point dataslice ID.
- subroutine nzaeGetConnectionPointSessionId(id)
  Gets the connection point session ID.
- subroutine nzaeGetConnectionPointTransactionId(idString)
  Gets the connection point transaction ID.
- subroutine nzaeIsForkingEnabled(isEnabled)
  Determines if forking is enabled in the remote AE.
- subroutine nzaeIsRemoteProtocolCodeControlData(code, result)
  Determines if a remote protocol code is a CONTROL_DATA code.
- subroutine nzaeIsRemoteProtocolCodePing(code, result)
  Determines if a remote protocol code is a PING code.
- subroutine nzaeIsRemoteProtocolCodeRequest(code, result)

Determines if a remote protocol code is a REQUEST code.

- subroutine nzaeIsRemoteProtocolCodeStatus(code, result)
  Determines if a remote protocol code is a STATUS code.
- subroutine nzaeIsRemoteProtocolCodeStop(code, result)
  Determines if a remote protocol code is a STOP code.
- subroutine nzaeSetConnectionPointName(name)
  Sets the remote AE connection point name.
- subroutine nzaeSetConnectionPointDatasliceId(id)
  Sets the remote dataslice ID.
- subroutine nzaeSetConnectionPointSessionId(id)
  Sets the remote session ID.
- subroutine nzaeSetConnectionPointTransactionId(idString, result)
  Sets the connection point transaction ID.
- subroutine nzaeSetRemoteProtocolCallback(callback)
  Sets the subroutine that will be called for handling remote protocol messages.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeDisableForking()**
  Disables forking in the remote AE.

  Forking is enabled by default in remote AEs. This routine disables it.

- **subroutine nzaeEnableForking()**
  Enables forking in the remote AE.

  Forking is enabled by default in remote AEs. If it was previously disabled, this function re-enables forking for the AE.

- **subroutine nzaeGetConnectionPointName(name)**
  Gets the remote AE connection point name.
  - Parameters
    - **name**
      (character*) The remote connection point name.

- **subroutine nzaeGetConnectionPointDatasliceId(id)**
  Gets the connection point dataslice ID.
  - Parameters
    - **id**
      (integer) The remote dataslice ID.

This routine is generally not needed if using the standard remote launch mechanism since the AE gets the ID from the environment set up by the remote launch mechanism.

- **subroutine nzaeGetConnectionPointSessionId(id)**
  Gets the connection point session ID.
  - Parameters
    - **id**
      (integer) The remote session ID.

  This routine is generally not needed if using the standard remote launch mechanism as the AE gets the ID from the environment set up by the remote launch mechanism.

- **subroutine nzaeGetConnectionPointTransactionId(idString)**
  Gets the connection point transaction ID.
  - Parameters
    - **idString**
      (character*) A string representing the numeric connection point transaction ID. The string must be at least 21 characters long or else the result string is filled with the NULL character (''').

- **subroutine nzaeIsForkingEnabled(isEnabled)**
  Determines if forking is enabled in the remote AE.
  - Parameters
    - **isEnabled**
      (integer) The value is 1 if forking is enabled, which is the default; 0 if not enabled.

- **subroutine nzaeIsRemoteProtocolCodeControlData(code, result)**
  Determines if a remote protocol code is a CONTROL_DATA code.
  - Parameters
    - **code**
      (integer) The remote protocol code passed in to the remote protocol callback.
    - **result**
      (integer) The value is 1 if the code is a CONTROL_DATA code; 0 otherwise.

- **subroutine nzaeIsRemoteProtocolCodePing(code, result)**
  Determines if a remote protocol code is a PING code.
  - Parameters
    - **code**
      (integer) The remote protocol code passed in to the remote protocol callback.
    - **result**

(integer) The value is 1 if the code is a PING code; 0 otherwise.

- **subroutine nzaeIsRemoteProtocolCodeRequest(code, result)**
  Determines if a remote protocol code is a REQUEST code.
  - Parameters
    - **code**
      (integer) The remote protocol code passed in to the remote protocol callback.
    - **result**
      (integer) The value is 1 if the code is a REQUEST code; 0 otherwise.

- **subroutine nzaeIsRemoteProtocolCodeStatus(code, result)**
  Determines if a remote protocol code is a STATUS code.
  - Parameters
    - **code**
      (integer) The remote protocol code passed in to the remote protocol callback.
    - **result**
      (integer) The value is 1 if the code is a STATUS code; 0 otherwise.

