

IBM® Netezza® Analytics
Release 11.x

*Analytic Executables Language
Development Kit API Reference*



Note: Before using this information and the product that it supports, read the information in "[Notices and Trademarks](#)" on page 99.

Contents

Preface

Audience for This Guide	xi
Purpose of This Guide	xi
Conventions.....	xi
If You Need Help	xi
Comments on the Documentation	xii

1 Module Documentation

Initialization APIs	13
Data Structures.....	13
Modules	13
Enumerations	14
Detailed Description	14
Enumeration Type Documentation	14
Local Initialization.....	14
Initialize from an AE Environment.....	15
Remote Connection Point	17
Remote Initialization.....	21
Data Connection APIs.....	29
Modules	29
Detailed Description	29
Function	29
Aggregate.....	37
Shaper and Sizer	45
Support APIs	55
Modules	55
Detailed Description	55
AE Manager Functionality	55
Date and Time Functions	55
Numeric Functions.....	60
Runtime and Environment Information.....	61
User Codes	63
Data Type Support	64

Data Structures.....	64
Typedefs.....	64
Enumerations	64
Detailed Description	65
Typedef Documentation	65
Enumeration Type Documentation	65

Data Structure Documentation

NZAE_HANDLE Struct Reference	67
Detailed Description	67
NZAEAGG_HANDLE Struct Reference	67
Detailed Description	67
NzaeAggAccumulate Struct Reference	67
Public Attributes.....	67
Detailed Description	68
Member Data Documentation	68
NzaeAggFieldFunctions Struct Reference	68
Public Member Functions	68
Public Attributes.....	69
Detailed Description	69
Public Member Function Documentation.....	69
Member Data Documentation	77
NzaeAggFinalResult Struct Reference.....	77
Public Attributes.....	77
Detailed Description	77
Member Data Documentation	77
NzaeAggInitialization Struct Reference.....	77
Public Attributes.....	77
Detailed Description	77
Member Data Documentation	78
NzaeAggInitializeState Struct Reference.....	78
Public Attributes.....	78
Detailed Description	78
Member Data Documentation	78
NzaeAggMerge Struct Reference.....	78
Public Attributes.....	78
Detailed Description	78

Member Data Documentation	79
NzaeAggMetadata Struct Reference.....	79
Public Attributes	79
Detailed Description	79
Member Data Documentation	79
NzaeAggReadOnlyFieldFunctions Struct Reference	79
Public Member Functions	79
Public Attributes	80
Detailed Description	80
Public Member Function Documentation.....	80
Member Data Documentation	81
NzaeApi Struct Reference	81
Public Attributes.....	81
Detailed Description	81
Member Data Documentation	81
NZAECONPT_HANDLE Struct Reference	82
Detailed Description	82
NZAEENV_HANDLE Struct Reference.....	82
NzaeEnvironmentEntry Struct Reference	82
Public Attributes.....	82
Detailed Description	82
Member Data Documentation	83
NzaeInitialization Struct Reference	83
Public Attributes.....	83
Detailed Description	83
Member Data Documentation	83
NzaeMetadata Struct Reference	83
Public Attributes.....	83
Detailed Description	84
Member Data Documentation	84
NzaeNumeric128BytesBigEndian Struct Reference.....	85
Public Attributes.....	85
Member Data Documentation	85
NzaeNumeric128BytesLittleEndian Struct Reference.....	85
Public Attributes.....	86
Member Data Documentation	86
NzaeNumeric32BytesBigEndian Struct Reference.....	86

Public Attributes	86
Member Data Documentation	86
NzAENumeric32BytesLittleEndian Struct Reference.....	86
Public Attributes	86
Member Data Documentation	86
NzAENumeric64BytesBigEndian Struct Reference.....	86
Public Attributes	86
Member Data Documentation	87
NzAENumeric64BytesLittleEndian Struct Reference.....	87
Public Attributes	87
Member Data Documentation	87
NZAEREMPROT_HANDLE Struct Reference.....	87
Detailed Description	87
NzAERemprotCallbackResult Struct Reference	87
Public Attributes	87
Detailed Description	87
Member Data Documentation	88
NzAERemprotInitialization Struct Reference.....	88
Public Attributes	88
Detailed Description	88
Member Data Documentation	88
NzAERuntime Struct Reference	89
Public Attributes	89
Detailed Description	89
Member Data Documentation	89
NzAERSharedLibraryInfo Struct Reference.....	90
Public Attributes	90
Detailed Description	90
Member Data Documentation	91
NZAESHV_HANDLE Struct Reference	91
Detailed Description	91
NzAERShpInitialization Struct Reference.....	91
Public Attributes	91
Detailed Description	91
Member Data Documentation	92
NzAERShpMetadata Struct Reference.....	92
Public Attributes	92

Detailed Description	92
Member Data Documentation	93
NzaeShpOutputColumnInfo Struct Reference.....	93
Public Attributes.....	93
Detailed Description	93
Member Data Documentation	93
NzudsData Struct Reference	94
Public Attributes.....	94
Detailed Description	95
Member Data Documentation	95
NzudsInterval Struct Reference	98
Public Attributes.....	98
Detailed Description	98
Member Data Documentation	98
NzudsNumeric128 Struct Reference.....	98
Public Attributes.....	98
Detailed Description	98
Member Data Documentation	99
NzudsNumeric32 Struct Reference.....	99
Public Attributes.....	99
Detailed Description	99
Member Data Documentation	99
NzudsNumeric64 Struct Reference.....	99
Public Attributes.....	99
Detailed Description	99
Member Data Documentation	99
NzudsTimeTz Struct Reference.....	99
Public Attributes.....	100
Detailed Description	100
Member Data Documentation	100

Notices and Trademarks

Notices	101
Trademarks.....	102
Regulatory and Compliance	103
Regulatory Notices	103
Homologation Statement	103

FCC - Industry Canada Statement.....	103
CE Statement (Europe)	103
VCCI Statement	103

Index

Preface

The LDK provides the base AE client interface on which all other AE adapters are built.

Audience for This Guide

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

Purpose of This Guide

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

Conventions

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

The following conventions apply:

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

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

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

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

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

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

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

If You Need Help

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

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

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

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

Comments on the Documentation

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

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

CHAPTER 1

Module Documentation

Initialization APIs

This API family is used to get an open data connection or to get an AE Environment that can be used to open a data connection.

Data Structures

struct NzaeApi
Contains a data connection handle.

Modules

Local Initialization

Initialization functions related to Local AEs. Local AEs are initialized using the function `nzaeLoc-protGetApi`. If an AE is local, function `nzaelsLocal` returns a TRUE value. If an AE is not local it is remote.

Initialize from an AE Environment.

Used to get a data connection from an AE Environment.

Remote Connection Point.

A Remote Connection Point is how the Netezza system addresses a Remote AE.

Remote Initialization.

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

- Create a connection point.

- Listen using that connection point.

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

Enumerations

```
enum NzaeApiTypes {  
    NZAE_API_UNKNOWN= 0, NZAE_API_FUNCTION= 1, NZAE_API_AGGREGATION=  
    2, NZAE_API_SHAPER= 3 }
```

The Data Connection API Type.

Detailed Description

This API family is used to get an open data connection or to get an AE Environment that can be used to open a data connection.

Enumeration Type Documentation

```
enum NzaeApiTypes  
The Data Connection API Type.
```

NZAE_API_UNKNOWN

NZAE_API_FUNCTION

NZAE_API_AGGREGATION

NZAE_API_SHAPER

See Also

NzaeApi

nzaeRemprotGetEnvironmentApiType

Local Initialization

Initialization functions related to Local AEs. Local AEs are initialized using the function `nzaeLocprotGetApi`. If an AE is local, function `nzaelsLocal` returns a TRUE value. If an AE is not local it is re-mote.

Functions

```
int nzaelsLocal()
```

Returns TRUE if the AE is local.

```
int nzaelsRemote()
```

Returns a true value if this is a Remote AE.

```
int nzaeLocprotGetApi(NzaeApi *result, int ldkVersion, char *errorMessage, int  
errorMes-sageSize)
```

Returns the handle for a local AE.

Detailed Description

Initialization functions related to Local AEs. Local AEs are initialized using the function `nzaeLocprotGetApi`. If an AE is local, function `nzaelsLocal` returns a TRUE value. If an AE is not local it is re-mote.

Function Documentation

int nzaelsLocal()

Returns TRUE if the AE is local.

Returns
TRUE if the AE is local.

The lifecycle of a local process is controlled by the Netezza software.

int nzaelsRemote()

Returns a true value if this is a Remote AE.

Returns
True if this a local AE

int nzaelocprotGetApi(NzaeApi *result, int ldkVersion, char *errorMessage, int errorMessageSize) Returns the handle for a local AE.

Parameters

NzaeApi result

The returned API.

ldkVersion

The expected version.

errorMessage

The error message buffer.

errorMessageSize

The error message buffer size.

Returns
A value of 0 on success, -1 on error.

Returns 0 on success, -1 on error. The caller provides the errorMessage buffer and size. The suggested error message buffer size is 1050.

Initialize from an AE Environment.

Used to get a data connection from an AE Environment.

Data Structures

struct NzaeAggInitialization

An argument to function nzaeAggInitialize. Output parameters are handle and errorMessage.

struct NzaeInitialization

Argument to function nzaeInitialize. Output parameters are handle and errorMessage.

struct NzaeShpInitialization

Argument to function nzaeShpInitialize. Output parameters are handle and errorMessage.

Functions

NzaeAggRcCode nzaeAggInitialize(NzaeAggInitialization *arg)
Initialization to be called near the beginning of the process.

NzaeRcCode nzaeInitialize(NzaeInitialization *arg) Initialization must be called near the beginning of the process.

NzaeShpRcCode nzaeShpInitialize(NzaeShpInitialization *arg)
Initialization to be called near the beginning of the process.

Detailed Description

Used to get a data connection from an AE Environment.

Function Documentation

NzaeAggRcCode nzaeAggInitialize(NzaeAggInitialization *arg)
Initialization to be called near the beginning of the process.

Parameters

NzaeAggInitialization arg

The initialization argument.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeRcCode nzaeInitialize(NzaeInitialization *arg)
Initialization must be called near the beginning of the process.

Parameters

NzaeInitialization arg

The initialization argument.

Returns

NzaeRcCode

The function return code.

NzaeShpRcCode nzaeShpInitialize(NzaeShpInitialization *arg)
Initialization to be called near the beginning of the process.

Parameters

NzaeShpInitialization arg

The initialization argument.

Returns

NzaeShpRcCode

The Shaper return code.

Remote Connection Point.

A Remote Connection Point is how the Netezza system addresses a Remote AE.

Functions

`const char* nzaeconptBuildFileName(NZAECONPT_HANDLE handle)`

Builds and returns the file type connection point fully qualified name for file format connection proto-cols, such as Unix Sockets.

`void nzaeconptClose(NZAECONPT_HANDLE handle)` Closes the connection point.

`NZAECONPT_HANDLE nzaeconptCreate()`
Creates and returns a new NZAECONPT_HANDLE handle.

`int32_t nzaeconptGetDataSliceId(const NZAECONPT_HANDLE handle)` Gets the data slice ID for a connection point.

`const char* nzaeconptGetName(const NZAECONPT_HANDLE handle)` Returns the connection point name.

`int32_t nzaeconptGetSessionId(const NZAECONPT_HANDLE handle)` Gets the session ID for a connection point.

`int64_t nzaeconptGetTransactionId(const NZAECONPT_HANDLE handle)` Gets the transaction ID setting for a connection point.

`NzaeConptType nzaeconptGetType(const NZAECONPT_HANDLE handle)` Get the connection point type.

`void nzaeconptSetDataSliceId(NZAECONPT_HANDLE handle, int32_t dataSliceId)` Optionally sets the dataslice ID.

`int nzaeconptSetName(NZAECONPT_HANDLE handle, const char *name)`
Optionally sets the connection point name. Returns 0 on success, -1 on error.

`void nzaeconptSetSessionId(NZAECONPT_HANDLE handle, int32_t sessionId)` Optionally sets the session ID.

`void nzaeconptSetTransactionId(NZAECONPT_HANDLE handle, int64_t transactionId)` Optionally sets the transaction ID.

`void nzaeconptSetType(NZAECONPT_HANDLE handle, NzaeConptType conptType)`
Optional Function: manually set the connection point type. Usually the connection point uses a default type based on the AE Environment variables of the AE process such as NZAE_REMOTE.

Enumerations

`enum NzaeConptType {`
`NZAE_CONPT_UNKNOWN= 0, NZAE_CONPT_REMOTE,`
`NZAE_CONPT_EXTERNAL }` Connection point types.

Detailed Description

A Remote Connection Point is how the Netezza system addresses a Remote AE.

Function Documentation

const char* nzaeconptBuildFileName(NZAECONPT_HANDLE handle)

Builds and returns the file type connection point fully qualified name for file format connection protocols, such as Unix Sockets.

Parameters

handle

The connection point handle.

Returns

The connection point file name; NULL on error.

Returns NULL on error. The name format is based on name, transaction ID, session ID and data slice ID, of which one or more must be specified. This function is used internally but may also be used for logging or diagnostic purposes. The qualified name format is subject to change between releases.

void nzaeconptClose(NZAECONPT_HANDLE handle)

Closes the connection point.

Parameters

handle

The connection point handle.

NZAECONPT_HANDLE nzaeconptCreate()

Creates and returns a new NZAECONPT_HANDLE handle.

Returns

The connection point handle.

int32_t nzaeconptGetDataSliceId(const NZAECONPT_HANDLE handle)

Gets the data slice ID for a connection point.

Parameters

handle

The connection point handle.

Returns

The connection point dataslice ID.

const char* nzaeconptGetName(const NZAECONPT_HANDLE handle)

Returns the connection point name.

Parameters

handle

The connection point handle.

Returns
The connection point name.

int32_t nzaeconptGetSessionId(const NZAECONPT_HANDLE handle) Gets the session ID for a connection point.

Parameters
handle
The connection point handle.

Returns
The connection point session ID.

int64_t nzaeconptGetTransactionId(const NZAECONPT_HANDLE handle) Gets the transaction ID setting for a connection point.

Parameters
handle
The connection point handle.

Returns
The connection point transaction ID.

NzaeConptType nzaeconptGetType(const NZAECONPT_HANDLE handle) Get the connection point type.

Parameters
handle
The connection point handle.

Returns
NzaeConptType
The connection point type as defined in NzaeConptType.

void nzaeconptSetDataSliceId(NZAECONPT_HANDLE handle, int32_t dataSliceId) Optionally sets the dataslice ID.

Parameters
handle
The connection point handle.
dataSliceId The
dataslice ID.

int nzaeconptSetName(NZAECONPT_HANDLE handle, const char *name)
Optionally sets the connection point name. Returns 0 on success, -1 on error.

Parameters
handle

The connection point handle.

name

The connection point name.

Returns

A value of 0 on success, -1 on error.

void nzaeconptSetSessionId(NZAECONPT_HANDLE handle, int32_t sessionId) Optionally sets the session ID.

Parameters

handle

The connection point handle.

sessionId

The session ID.

void nzaeconptSetTransactionId(NZAECONPT_HANDLE handle, int64_t transactionId) Optionally sets the transaction ID.

Parameters

handle

The connection point handle.

transactionId

The transaction ID.

void nzaeconptSetType(NZAECONPT_HANDLE handle, NzaeConptType conptType)

Optional Function: manually set the connection point type. Usually the connection point uses a default type based on the AE Environment variables of the AE process such as NZAE_RE-MOTE.

Parameters

handle

The connection point handle.

NzaeConptType conptType

The connection point type as defined in NzaeConptType.

Enumeration Type Documentation

enum NzaeConptType

Connection point types.

NZAE_CONPT_UNKNOWN NZAE_CONPT_REMOTE

NZAE_CONPT_EXTERNAL External AE is not supported

Remote Initialization.

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

Create a connection point.

Listen using that connection point.

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

Data Structures

struct NZAECONPT_HANDLE

The ConnectionPoint Handle. An opaque handle used with Connection Point AE functions.

struct NZAEREMPROT_HANDLE

The Remote Protocol Handle. An opaque handle used with Remote Protocol AE functions.

struct NzaeremprotInitialization

Initializes a Remote AE Notification Connection.

Typedefs

NzaeRemprotCallback

Callback typedef.

Functions

NzaeRemprotRcCode nzaeRemprotAcceptApi(NZAEREMPROT_HANDLE handle, NzaeApi *result) Returns an AE API Handle from the connection point.

NzaeRemprotRcCode nzaeRemprotAcceptApiWithTimeout(NZAEREMPROT_HANDLE handle, int timeoutMilliseconds, NzaeApi *result)
Returns an AE API Handle from the connection point.

NzaeRemprotRcCode nzaeRemprotAcceptEnvironment(NZAEREMPROT_HANDLE handle, NZAEENV_HANDLE *result)
Returns an AE Environment from the connection point.

NzaeRemprotRcCode nzaeRemprotAcceptEnvironmentWithTimeout(NZAEREMPROT_HANDLE handle, int timeoutMilliseconds, NZAEENV_HANDLE *result)
Returns an AE Environment from the connection point.

void nzaeRemprotClose(NZAEREMPROT_HANDLE handle) Closes a listener.

NzaeRemprotRcCode nzaeRemprotCreateListener(NzaeremprotInitialization *args) Creates a new listener on a connection point.

void nzaeRemprotFreeResources(NZAEREMPROT_HANDLE handle)
Releases resources such as handles and memory without shutting down the underlying communication connection.

int nzaeRemprotGetAcceptSocket(NZAEREMPROT_HANDLE handle) Returns the socket used to accept Remprot commands.

NzaeRemprotCallback nzaeRemprotGetCallback(NZAEREMPROT_HANDLE handle, void **userContext)
Gets the Remote protocol Callback. A remote protocol handler function is used to handle remote com-

mands such as stop and status.

`NzaeApiTypes nzaeRemprotGetEnvironmentApiType(NZAEENV_HANDLE hEnv)` Gets the API type from the environment.

`char* nzaeRemprotGetLastErrorText(NZAEREMPROT_HANDLE handle)` Gets the text of the last error.

`int32_t nzaeRemprotGetRemoteDataSliceId()`
Gets the remote dataslice ID from the environment.

`const char* nzaeRemprotGetRemoteName()`
Gets the remote name from the environment.

`int32_t nzaeRemprotGetRemoteSessionId()`
Gets the remote session ID from the environment.

`int64_t nzaeRemprotGetRemoteTransactionId()`
Gets the remote transaction ID from the environment.

`int nzaeRemprotIsError(NZAEREMPROT_HANDLE handle)`
Returns TRUE if an error has occurred; FALSE if not.

`void nzaeRemprotSetCallback(NZAEREMPROT_HANDLE handle, NzaeRemprotCallback call-back, void *userContext)`
Sets the Remote Protocol Callback. A remote protocol handler function is used to handle re-mote commands such as stop and status.

`NzaeRemprotRcCode nzaeRemprotWaitForPingOrStop(NZAEREMPROT_HANDLE handle, int *bStopCommand)`
Waits for ping or stop.