- **subroutine nzaeIsRemoteProtocolCodeStop(code, result)**
  Determines if a remote protocol code is a STOP code.
  - Parameters
    - **code**
      (integer) The remote protocol code passed in to the remote protocol callback.
    - **result**
      (integer) The value is 1 if the code is a STOP code; 0 otherwise.

- **subroutine nzaeSetConnectionPointName(name)**
  Sets the remote AE connection point name.
  - Parameters
    - **name**
      (character*) The remote connection point name.

- **subroutine nzaeSetConnectionPointDatasliceId(id)**
  Sets the remote dataslice ID.
  - Parameters
    - **id**
      (integer) The remote dataslice ID.

  This routine is generally not needed if using the standard remote launch mechanism since the AE gets the ID from the environment set up by the remote launch mechanism.

- **subroutine nzaeSetConnectionPointSessionId(id)**
  Sets the remote session ID.
  - Parameters
    - **id**
      (integer) The remote session ID.

  This routine is generally not needed if using the standard remote launch mechanism since the AE gets the ID from the environment set up by the remote launch mechanism.

- **subroutine nzaeSetConnectionPointTransactionId(idString, result)**
  Sets the connection point transaction ID.
  - Parameters
    - **idString**
      (character*) A string representing the numeric remote transaction ID.
    - **result**
      (integer) The result is 1 if the function succeeded; 0 otherwise.

  This routine is generally not needed if using the standard remote launch mechanism since the AE gets the ID from the environment set up by the remote launch mechanism. A string is required because the Fortran 77 specification does not handle 64 bit integers.

- **subroutine nzaeSetRemoteProtocolCallback(callback)**
  Sets the subroutine that will be called for handling remote protocol messages.
  - Parameters
    - **callback**
      (subroutine(integer, character*)) The subroutine to be called.

  The subroutine that is passed in receives an integer code that can be identified with the nzaeIsRemoteProtocolCodeXXX() functions. The string that is passed in may contain data from the AE subsystem. Both parameters are also used as output to the AE subsystem. The length of the string that is passed in must be at least 1000 bytes.

Row Fetching Functions.

Functions/Subroutines

- subroutine nzaeGetInputBoolean(handle, columnIndex, value, isNull)
  Returns the value of the current row at the specified column index.
- subroutine nzaeGetInputDouble(handle, columnIndex, value, isNull)
  Returns the value of the current row at a specified column index.
- subroutine nzaeGetInputFloat(handle, columnIndex, value, isNull)
  Returns the value of the current row at a specified column index.
- subroutine nzaeGetInputInt8(handle, columnIndex, value, isNull)

Returns the value of the current row at a specified column index.

- subroutine nzaeGetInputInt16(handle, columnIndex, value, isNull)
  Returns the value of the current row at a specified column index.
- subroutine nzaeGetInputInt32(handle, columnIndex, value, isNull)
  Returns the value of the current row at a specified column index.
- subroutine nzaeGetInputString(handle, columnIndex, value, isNull)
  Returns the value of the current row at a specified column index.
- subroutine nzaeGetNext(handle)
  Gets the next input row.
- subroutine nzaeIsInputNull(handle, columnIndex, isNull)
  Determines if the value of the current row at the specified column index is NULL.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetInputBoolean(handle, columnIndex, value, isNull)**
  Returns the value of the current row at the specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (integer) The value is 1 if the value is TRUE; 0 if the value is FALSE.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputDouble(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (real*8) The value of the row at the specified index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputFloat(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (real) The value of the row at the specified index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputInt8(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (integer) The value of the row at the specified index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputInt16(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (integer) The value of the row at the specified index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputInt32(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.

- Parameters
  - **handle**
    (integer) The handle passed to nzaeHandleRequest .
  - **columnIndex**
    (integer) The column index.
  - **value**
    (integer) The value of the row at the specified index.
  - **isNull**
    (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetInputString(handle, columnIndex, value, isNull)**
  Returns the value of the current row at a specified column index.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **value**
      (character*) The value of the row at the specified index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

- **subroutine nzaeGetNext(handle)**
  Gets the next input row.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .

- **subroutine nzaeIsInputNull(handle, columnIndex, isNull)**
  Determines if the value of the current row at the specified column index is NULL.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index.
    - **isNull**
      (integer) The value is 1 if the value is NULL; 0 otherwise.

Row Outputting Functions.