Enumerations

`enum NzaeRemprotCmd {`
`NZAE_REMPROT_CMD_REQUEST, NZAE_REMPROT_CMD_PING,`
`NZAE_REMPROT_CMD_STATUS, NZAE_REMPROT_CMD_STOP,`
`NZAE_REMPROT_CMD_CONTROL_DATA, NZAE_REMPROT_CMD_SIGNAL }`

Remote AE Messages. Only `NZAE_REMPROT_CMD_STATUS`, `NZAE_REMPROT_CMD_STOP`, `NZAE_REMPROT_CMD_SIGNAL`, and `NZAE_REMPROT_CMD_CONTROL_DATA` are received by a user call back function.

`enum NzaeRemprotRcCode {`
`NZAEREMPROT_RC_ERROR= -1, NZAEREMPROT_RC_NORMAL=`
`0, NZAEREMPROT_RC_TIMEOUT= 1 }`

Remote Protocol return codes.

Detailed Description

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

Create a connection point.

Listen using that connection point.

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

Typedef Documentation

```
typedef int(* NzaeRemprotCallback)(void *userContext, int code, int dataLen, const char *data, NzaeRemprotCallbackResult *result)
```

Callback typedef.

Parameters

userContext

Any user application-specific data. May be NULL.

code

The remote message received (NZAЕ_REMPROT_CMD_STATUS, NZAЕ_REMPROT_CMD_STOP, NZAЕ_REMPROT_CMD_SIGNAL, NZAЕ_REMPROT_CMD_CONTROL_DATA).

dataLen

The argument data length. May be 0.

data

The argument data. May be NULL.

result

The structure to place callback function result.

Returns

A value of 0 on success, -1 on error.

See Also

NzaeRemprotCmd

nzaeRemprotSetCallback

nzaeRemprotGetCallback

Function Documentation

```
NzaeRemprotRcCode nzaeRemprotAcceptApi(NZAEREMPROT_HANDLE handle, NzaeApi *result) Returns an AE API Handle from the connection point.
```

Parameters

handle

The remote protocol handle.

NzaeApi result

The accepted API.

Returns

NzaeRemprotRcCode

The return code.

The caller has ownership of the returned handle. Waits indefinitely.

```
NzaeRemprotRcCode nzaeRemprotAcceptApiWithTimeout(NZAEREMPROT_HANDLE handle, int timeoutMilliseconds, NzaeApi *result)
```

Returns an AE API Handle from the connection point.

Parameters

handle

The remote protocol handle.

NzaeApi result

The accepted API.

timeoutMilliseconds

The timeout in milliseconds.

Returns

NzaeRemprotRcCode

The return code.

The caller has ownership of the returned handle. Waits for a connection for the given number of milliseconds.

NzaeRemprotRcCode nzaeRemprotAcceptEnvironment(NZAEREMPROT_HANDLE handle, NZAEENV_HANDLE *result)

Returns an AE Environment from the connection point.

Parameters

handle

The remote protocol handle.

result

The accepted Environment handle.

Returns

NzaeRemprotRcCode

The return code.

The caller has ownership of the returned handle. Waits indefinitely.

NzaeRemprotRcCode nzaeRemprotAcceptEnvironmentWith-Timeout(NZAEREMPROT_HANDLE handle, int timeoutMilliseconds, NZAEENV_HANDLE *result)

Returns an AE Environment from the connection point.

Parameters

handle

The remote protocol handle.

result

The accepted Environment handle.

timeoutMilliseconds

The timeout in milliseconds.

Returns
NzaeRemprotRcCode

The return code.

The caller has ownership of the returned handle. Waits for a connection for the given number of mill-seconds.

void nzaeRemprotClose(NZAEREMPROT_HANDLE handle) Closes a listener.

Parameters
handle
 The remote protocol handle.

NzaeRemprotRcCode nzaeRemprotCreateListener(NzaeremprotInitialization *args) Creates a new listener on a connection point.

Parameters
NzaeremprotInitialization args
 The initialization arguments.

Returns
NzaeRemprotRcCode

The return code.

void nzaeRemprotFreeResources(NZAEREMPROT_HANDLE handle)

Releases resources such as handles and memory without shutting down the underlying communication connection.

Parameters
handle
 The remote protocol handle.

Usually called by a child process forked from a Remote AE parent. Note that nzaeRemprotFreeResources and nzaeRemprotClose are never called in the same process. Typically nzaeRemprotClose is called in a Remote AE.

int nzaeRemprotGetAcceptSocket(NZAEREMPROT_HANDLE handle) Returns the socket used to accept Remprot commands.

Parameters
handle
 The remote protocol handle.

Returns
 The remote socket.

Once identified, the socket can be used with Linux select or poll.

NzaeRemprotCallback nzaeRemprotGetCallback(NZAEREMPROT_HANDLE handle, void **userContext)

Gets the Remote protocol Callback. A remote protocol handler function is used to handle re-mote commands such as stop and status.

Parameters

handle

The remote protocol handle.

userContext

The returned argument to callback.

Returns

NzaeRemprotCallback

The callback.

NzaeApiTypes nzaeRemprotGetEnvironmentApiType(NZAEENV_HANDLE hEnv) Gets the API type from the environment.

Parameters

hEnv

The Environment handle.

Returns

NzaeApiTypes

API type

char* nzaeRemprotGetLastErrorText(NZAEREMPROT_HANDLE handle) Gets the text of the last error.

Parameters

handle

The remote protocol handle.

Returns

The message text of the last occurring error.

int32_t nzaeRemprotGetRemoteDataSliceId() Gets the remote dataslice ID from the environment.

Returns

The remote dataslice ID.

Set if the AE launcher is used. Returns -1 if a dataslice ID is not found.

const char* nzaeRemprotGetRemoteName()

Gets the remote name from the environment.

Returns
The Remote Name.

Set if the AE launcher is used. Returns NULL if a name is not found.

int32_t nzaeRemprotGetRemoteSessionId() Gets
the remote session ID from the environment.

Returns
The remote Session ID.

Set if the AE launcher is used. Returns -1 if a session ID is not found.

int64_t nzaeRemprotGetRemoteTransactionId() Gets
the remote transaction ID from the environment.

Returns
The remote transaction ID.

Set if the AE launcher is used. Returns -1 if a transaction ID is not found.

int nzaeRemprotIsError(NZAEREMPROT_HANDLE handle)

Returns TRUE if an error has occurred; FALSE if not.

Parameters
handle
The remote protocol handle.

Returns
TRUE if an error occurred.

**void nzaeRemprotSetCallback(NZAEREMPROT_HANDLE handle, NzaeRemprotCallback
callback, void *userContext)**

Sets the Remote Protocol Callback. A remote protocol handler function is used to handle remote com-mands such as stop and status.

Parameters
handle
The remote protocol handle.
**NzaeRemprotCallback
callback** The callback function.
userContext
The argument to callback.

**NzaeRemprotRcCode nzaeRemprotWaitForPingOrStop(NZAEREMPROT_HANDLE handle, int
*bStop-Command)**

Waits for ping or stop.

Parameters

handle

The remote protocol handle.

bStopCommand

The pointer to the returned boolean, which indicates whether the AE is stopped.

Returns

NzaeRemprotRcCode

The return code.

This function is not used in a normal data-driven Remote AE. The code to accept AE APIs and AE Environments by default services ping and stop requests.

Thus, this function is only used in a launched Remote AE used as a control program, not as a true data AE. A launched Remote AE calls this function once on execution to satisfy the Netezza system AE launcher.

You can handle subsequent pings and stops in one of the two following ways:

Use a dedicated thread that invokes the `nzaeRemprotWaitForPingOrStop` command while the function waits indefinitely until it receives a message or is interrupted. When the function is in the wait state, it returns if it is interrupted by a signal even if no ping or stop message is received.

Use the `select()` or `poll()` C function calls on the AE Remote Protocol (`remprot`) accept socket file descriptor to check for pending actions with an appropriate timeout. If a pending action is detected, call the `nzaeRemprotWaitForPingOrStop` command, which does not block and should return a boolean value immediately. If no pending actions are detected via `select` or `poll`, regular operations are done and the call to `nzaeRemprotWaitForPingOrStop` is skipped. The whole process will then repeat.

When using this function, be careful of race conditions.

`bStopCommand` evaluates to `TRUE` if a stop request has been received, otherwise it is `FALSE`.

Enumeration Type Documentation

enum `NzaeRemprotCmd`

Remote AE Messages. Only `NZAE_REMPROT_CMD_STATUS`, `NZAE_REMPROT_CMD_STOP`, `NZAE_REMPROT_CMD_SIGNAL`, and `NZAE_REMPROT_CMD_CONTROL_DATA` are received by a user call back function.

NZAE_REMPROT_CMD_REQUEST

NZAE_REMPROT_CMD_PING

NZAE_REMPROT_CMD_STATUS The Remote AE is queried for status. Status data may be re-turned.

NZAE_REMPROT_CMD_STOP The Remote AE is being stopped.

NZAE_REMPROT_CMD_CONTROL_DATA The Remote AE is being sent control data. Data may be returned.

NZAE_REMPROT_CMD_SIGNAL The Remote AE has received a supported signal.

See Also

NzaeRemprotCallback
 nzaeRemprotSetCallback
 nzaeRemprotGetCallback

enum NzaeRemprotRcCode
 Remote Protocol return codes.

NZAEREMPROT_RC_ERROR

NZAEREMPROT_RC_NORMAL

NZAEREMPROT_RC_TIMEOUT

See Also

Remote Initialization.

Data Connection APIs

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

Modules

Function

Function AEs are called from 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. Sizers are optionally called for Scalar 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.

Data Structures

struct NZAE_HANDLE

The Function Handle. An opaque handle used with Function AE functions.

Functions

void nzaeClose(NZAE_HANDLE handle)

Closes the handle when done.

NzaeRcCode nzaeDone(NZAE_HANDLE handle)

Indicates that the AE is finishing and does not get any more rows or output any more results.

`NzaeRcCode nzaeGetEnv(NZAE_HANDLE handle, const char *name, const char **result)` Gets an AE or system environment variable. The AE has precedence.

`void nzaeGetFirstEnvironmentEntry(NZAE_HANDLE handle, NzaeEnvironmentEntry *entry)` Returns the first environment entry.

`NzaeRcCode nzaeGetInputColumn(NZAE_HANDLE handle, int index, NzudsData **data)` Gets input column data. The index is zero-based.

`AeUserCode nzaeGetLastErrorCode(NZAE_HANDLE handle)` Gets the code for the last error that occurred.

`const char* nzaeGetLastErrorText(NZAE_HANDLE handle)`
Get the message text for the last error that occurred.

`const char* nzaeGetLibraryFullPath(NZAE_HANDLE h, const char *libraryName, bool caseSensitive)`
Gets the file path for a library name.

`NzaeSharedLibraryInfo* nzaeGetLibraryInfo(NZAE_HANDLE h)`
Returns `NzaeSharedLibraryInfo` of the shared library for the request.

`NzaeSharedLibraryInfo* nzaeGetLibraryProcessInfo(NZAE_HANDLE h)`
Returns `NzaeSharedLibraryInfo` of the shared library for the process.

`NzaeRcCode nzaeGetMetadata(NZAE_HANDLE handle, NzaeMetadata *arg)` Gets metadata about the AE.

`NzaeRcCode nzaeGetNext(NZAE_HANDLE handle)`
Gets the next input row; returns `NZAE_RC_END` at End of File.

`bool nzaeGetNextEnvironmentEntry(NZAE_HANDLE handle, NzaeEnvironmentEntry *entry)` Returns the next environment entry.

`NzaeRcCode nzaeGetNextPartition(NZAE_HANDLE handle)`
Gets the next partition; returns `NZAE_RC_END` at End of Partition.

`int nzaeGetNumberOfParameters(NZAE_HANDLE h)`
Returns the number of parameters.

`const char* nzaeGetParameter(NZAE_HANDLE h, int index)` Returns a parameter.

`NzaeRcCode nzaeGetRuntime(NZAE_HANDLE handle, NzaeRuntime *arg)` Gets runtime information about the AE.

`NzaeRcCode nzaeLog(NZAE_HANDLE handle, NzaeLogLevel level, const char *message)` Logs the specified message.

`NzaeRcCode nzaeOutputResult(NZAE_HANDLE handle)`
Outputs a result row containing the current column values.

`NzaeRcCode nzaePing(NZAE_HANDLE handle)`
Indicates that the AE is still active and not hanging.

NzeRcCode nzeUserError(NZAE_HANDLE handle, const char *_template,...) Indicates this AE has encountered an error condition.

Enumerations

```
enum NzeCorrelationType {
  NzeUnknownCorrelationType= 0, NzeUncorrelated= 1, NzeInnerCorrelation= 2,
  NzeLeftCorrelation= 3 }
```

Specialized information about how this AE is being invoked.

```
enum NzeRcCode {
  NZAE_RC_ERROR= -1, NZAE_RC_NORMAL= 0, NZAE_RC_END=
  1 }
```

Return codes from nze functions.

Detailed Description

Function AEs are called from Scalar or Table SQL Functions.

Function Documentation

void nzeClose(NZAE_HANDLE handle) Closes the handle when done.

Parameters

handle

The function handle.

NzeRcCode nzeDone(NZAE_HANDLE handle)

Indicates that the AE is finishing and does not get any more rows or output any more results.

Parameters

handle

The function handle.

Returns

NzeRcCode

The function return code.

NzeRcCode nzeGetEnv(NZAE_HANDLE handle, const char *name, const char **result) Gets an AE or system environment variable. The AE has precedence.

Parameters

handle

The function handle.

name

The variable name.

result

The variable value or NULL if not found.

Returns

NzaeRcCode

The function return code.

void nzaeGetFirstEnvironmentEntry(NZAE_HANDLE handle, NzaeEnvironmentEntry *entry) Returns the first environment entry.

Parameters

handle

The function handle.

NzaeEnvironmentEntry

entry First entry.

This function call is followed by repeated calls to `nzaeGetNextEnvironmentEntry`. The AE system owns the memory from this call.

NzaeRcCode nzaeGetInputColumn(NZAE_HANDLE handle, int index, NzudsData **data) Gets input column data. The index is zero-based.

Parameters

handle

The function handle.

index

The input index.

NzudsData data

The UDS data.

Returns

NzaeRcCode

The function return code.

`NzudsData` is defined in `nzuds.h`. The data belongs to the framework and should not be freed. Called after `nzaeGetNext` is used to return the next row.

AeUserCode nzaeGetLastErrorCode(NZAE_HANDLE handle) Gets the code for the last error that occurred.

Parameters

handle

The function handle.

Returns

AeUserCode

The function error code for the last occurring error.

const char* nzaeGetLastErrorText(NZAE_HANDLE handle)

Get the message text for the last error that occurred.

Parameters

handle

The function handle.

Returns

The message text of the last occurring error.

const char* nzaeGetLibraryFullPath(NZAE_HANDLE h, const char *libraryName, bool caseSensitive) Gets the file path for a library name.

Parameters

h

The function handle.

libraryName The

library name.

caseSensitive

If TRUE, the lookup is case-sensitive.

Returns

File path if found; NULL otherwise

Returns NULL if the library is not found. The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeGetLibraryInfo(NZAE_HANDLE h)

Returns NzaeSharedLibraryInfo of the shared library for the request.

Parameters

h

The function handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information.

The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeGetLibraryProcessInfo(NZAE_HANDLE h)

Returns NzaeSharedLibraryInfo of the shared library for the process.

Parameters

h

The function handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information

Returns NULL if this is not a Remote AE. The AE system owns the memory from this call.

NzaeRcCode nzaeGetMetadata(NZAE_HANDLE handle, NzaeMetadata *arg) Gets metadata about the AE.

Parameters

handle

The function handle.

NzaeMetadata arg

Metadata to be filled out. Created by the caller.

Returns

NzaeRcCode

The function return code.

NzaeRcCode nzaeGetNext(NZAE_HANDLE handle)

Gets the next input row; returns NZAE_RC_END at End of File.

Parameters

handle

The function handle.

Returns

NzaeRcCode

The function return code.

Invalidates previous data returned by nzaeGetInputColumn.

bool nzaeGetNextEnvironmentEntry(NZAE_HANDLE handle, NzaeEnvironmentEntry *entry)

Returns the next environment entry.

Parameters

handle

The function handle.

NzaeEnvironmentEntry

entry The next entry.

Returns FALSE

on end.

The first nzaeGetNextEnvironmentEntry must follow a call to nzaeGetFirstEnvironmentEntry. Returns FALSE on end. Key names may repeat but the current version of a keyname comes last. The AE system owns the memory from this call.

NzaeRcCode nzaeGetNextPartition(NZAE_HANDLE handle) Gets the next partition; returns NZAE_RC_END at End of Partition.

Parameters

handle

The function handle.

Returns

NzaeRcCode

The function return code.

Invalidates previous data returned by `nzaeGetInputColumn`.

int nzaeGetNumberOfParameters(NZAE_HANDLE h) Returns the number of parameters.

Parameters

h

The function handle.

Returns

The number of parameters.

const char* nzaeGetParameter(NZAE_HANDLE h, int index) Returns a parameter.

Parameters

h

The function handle.

index

The parameter index.

Returns

The parameter value.

The index is zero-based.

NzaeRcCode nzaeGetRuntime(NZAE_HANDLE handle, NzaeRuntime *arg) Gets runtime information about the AE.

Parameters

handle

The function handle.

NzaeRuntime arg

Runtime to be filled out. Created by the caller.

Returns

NzaeRcCode

The function return code.

NzaeRcCode nzaeLog(NZAE_HANDLE handle, NzaeLogLevel level, const char *message) Logs the specified message.

Parameters

handle

The function handle.

NzaeLogLevel

level The log level.

message

The log message.

Returns

NzaeRcCode

The function return code.

NzaeRcCode nzaeOutputResult(NZAE_HANDLE handle)

Outputs a result row containing the current column values.

Parameters

handle

The function handle.

Returns

NzaeRcCode

The function return code.

NzaeRcCode nzaePing(NZAE_HANDLE handle)

Indicates that the AE is still active and not hanging.

Parameters

handle

The function handle.

Returns

NzaeRcCode

The function return code.

NzaeRcCode nzaeUserError(NZAE_HANDLE handle, const char *_template,...) Indicates this AE has encountered an error condition.

Parameters

handle

The function handle.

_template

The printf-style template.

Returns

NzaeRcCode

The function return code.

Implies nzaeDone. Message is built like printf.

Enumeration Type Documentation

enum NzaeCorrelationType

Specialized information about how this AE is being invoked.

NzaeUnknownCorrelationType

NzaeUncorrelated

NzaeInnerCorrelation

NzaeLeftCorrelation

See Also

NzaeMetadata

enum NzaeRcCode

Return codes from nzae functions.

NZAE_RC_ERROR

NZAE_RC_NORMAL

NZAE_RC_END

See Also

Function

Aggregate

Aggregate AEs are called from Aggregate SQL Functions.

Data Structures

struct NZAEAGG_HANDLE

The Aggregate Handle. An opaque handle used with Aggregate AE functions.

struct NzaeAggAccumulate

The Accumulate structure.

struct NzaeAggFieldFunctions

Read and write record functions for Aggregation.

struct NzaeAggFinalResult

The Final Result structure.

struct NzaeAggInitializeState

The InitializeState structure.

struct NzaeAggMerge

The Merge structure.

struct NzaeAggMetadata

NzaeAggMetatadata.

struct NzaeAggReadOnlyFieldFunctions
Read-only record functions for Aggregation.