Functions/Subroutines

- subroutine nzaeOutputInputColumn(handle, inputColumnIndex, outputColumnIndex)
  Copies an input value to the output row.
- subroutine nzaeSetOutputBoolean(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputDouble(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputFloat(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputInt8(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputInt16(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputInt32(handle, columnIndex, value)
  Sets the value of an output.
- subroutine nzaeSetOutputNull(handle, columnIndex)
  Sets the value of an output to NULL.
- subroutine nzaeSetOutputString(handle, columnIndex, value)
  Sets the value of an output.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeOutputInputColumn(handle, inputColumnIndex, outputColumnIndex)**
  Copies an input value to the output row.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **inputColumnIndex**
      (integer) The input column index to copy.
    - **outputColumnIndex**
      (integer) The output column index to set.

- **subroutine nzaeSetOutputBoolean(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**

(integer) The handle passed to nzaeHandleRequest .

- **columnIndex**
  (integer) The column index to set.
- **value**
  (integer) The value to set.

- **subroutine nzaeSetOutputDouble(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (real*8) The value to set.

- **subroutine nzaeSetOutputFloat(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (real) The value to set.

- **subroutine nzaeSetOutputInt8(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The value to set.

- **subroutine nzaeSetOutputInt16(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**

(integer) The handle passed to nzaeHandleRequest .

- **columnIndex**
  (integer) The column index to set.
- **value**
  (integer) The value to set.

- **subroutine nzaeSetOutputInt32(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (integer) The value to set.

- **subroutine nzaeSetOutputNull(handle, columnIndex)**
  Sets the value of an output to NULL.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.

- **subroutine nzaeSetOutputString(handle, columnIndex, value)**
  Sets the value of an output.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **columnIndex**
      (integer) The column index to set.
    - **value**
      (character*) The value to set.

Run-Time Functions.

Functions/Subroutines

- subroutine nzaeGetUsername(handle, result)

Returns the database username for the current query the AE is running.

- subroutine nzaeGetDatasliceId(handle, result)
  Returns the dataslice ID on which the current AE is running.
- subroutine nzaeGetHardwareId(handle, result)
  Returns the hardware ID on which the current AE is running.
- subroutine nzaeGetNumberOfDataslices(handle, result)
  Returns the number of dataslices on which the NPS is running.
- subroutine nzaeGetNumberOfSpus(handle, result)
  Returns the number of SPUs on which the NPS is running.
- subroutine nzaeGetSessionId(handle, result)
  Returns the session ID on which the current AE is running.
- subroutine nzaeGetSuggestedMemoryLimit(handle, result)
  Returns the suggested memory limit for the AE in bytes.
- subroutine nzaeGetTransactionId(handle, result)
  Returns a string representation of the current transaction ID.
- subroutine nzaeIsRunningInPostgres(handle, result)
  Specifies whether the current AE is being run from Postgres.
- subroutine nzaeIsRunningInDbos(handle, result)
  Specifies whether the current AE is being run from DBOS.
- subroutine nzaeIsRunningOnSpu(handle, result)
  Specifies whether the current AE is being run on a SPU.
- subroutine nzaeIsUserQuery(handle, result)
  Specifies whether the current AE is being run on a SPU. Returns 1 if the current AE is being run as a user query and 0 otherwise.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeGetUsername(handle, result)**
  Returns the database username for the current query the AE is running.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (character*) The current username. An error results if the passed string is not long enough.

- **subroutine nzaeGetDatasliceId(handle, result)**
  Returns the dataslice ID on which the current AE is running.
  - Parameters
    - **handle**

(integer) The handle passed to nzaeHandleRequest .

- **result**
  (integer) The return value of this function.

- **subroutine nzaeGetHardwareId(handle, result)**
  Returns the hardware ID on which the current AE is running.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (character*) The return value of this function. If the passed string is not at least 21 characters long, the result is filled with the NULL character (''').

- **subroutine nzaeGetNumberOfDataslices(handle, result)**
  Returns the number of dataslices on which the NPS is running.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function.

- **subroutine nzaeGetNumberOfSpus(handle, result)**
  Returns the number of SPUs on which the NPS is running.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function.

- **subroutine nzaeGetSessionId(handle, result)**
  Returns the session ID on which the current AE is running.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function.

- **subroutine nzaeGetSuggestedMemoryLimit(handle, result)**

Returns the suggested memory limit for the AE in bytes.

- Parameters
  - **handle**
    (integer) The handle passed to nzaeHandleRequest .
  - **result**
    (real*8) The return value of this function.

- **subroutine nzaeGetTransactionId(handle, result)**
  Returns a string representation of the current transaction ID.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (character*) The string representation of the current transaction ID.

- **subroutine nzaeIsRunningInPostgres(handle, result)**
  Specifies whether the current AE is being run from Postgres.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function. Returns 1 if the current AE is being run from Postgres; 0 otherwise.