Functions

void nzaeAggClose(NZAEAGG_HANDLE
handle) Closes the handle when done.

NzaeAggRcCode nzaeAggGetEnv(NZAEAGG_HANDLE handle, const char *name, const
char **result)

Gets the AE or system environment variable. The AE variable has precedence.

void nzaeAggGetFirstEnvironmentEntry(NZAEAGG_HANDLE handle,
NzaeEnvironmentEntry *entry)

Returns the first environment entry.

AeUserCode nzaeAggGetLastErrorCode(NZAEAGG_HANDLE
handle) Gets the code for the last error that occurred.

const char* nzaeAggGetLastErrorText(NZAEAGG_HANDLE
handle) Gets the message text for the last error that occurred.

const char* nzaeAggGetLibraryFullPath(NZAEAGG_HANDLE h, const char *libraryName,
bool caseSensitive)

Gets the file path for the library name.

NzaeSharedLibraryInfo* nzaeAggGetLibraryInfo(NZAEAGG_HANDLE h) Returns
NzaeSharedLibraryInfo for the requested Shared Library information.

NzaeSharedLibraryInfo* nzaeAggGetLibraryProcessInfo(NZAEAGG_HANDLE h)
Returns NzaeSharedLibraryInfo shared library information for the process. Returns NULL if
the AE is not Remote. The AE system owns the memory from this call.

bool nzaeAggGetNextEnvironmentEntry(NZAEAGG_HANDLE handle,
NzaeEnvironmentEntry *entry)

Returns the next environment entry.

int nzaeAggGetNumberOfParameters(NZAEAGG_HANDLE
h) Returns the number of parameters.

const char* nzaeAggGetParameter(NZAEAGG_HANDLE h, int
index) Returns the parameter.

NzaeAggRcCode nzaeAggGetRuntime(NZAEAGG_HANDLE handle, NzaeRuntime
*arg) Gets runtime information about the AE Aggregate.

const char* nzaeAggGetSystemLogFileName()
Gets the AE Aggregate System Log File name.

NzaeAggType nzaeAggGetType(NZAEAGG_HANDLE
handle) Returns the Aggregation Type.

NzaeAggRcCode nzaeAggLog(NZAEAGG_HANDLE handle, NzaeLogLevel level, const char
*mes-sage)

Logs the specified message.

void* nzaeAggNext(NZAEAGG_HANDLE handle, NzaeAggMessageType *messageType) Gets the next aggregation message.

NzaeAggRcCode nzaeAggPing(NZAEAGG_HANDLE handle)
Indicates that the AE Aggregate is still active and not hanging.

NzaeAggRcCode nzaeAggUpdate(NZAEAGG_HANDLE handle) Updates the result to the database.

NzaeAggRcCode nzaeAggUserError(NZAEAGG_HANDLE handle, const char *_template,...) Indicates that the AE encountered an error condition.

Enumerations

enum NzaeAggMessageType {
NZAEAGG_NOT_SET= -2, NZAEAGG_ERROR= -1, NZAEAGG_END= 0, NZAEAGG_INITIALIZE= 1, NZAEAGG_ACCUMULATE= 2, NZAEAGG_MERGE= 3, NZAEAGG_FINAL_RESULT= 4 }
Aggregate message types.

enum NzaeAggRcCode {
NZAEAGG_RC_ERROR= -1, NZAEAGG_RC_NORMAL= 0 }
Return codes from nzaeAgg aggregate functions.

enum NzaeAggType {
NzaeAggUnknown, NzaeAggGrouped, NzaeAggAnalytic
} The Aggregate Function Type.

Detailed Description

Aggregate AEs are called from Aggregate SQL Functions.

Function Documentation

void nzaeAggClose(NZAEAGG_HANDLE handle) Closes the handle when done.

Parameters
handle
The aggregate handle.

NzaeAggRcCode nzaeAggGetEnv(NZAEAGG_HANDLE handle, const char *name, const char **result) Gets the AE or system environment variable. The AE variable has precedence.

Parameters
handle
The aggregate handle.
name
The variable name.
result
The output variable value or NULL if not found.

Returns

NzaeAggRcCode

The aggregate return code.

void nzaeAggGetFirstEnvironmentEntry(NZAEAGG_HANDLE handle, NzaeEnvironmentEntry *entry)

Returns the first environment entry.

Parameters

handle

The aggregate handle.

NzaeEnvironmentEntry

entry The first entry.

This function call is followed by repeated calls to `nzaeGetNextEnvironmentEntry`. The AE system owns the memory from this call.

AeUserCode nzaeAggGetLastErrorCode(NZAEAGG_HANDLE handle) Gets the code for the last error that occurred.

Parameters

handle

The aggregate handle.

Returns

AeUserCode

The aggregate error code.

const char* nzaeAggGetLastErrorText(NZAEAGG_HANDLE handle) Gets the message text for the last error that occurred.

Parameters

handle

The aggregate handle.

Returns

The text of the last error.

const char* nzaeAggGetLibraryFullPath(NZAEAGG_HANDLE h, const char *libraryName, bool caseSensitive)

Gets the file path for the library name.

Parameters

h

The aggregate handle.

libraryName The library name.

caseSensitive

If TRUE, the lookup is case-sensitive.

Returns

The file path if found; NULL otherwise.

Returns NULL if the library is not found. The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeAggGetLibraryInfo(NZAEAGG_HANDLE h)

Returns NzaeSharedLibraryInfo for the requested Shared Library information.

Parameters

h

The aggregate handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information.

The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeAggGetLibraryProcessInfo(NZAEAGG_HANDLE h)

Returns NzaeSharedLibraryInfo shared library information for the process. Returns NULL if the AE is not Remote. The AE system owns the memory from this call.

Parameters

h

The aggregate handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information.

bool nzaeAggGetNextEnvironmentEntry(NZAEAGG_HANDLE handle, NzaeEnvironmentEntry *entry) Returns the next environment entry.

Parameters

handle

The aggregate handle.

NzaeEnvironmentEntry

entry The next entry.

Returns FALSE

on end.

The first nzaeGetNextEnvironmentEntry must follow a call to nzaeGetFirstEnvironmentEntry. Returns FALSE on end. Key names may repeat but the current version of a keyname is given last. The AE system owns the memory from this call.

int nzaeAggGetNumberOfParameters(NZAEAGG_HANDLE h) Returns the number of parameters.

Parameters

h

The aggregate handle.

Returns

The number of parameters.

const char* nzaeAggGetParameter(NZAEAGG_HANDLE h, int index) Returns the parameter.

Parameters

h

The aggregate handle.

index

The parameter index.

Returns

The parameter value.

The Index is zero-based.

NzaeAggRcCode nzaeAggGetRuntime(NZAEAGG_HANDLE handle, NzaeRuntime *arg) Gets runtime information about the AE Aggregate.

Parameters

handle

The aggregate handle.

NzaeRuntime arg

The caller-created runtime to be filled out.

Returns

NzaeAggRcCode

The aggregate return code.

const char* nzaeAggGetSystemLogFileName()

Gets the AE Aggregate System Log File name.

Returns

The log file name

NzaeAggType nzaeAggGetType(NZAEAGG_HANDLE handle) Returns the Aggregation Type.

Parameters

handle

The aggregate handle.

Returns

NzaeAggType

The aggregate type.

NzaeAggRcCode nzaeAggLog(NZAEAGG_HANDLE handle, NzaeLogLevel level, const char *message) Logs the specified message.

Parameters

handle

The aggregate handle.

NzaeLogLevel

level The log level.

message

The log message.

Returns

NzaeAggRcCode

The aggregate return code.

void* nzaeAggNext(NZAEAGG_HANDLE handle, NzaeAggMessageType *messageType) Gets the next aggregation message.

Parameters

handle

The aggregate handle.

NzaeAggMessageType messageType

The returned message type.

Returns

The structure as void * .

Returns a NzaeAggInitialize, NzaeAggAccumulate , NzaeAggMerge , or NzaeAggFinalResult struct pointer. Use the messageType parameter to determine the return type, end of input, and error. Returns NULL on error or at the end of data.

NzaeAggRcCode nzaeAggPing(NZAEAGG_HANDLE handle)

Indicates that the AE Aggregate is still active and not hanging.

Parameters

handle

The aggregate handle.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode nzaeAggUpdate(NZAEAGG_HANDLE handle) Updates the result to the database.

Parameters

handle

The aggregate handle.

Returns

NzaeAggRcCode

A NzaeAggInitialize, NzaeAggAccumulate, NzaeAggMerge, or NzaeAggFinalResult struct pointer. Can be NULL on error.

See Also

NzaeAggMessageType

NzaeAggRcCode nzaeAggUserError(NZAEAGG_HANDLE handle, const char *_template,...) Indicates that the AE encountered an error condition.

Parameters

handle

The aggregate handle.

_template

The printf-style template.

Returns

NzaeAggRcCode

The aggregate return code.

The AE is complete and should exit after this call. The message is built like printf.

Enumeration Type Documentation

enum NzaeAggMessageType

Aggregate message types.

NZAEAGG_NOT_SET

NZAEAGG_ERROR

NZAEAGG_END

NZAEAGG_INITIALIZE

NZAEAGG_ACCUMULATE

NZAEAGG_MERGE

NZAEAGG_FINAL_RESULT

See Also

`nzaeAggNext`

enum `NzaeAggRcCode`

Return codes from `nzaeAgg` aggregate functions.

NZAEAGG_RC_ERROR

NZAEAGG_RC_NORMAL

See Also

`Aggregate`

enum `NzaeAggType`

The Aggregate Function Type.

NzaeAggUnknown

NzaeAggGrouped

NzaeAggAnalytic

Shaper and Sizer

Shapers are optionally called for Table Function AEs. Sizers are optionally called for Scalar Function AEs.

Data Structures

struct `NZAESHP_HANDLE`

The Shaper Handle. An opaque handle used with Shaper and Sizer AE functions.

Functions

`NzaeShpRcCode nzaeShpAddOutputColumn(NZAESHP_HANDLE handle, NzudsDataType dataType, const char *columnName)`

Adds Non string/numeric Output Columns.

`NzaeShpRcCode nzaeShpAddOutputColumnNumeric(NZAESHP_HANDLE handle, NzudsDataType dataType, const char *columnName, int precision, int scale)`

Adds Numeric Output Columns.

`NzaeShpRcCode nzaeShpAddOutputColumnString(NZAESHP_HANDLE handle, NzudsDataType dataType, const char *columnName, int size)`

Adds String Output Columns.

`void nzaeShpClose(NZAESHP_HANDLE handle)` Closes the handle when done.

`NzaeShpRcCode nzaeShpGetEnv(NZAESHP_HANDLE handle, const char *name, const char **result)` Gets an AE or system environment variable. AE has precedence.

`void nzaeShpGetFirstEnvironmentEntry(NZAESHP_HANDLE handle, NzaeEnvironmentEntry *entry)` Returns the first environment entry.

`NzaeShpRcCode nzaeShpGetInputColumn(NZAESHP_HANDLE handle, int index, NzudsData **data)` Gets the input column data.

`AeUserCode nzaeShpGetLastErrorCode(NZAESHP_HANDLE handle)` Gets the code for last error that occurred.

`const char* nzaeShpGetLastErrorText(NZAESHP_HANDLE handle)` Gets the message text for last error that occurred.

`const char* nzaeShpGetLibraryFullPath(NZAESHP_HANDLE h, const char *libraryName, bool caseSensitive)`

Gets the file path for a library name.

`NzaeSharedLibraryInfo* nzaeShpGetLibraryInfo(NZAESHP_HANDLE h)`

Returns `NzaeSharedLibraryInfo` of the shared library for this request.

`NzaeSharedLibraryInfo* nzaeShpGetLibraryProcessInfo(NZAESHP_HANDLE h)`

Return `NzaeSharedLibraryInfo` of the shared library for the process. Returns NULL if the AE is not remote.

`NzaeShpRcCode nzaeShpGetMetadata(NZAESHP_HANDLE handle, NzaeShpMetadata *arg)` Gets metadata about the AE Shaper.

`bool nzaeShpGetNextEnvironmentEntry(NZAESHP_HANDLE handle, NzaeEnvironmentEntry *entry)`

Returns the next environment entry.

`int nzaeShpGetNumberOfParameters(NZAESHP_HANDLE h)` Returns the number of parameters.

`int nzaeShpGetNumOutputColumns(NZAESHP_HANDLE handle)`

Returns the number of output columns added by the user.

`NzaeShpRcCode nzaeShpGetOutputColumnInfo(NZAESHP_HANDLE handle, int index, NzaeSh-pOutputColumnInfo *info)`

Gets information about an output column added by the user.

`const char* nzaeShpGetParameter(NZAESHP_HANDLE h, int index)` Returns a parameter.

`NzaeShpRcCode nzaeShpGetRuntime(NZAESHP_HANDLE handle, NzaeRuntime *arg)` Gets runtime information about the AE Shaper.

`const char* nzaeShpGetSystemLogFileName(NZAESHP_HANDLE handle)` Gets the AE System Log File name.

`NzaeShpRcCode nzaeShpGetUdfReturnType(NZAESHP_HANDLE handle, NzudsDataType *data-Type)`

For a UDF only, gets the predetermined single return data type.

`NzaeShpRcCode nzaeShpLog(NZAESHP_HANDLE handle, NzaeLogLevel level, const char *mes-sage)`

Logs the specified message.

`NzaeShpRcCode nzaeShpPing(NZAESHP_HANDLE handle)`

Indicates that the AE Shaper is still active and not hanging.

`NzaeShpRcCode nzaeShpSystemCatalogIsUpper(NZAESHP_HANDLE handle, bool *result)` Returns TRUE if the default for system catalog names is upper case.

NzaeShpRcCode **nzaeShpUpdate**(NZAESHP_HANDLE handle)

Updates the shape and size information in the Netezza system.

NzaeShpRcCode **nzaeShpUserError**(NZAESHP_HANDLE handle, const char *_template,...) Indicates that this AE has encountered an error condition.

Enumerations

enum **NzaeShpRcCode** {

NZAESHP_RC_ERROR= -1, NZAESHP_RC_NORMAL= 0

} Return codes from **nzaeShp** Shaper functions.

Detailed Description

Shapers are optionally called for Table Function AEs. Sizers are optionally called for Scalar Function AEs.

Function Documentation

NzaeShpRcCode **nzaeShpAddOutputColumn**(NZAESHP_HANDLE handle, NzudsDataType **dataType**, const char *_columnName)

Adds Non string/numeric Output Columns.

Parameters

handle

The Shaper handle.

NzudsDataType

dataType The data type.

columnName The column name.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode **nzaeShpAddOutputColumnNumeric**(NZAESHP_HANDLE handle, NzudsDataType **dataType**, const char *_columnName, int **precision**, int **scale**)

Adds Numeric Output Columns.

Parameters

handle

The Shaper handle.

NzudsDataType

dataType The data type.

columnName The column name.

precision

The column precision.

scale

The column scale.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode **nzaeShpAddOutputColumnString**(NZAESHP_HANDLE handle,
NzudsData-Type **dataType**, const char ***columnName**, int **size**)

Adds String Output Columns.

Parameters

handle

The Shaper handle.

NzudsDataType

dataType The data type.

columnName The
column name.

size

The column size.

Returns

NzaeShpRcCode

The Shaper return code.

void **nzaeShpClose**(NZAESHP_HANDLE
handle) Closes the handle when done.

Parameters

handle

The Shaper handle.

NzaeShpRcCode **nzaeShpGetEnv**(NZAESHP_HANDLE handle, const char ***name**, const
char ****result**)

Gets an AE or system environment variable. AE has precedence.

Parameters

handle

The Shaper handle.

name

The variable name.

result

The variable value or NULL if not found.

Returns

NzaeShpRcCode

The Shaper return code.

void nzaeShpGetFirstEnvironmentEntry(NZAESHP_HANDLE handle, NzaeEnvironmentEntry *entry) Returns the first environment entry.

Parameters

handle

The Shaper handle.

NzaeEnvironmentEntry

entry The first entry.

This function call is followed by repeated calls to `nzaeGetNextEnvironmentEntry`. The AE system owns the memory from this call.

NzaeShpRcCode nzaeShpGetInputColumn(NZAESHP_HANDLE handle, int index, NzudsData **data) Gets the input column data.

Parameters

handle

The Shaper handle.

index

The input index.

NzudsData data

The input data.

Returns

NzaeShpRcCode

The Shaper return code.

If `isConstant` is FALSE then the value is always NULL.

AeUserCode nzaeShpGetLastErrorCode(NZAESHP_HANDLE handle) Gets the code for last error that occurred.

Parameters

handle

The Shaper handle.

Returns

AeUserCode

The error code of the last occurring error.

const char* nzaeShpGetLastErrorText(NZAESHP_HANDLE handle) Gets the message text for last error that occurred.

Parameters

handle

The Shaper handle.

Returns

The message text of the last occurring error.

const char* nzaeShpGetLibraryFullPath(NZAESHP_HANDLE h, const char *libraryName, bool caseSensitive)

Gets the file path for a library name.

Parameters

h

The Shaper handle.

libraryName The library name.

caseSensitive

If TRUE, the lookup is case-sensitive.

Returns

The file path if found; NULL otherwise.

Returns NULL if the library is not found. The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeShpGetLibraryInfo(NZAESHP_HANDLE h)

Returns NzaeSharedLibraryInfo of the shared library for this request.

Parameters

h

The Shaper handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information

The AE system owns the memory from this call.

NzaeSharedLibraryInfo* nzaeShpGetLibraryProcessInfo(NZAESHP_HANDLE h)

Return NzaeSharedLibraryInfo of the shared library for the process. Returns NULL if the AE is not remote.

Parameters

h

The Shaper handle.

Returns

NzaeSharedLibraryInfo

The Shared Library information

The AE system owns the memory from this call.

NzaeShpRcCode nzaeShpGetMetadata(NZAESHP_HANDLE handle, NzaeShpMetadata *arg) Gets metadata about the AE Shaper.

Parameters

handle

The Shaper handle.

NzaeShpMetadata arg

The metadata to be filled out. Created by the caller.

Returns

NzaeShpRcCode

The Shaper return code.

bool nzaeShpNextEnvironmentEntry(NZAESHP_HANDLE handle, NzaeEnvironmentEntry *entry) Returns the next environment entry.

Parameters

handle

The Shaper handle.

NzaeEnvironmentEntry

entry The next entry.

Returns FALSE

on end.

The first nzaeGetNextEnvironmentEntry must follow a call to nzaeGetFirstEnvironmentEntry.

Returns FALSE on end. Key names may repeat but the current version of a keyname is given last.

The AE system owns the memory from this call.

int nzaeShpGetNumberOfParameters(NZAESHP_HANDLE h) Returns the number of parameters.

Parameters

h

The Shaper handle.

Returns

The number of parameters

int nzaeShpGetNumOutputColumns(NZAESHP_HANDLE handle)

Returns the number of output columns added by the user.

Parameters

handle

The Shaper handle.