- **subroutine nzaeIsRunningInDbos(handle, result)**
  Specifies whether the current AE is being run from DBOS.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function. Returns 1 if the current AE is being run from DBOS; 0 otherwise.

- **subroutine nzaeIsRunningOnSpu(handle, result)**
  Specifies whether the current AE is being run on a SPU.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function. Returns 1 if the current AE is being run on a SPU; 0 otherwise.

- **subroutine nzaeIsUserQuery(handle, result)**
  Specifies whether the current AE is being run on a SPU. Returns 1 if the current AE is being run as a user query and 0 otherwise.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **result**
      (integer) The return value of this function. Returns 1 if the current AE is being run as a user query; 0 otherwise.

Shaper and Sizer Functions.

Functions/Subroutines

- subroutine nzaeAddOutputColumnString(handle, dataType, columnName, size)
  Adds a string output column.
- subroutine nzaeAddOutputColumnNumeric(handle, dataType, columnName, precision, scale)
  Adds a numeric output column.
- subroutine nzaAddOutputColumn(handle, dataType, columnName)
  Adds a non-string, non-numeric output column.
- subroutine nzaeIsInputConstant(handle, columnIndex, isConstant)
  Determines if the input to the shaper/sizer function is a constant, and therefore available.
- subroutine nzaeIsSystemCatalogUpperCase(handle, isUpperCase)
  Determines if the the NPS catalog is in upper case.
- subroutine nzaeIsUdfSizer(handle, isUdfSizer)
  Determines if the AE that is running is a UDF sizer.
- subroutine nzaeIsUdtfShaper(handle, isUdtfShaper)
  Determines if the AE that is running is a UDTF shaper.

Detailed Description

Function/Subroutine Documentation

- **subroutine nzaeAddOutputColumnString(handle, dataType, columnName, size)**
  Adds a string output column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **dataType**

(integer) The UDS string type.

- **columnName**
  (character*) The column name to add. Ignored for sizer AEs.
- **size**
  (integer) The size/width of the string column to add.

The possible UDS string types are Fixed/Char (0), Variable/Varchar (1), National-Fixed (2), and National-Variable (3).

- **subroutine nzaeAddOutputColumnNumeric(handle, dataType, columnName, precision, scale)**
  Adds a numeric output column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **dataType**
      (integer) The UDS numeric type.
    - **columnName**
      (character*) The column name to add. Ignored for sizer AEs.
    - **precision**
      (integer) The precision of the numeric column to add.
    - **scale**
      (integer) The scale of the numeric column to add.

  The possible UDS numeric types are numeric32 (8), numeric64 (9), and numeric128 (10).

- **subroutine nzaAddOutputColumn(handle, dataType, columnName)**
  Adds a non-string, non-numeric output column.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **dataType**
      (integer) The UDS numeric type.
    - **columnName**
      (character*) The column name to add. Ignored for sizer AEs.
    - **precision**
      (integer) The precision of the numeric column to add.
    - **scale**
      (integer) The scale of the numeric column to add.

  The possible UDS types are bool (4), date (5), time (6), timezone (7), float (11), double (12), interval (13), 8-bit integer (14), 16-bit integer (15), 32-bit integer (16), 64-bit integer (17), and timestamp (18).

- **subroutine nzaeIsInputConstant(handle, columnIndex, isConstant)**

Determines if the input to the shaper/sizer function is a constant, and therefore available.

- Parameters
  - **handle**
    (integer) The handle passed to nzaeHandleRequest .
  - **columnIndex**
    (integer) The column index.
  - **isConstant**
    (integer) The value is 1 if the input is a constant; 0 otherwise.

- **subroutine nzaeIsSystemCatalogUpperCase(handle, isUpperCase)**
  Determines if the the NPS catalog is in upper case.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUpperCase**
      (integer) The value is 1 if the system catalog is upper-case; 0 otherwise.

- **subroutine nzaeIsUdfSizer(handle, isUdfSizer)**
  Determines if the AE that is running is a UDF sizer.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUdfSizer(integer)**
      The value is 1 if the AE is a UDF sizer; 0 otherwise.

- **subroutine nzaeIsUdtfShaper(handle, isUdtfShaper)**
  Determines if the AE that is running is a UDTF shaper.
  - Parameters
    - **handle**
      (integer) The handle passed to nzaeHandleRequest .
    - **isUdtfShaper**
      (integer) The value is 1 if the AE is a UDTF shaper; 0 otherwise.

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 trademark 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

D

E

G

L

M

N

P

R

S