Returns

The number of output columns.

NzaeShpRcCode **nzaeShpGetOutputColumnInfo**(NZAESHP_HANDLE handle, int index, NzaeShpOutputColumnInfo *info)

Gets information about an output column added by the user.

Parameters

handle

The Shaper handle.

index

The output column index.

NzaeShpOutputColumnInfo

info The output information.

Returns

NzaeShpRcCode

The Shaper return code.

const char* **nzaeShpGetParameter**(NZAESHP_HANDLE h, int index) Returns a parameter.

Parameters

h

The Shaper handle.

index

The parameter index.

Returns

Parameter value

The index is zero-based.

NzaeShpRcCode **nzaeShpGetRuntime**(NZAESHP_HANDLE handle, NzaeRuntime *arg) Gets runtime information about the AE Shaper.

Parameters

handle

The Shaper handle.

NzaeRuntime arg

The runtime to be filled out. Created by the caller.

Returns

NzaeShpRcCode

The Shaper return code.

const char* nzaeShpGetSystemLogFileName(NZAESHP_HANDLE handle) Gets the AE System Log File name.

Parameters

handle

The Shaper handle.

Returns

The log file name

NzaeShpRcCode nzaeShpGetUdfReturnType(NZAESHP_HANDLE handle, NzudsDataType *dataType) For a UDF only, gets the predetermined single return data type.

Parameters

handle

The Shaper handle.

NzudsDataType dataType

The return type.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode nzaeShpLog(NZAESHP_HANDLE handle, NzaeLogLevel level, const char *message) Logs the specified message.

Parameters

handle

The Shaper handle.

NzaeLogLevel

level The log level.

message

The log message.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode nzaeShpPing(NZAESHP_HANDLE handle)

Indicates that the AE Shaper is still active and not hanging.

Parameters

handle

The Shaper handle.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode nzaeShpSystemCatalogIsUpper(NZAESHP_HANDLE handle, bool *result) Returns TRUE if the default for system catalog names is upper case.

Parameters

handle

The Shaper handle.

result

TRUE if catalog is upper case.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode nzaeShpUpdate(NZAESHP_HANDLE handle)

Updates the shape and size information in the Netezza system.

Parameters

handle

The Shaper handle.

Returns

NzaeShpRcCode

The Shaper return code.

NzaeShpRcCode nzaeShpUserError(NZAESHP_HANDLE handle, const char *_template,...) Indicates that this AE has encountered an error condition.

Parameters

handle

The Shaper handle.

_template

The printf-stlye template.

Returns

NzaeShpRcCode

The Shaper return code.

The AE is complete and should exit after this call. Message is built like printf.

Enumeration Type Documentation

enum NzaeShpRcCode

Return codes from nzaeShp Shaper functions.

NZAESHP_RC_ERROR**NZAESHP_RC_NORMAL**

See Also

▲ Shaper and Sizer

Support APIs

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

Modules

AE Manager Functionality

AE Manager Functionality - end user control over AE launch and runtime behavior.

Date and Time Functions

Date and Time helper functions used to convert to and from the Netezza date and time formats.

Numeric Functions

Numeric Conversion Routines.

Runtime and Environment Information

Runtime, Environment, and Shared Library Information.

User Codes

Symbolic return codes that can be used to support multiple human languages.

Detailed Description

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

AE Manager Functionality

AE Manager Functionality - end user control over AE launch and runtime behavior.

Detailed Description

AE Manager Functionality - end user control over AE launch and runtime behavior.

Date and Time Functions

Date and Time helper functions used to convert to and from the Netezza date and time formats.

Functions

`int64_t nzaeIntervalToMilliseconds(const NzudsInterval *nzInterval)` Converts an NZ Interval to milliseconds.

`int64_t nzaeIntervalToSeconds(const NzudsInterval *nzInterval)` Converts an NZ Interval to seconds.

`void nzaeMillisecondsToInterval(int64_t milliseconds, NzudsInterval *nzInterval)` Convert Milliseconds to an NZ Interval.

`int64_t nzaeMillisecondsToNzTime(int32_t milliseconds)`

Converts Time in milliseconds to an NZ Time.

`int32_t nzaeMinutesToNzTimeTzOffset(int32_t minutes)` Convert Minutes to NZ TimeTz Offset.

`int64_t nzaeNzDateToPosixTimeMilliseconds(int32_t nzDate)`

Converts an NZ Date to an Epoch time in milliseconds.

`int64_t nzaeNzDateToPosixTimeSeconds(int32_t nzDate)`

Converts an NZ Date to an Epoch time in seconds.

`int64_t nzaeNzTimestampToPosixTimeMilliseconds(int64_t nzTimestamp)`

Converts an NZ Timestamp to Epoch time in milliseconds.

`int64_t nzaeNzTimestampToPosixTimeSeconds(int64_t nzTimestamp)`

Converts an NZ Timestamp to an Epoch time in seconds.

`int32_t nzaeNzTimeToMilliseconds(int64_t nzTime)`

Converts an NZ Time to time in milliseconds.

`int32_t nzaeNzTimeToSeconds(int64_t nzTime)`

Converts an NZ Time to time in seconds.

`int32_t nzaeNzTimeTzOffsetToMinutes(int32_t`

`nzTimeTzOffset)` Converts an NZ TimeTz Offset to minutes.

`int32_t nzaePosixTimeMillisecondsToNzDate(int64_t posixTimeMilliseconds)`

Converts an Epoch time in milliseconds to an NZ Date.

`int64_t nzaePosixTimeMillisecondsToNzTimestamp(int64_t posixTimeMilliseconds)`

Converts an Epoch time in milliseconds to an NZ Timestamp.

`int32_t nzaePosixTimeSecondsToNzDate(int64_t posixTimeSeconds)`

Converts an Epoch time in seconds to an NZ Date.

`int64_t nzaePosixTimeSecondsToNzTimestamp(int64_t posixTimeSeconds)`

Converts an Epoch time in seconds to an NZ Timestamp.

`void nzaeSecondsToInterval(int64_t seconds, NzudsInterval *nzInterval)` Convert Seconds to an NZ Interval.

`int64_t nzaeSecondsToNzTime(int32_t seconds)`

Converts Time in seconds to an NZ Time.

Detailed Description

Date and Time helper functions used to convert to and from the Netezza date and time formats.

Function Documentation

`int64_t nzaeIntervalToMilliseconds(const NzudsInterval *nzInterval)` Converts an NZ Interval to milliseconds.

▲ Parameters

NzudsInterval nzInterval

The NZ-encoded interval.

Returns

Milliseconds.

int64_t nzaeIntervalToSeconds(const NzudsInterval *nzInterval) Converts an NZ Interval to seconds.

Parameters

NzudsInterval nzInterval

The NZ-encoded interval.

Returns

Seconds.

void nzaeMillisecondsToInterval(int64_t milliseconds, NzudsInterval *nzInterval) Convert Milliseconds to an NZ Interval.

Parameters

milliseconds

Milliseconds.

NzudsInterval nzInterval

The interval output.

int64_t nzaeMillisecondsToNzTime(int32_t milliseconds)

Converts Time in milliseconds to an NZ Time.

Parameters

milliseconds

The time in milliseconds.

Returns

The NZ Time.

int32_t nzaeMinutesToNzTimeTzOffset(int32_t minutes) Convert Minutes to NZ TimeTz Offset.

Parameters

minutes

Minutes.

Returns

The NZ TimeTz offset.

int64_t nzaeNzDateToPosixTimeMilliseconds(int32_t nzDate)

Converts an NZ Date to an Epoch time in milliseconds.

Parameters

nzDate

NZ encoded date.

Returns

The Epoch time in milliseconds.

int64_t nzaeNzDateToPosixTimeSeconds(int32_t nzDate)

Converts an NZ Date to an Epoch time in seconds.

Parameters

nzDate

The NZ-encoded date.

Returns

The Epoch time in seconds.

int64_t nzaeNzTimestampToPosixTimeMilliseconds(int64_t nzTimestamp) Converts an NZ Timestamp to Epoch time in milliseconds.

Parameters

nzTimestamp NZ-
encoded timestamp.

Returns

The Epoch time in milliseconds.

int64_t nzaeNzTimestampToPosixTimeSeconds(int64_t nzTimestamp)

Converts an NZ Timestamp to an Epoch time in seconds.

Parameters

nzTimestamp

The NZ-encoded timestamp.

Returns

The Epoch time in seconds.

int32_t nzaeNzTimeToMilliseconds(int64_t nzTime)

Converts an NZ Time to time in milliseconds.

Parameters

nzTime

The NZ-encoded time.

Returns

Time in milliseconds

int32_t nzaeNzTimeToSeconds(int64_t nzTime)

Converts an NZ Time to time in seconds.

Parameters

nzTime

The NZ-encoded time.

Returns

The time in seconds.

**int32_t nzaeNzTimeTzOffsetToMinutes(int32_t
nzTimeTzOffset)** Converts an NZ TimeTz Offset to minutes.

Parameters

nzTimeTzOffset

The NZ TimeTz offset.

Returns

Minutes.

int32_t nzaePosixTimeMillisecondsToNzDate(int64_t posixTimeMilliseconds)

Converts an Epoch time in milliseconds to an NZ Date.

Parameters

posixTimeMilliseconds

The Posix time in milliseconds.

Returns

An NZ Date.

int64_t nzaePosixTimeMillisecondsToNzTimestamp(int64_t posixTimeMilliseconds)

Converts an Epoch time in milliseconds to an NZ Timestamp.

Parameters

posixTimeMilliseconds

The Posix time in milliseconds.

Returns

The NZ Timestamp.

int32_t nzaePosixTimeSecondsToNzDate(int64_t posixTimeSeconds)

Converts an Epoch time in seconds to an NZ Date.

Parameters

posixTimeSeconds

The Posix time in seconds.

Returns

The NZ Date.

int64_t nzaePosixTimeSecondsToNzTimestamp(int64_t posixTimeSeconds)

Converts an Epoch time in seconds to an NZ Timestamp.

Parameters

posixTimeSeconds

The Posix time in seconds.

Returns

The NZ Timestamp.

void nzaeSecondsToInterval(int64_t seconds, NzudsInterval *nzInterval) Convert Seconds to an NZ Interval.

Parameters

seconds

Seconds.

NzudsInterval nzInterval

The interval output.

int64_t nzaeSecondsToNzTime(int32_t seconds)

Converts Time in seconds to an NZ Time.

Parameters

seconds

The time in seconds.

Returns

The NZ Time.

Numeric Functions

Numeric Conversion Routines.

Data Structures

```
struct NzaeNumeric128BytesBigEndian
struct NzaeNumeric128BytesLittleEndian
struct NzaeNumeric32BytesBigEndian
struct NzaeNumeric32BytesLittleEndian
struct NzaeNumeric64BytesBigEndian
struct NzaeNumeric64BytesLittleEndian
```

Functions

double nzaeGetDoubleFromNumeric128(const NzudsNumeric128 *arg, int scale) Convert a Numeric128 to a double.

double nzaeGetDoubleFromNumeric32(const NzudsNumeric32 *arg, int scale) Converts a Numeric32 to a double.

double nzaeGetDoubleFromNumeric64(const NzudsNumeric64 *arg, int scale)

Converts a Numeric64 to a double.

Detailed Description

Numeric Conversion Routines.

Function Documentation

double nzaeGetDoubleFromNumeric128(const NzudsNumeric128 *arg, int scale) Convert a Numeric128 to a double.

Parameters

NzudsNumeric128

arg The Numeric 128.

scale The
scale.

Returns

The double value.

Due to size differences, this function may not work as expected for certain values.

double nzaeGetDoubleFromNumeric32(const NzudsNumeric32 *arg, int scale) Converts a Numeric32 to a double.

Parameters

NzudsNumeric32 arg

The Numeric 32.

scale The
scale.

Returns

The double value.

double nzaeGetDoubleFromNumeric64(const NzudsNumeric64 *arg, int scale) Converts a Numeric64 to a double.

Parameters

NzudsNumeric64 arg

The Numeric 64.

scale The
scale.

Returns

The double value.

Runtime and Environment Information

Runtime, Environment, and Shared Library Information.

Data Structures

struct NzaeEnvironmentEntry

The environment entry.

struct NzaeRuntime

Runtime information.

struct NzaeSharedLibraryInfo

Shared library information.

Enumerations

enum NzaeAdapterType {

NZAE_ADAPTER_OTHER= 0, NZAE_ADAPTER_UDTF= 1, NZAE_ADAPTER_UDF= 2,
NZAE_ADAPTER_UDA= 3 }

Adapter types.

enum NzaeLocus {

NZAE_LOCUS_POSTGRES= 0, NZAE_LOCUS_DBOS= 1, NZAE_LOCUS_SPU=
2 } The execution locus.

enum NzaeLogLevel {

NZAE_LOG_TRACE= 1, NZAE_LOG_DEBUG= 2

} Log levels.

Detailed Description

Runtime, Environment, and Shared Library Information.

Enumeration Type Documentation

enum NzaeAdapterType

Adapter types.

NZAE_ADAPTER_OTHER

NZAE_ADAPTER_UDTF

NZAE_ADAPTER_UDF

NZAE_ADAPTER_UDA

See Also

▲ [NzaeRuntime](#)

enum NzaeLocus

The execution locus.

NZAE_LOCUS_POSTGRES

NZAE_LOCUS_DBOS

NZAE_LOCUS_SPU

See Also

NzaeRuntime

enum NzaeLogLevel

Log levels.

NZAE_LOG_TRACE**NZAE_LOG_DEBUG**

See Also

nzaeLog

nzaeAggLog

nzaeShpLog

User Codes

Symbolic return codes that can be used to support multiple human languages.

Enumerations

```
enum AeUserCode {
  AE_UC_OK= 0, AE_UC_INTERNAL= 1, AE_UC_BAD_INPUT_INDEX= 2, AE_UC_BAD_OUTPUT_INDEX= 3,
  AE_UC_INVALID_NULL_PARM= 4, AE_UC_INVALID_NON_POSITIVE_PARM= 5,
  AE_UC_FEATURE_NOT_IMPLEMENTED= 6, AE_UC_INPUT_NOT_AVAILABLE= 7, AE_UC_INPUT_NOT_ALLOWED= 8,
  AE_UC_OUTPUT_NOT_ALLOWED= 9, AE_UC_INVALID_CONVERSION= 10,
  AE_UC_INVALID_HANDLE= 11, AE_UC_UPDATE_EXPECTED= 12, AE_UC_UPDATE_NOT_EXPECTED= 13,
  AE_UC_INVALID_SHAPER_TYPE= 14, AE_UC_INVALID_SHAPER_PRECISION= 15,
  AE_UC_INVALID_SIZER_TYPE= 16, AE_UC_INVALID_SIZER_COUNT= 17, AE_UC_INVALID_SIZER_CATALOG_IS_UPPER= 18,
  AE_UC_INVALID_SIZER_GET_UDF_RETURN_TYPE= 19, AE_UC_BAD_NEXT_ENVIRONMENT= 20, AE_UC_NOT_IN_PARTITION_MODE= 21,
  AE_UC_PARTITION_NOT_ALLOWED= 22, AE_UC_BAD_SQL_PARAMETER_INDEX= 23 }
```

Detailed Description

Symbolic return codes that can be used to support multiple human languages.

Enumeration Type Documentation

enum AeUserCode

AE_UC_OK**AE_UC_INTERNAL****AE_UC_BAD_INPUT_INDEX****AE_UC_BAD_OUTPUT_INDEX****AE_UC_INVALID_NULL_PARM****AE_UC_INVALID_NON_POSITIVE_PARM****AE_UC_FEATURE_NOT_IMPLEMENTED**

AE_UC_INPUT_NOT_AVAILABLE
AE_UC_INPUT_NOT_ALLOWED
AE_UC_OUTPUT_NOT_ALLOWED
AE_UC_INVALID_CONVERSION
AE_UC_INVALID_HANDLE
AE_UC_UPDATE_EXPECTED
AE_UC_UPDATE_NOT_EXPECTED
AE_UC_INVALID_SHAPER_TYPE
AE_UC_INVALID_SHAPER_PRECISION
AE_UC_INVALID_SIZER_TYPE
AE_UC_INVALID_SIZER_COUNT
AE_UC_INVALID_SIZER_CATALOG_IS_UPPER
AE_UC_INVALID_SIZER_GET_UDF_RETURN_TYPE
AE_UC_BAD_NEXT_ENVIRONMENT
AE_UC_NOT_IN_PARTITION_MODE
AE_UC_PARTITION_NOT_ALLOWED
AE_UC_BAD_SQL_PARAMETER_INDEX

Data Type Support.

The data APIs work with these data types.

Data Structures

```
struct NzudsData  
struct NzudsInterval  
struct NzudsNumeric128  
struct NzudsNumeric32  
struct NzudsNumeric64  
struct NzudsTimeTz
```

Typedefs

NzudsNumericDigit

Enumerations

```
enum NzudsDataType {  
  NZUDSUDX_UNKNOWN=-1, NZUDSUDX_FIXED, NZUDSUDX_VARIABLE, NZUDSUDX_NATION-  
  AL_FIXED, NZUDSUDX_NATIONAL_VARIABLE, NZUDSUDX_BOOL, NZUDSUDX_DATE, NZUD-  
  SUDX_TIME, NZUDSUDX_TIMETZ, NZUDSUDX_NUMERIC32, NZUDSUDX_NUMERIC64, NZUD-
```

SUDX_NUMERIC128, NZUDSUDX_FLOAT, NZUDSUDX_DOUBLE, NZUDSUDX_INTERVAL,
 NZUDSUDX_INT8, NZUDSUDX_INT16, NZUDSUDX_INT32, NZUDSUDX_INT64, NZUDSUDX_TIMESTAMP,
 NZUDSUDX_GEOMETRY, NZUDSUDX_VARBINARY, NZUDSUDX_MAX_TYPE }

Detailed Description

The data APIs work with these data types.

Typedef Documentation

typedef int32_t NzudsNumericDigitNzudsNumericDigit

Digit definition for numeric data types

▲ See Also

► Data Type Support.

Enumeration Type Documentation

enum NzudsDataType

NZUDSUDX_UNKNOWN unknown data type

NZUDSUDX_FIXED fixed string **NZUDSUDX_VARIABLE**

variable string **NZUDSUDX_NATIONAL_FIXED** fixed

national string **NZUDSUDX_NATIONAL_VARIABLE**

variable national string **NZUDSUDX_BOOL** boolean

NZUDSUDX_DATE date

NZUDSUDX_TIME time

NZUDSUDX_TIMETZ time zone

NZUDSUDX_NUMERIC32 numeric 32

NZUDSUDX_NUMERIC64 numeric 64

NZUDSUDX_NUMERIC128 numeric

128 **NZUDSUDX_FLOAT** float

NZUDSUDX_DOUBLE double

NZUDSUDX_INTERVAL interval

NZUDSUDX_INT8 1 byte integer

NZUDSUDX_INT16 2 byte integer

NZUDSUDX_INT32 4 byte integer

NZUDSUDX_INT64 8 byte integer

NZUDSUDX_TIMESTAMP time stamp

NZUDSUDX_GEOMETRY geometry

NZUDSUDX_VARBINARY

NZUDSUDX_MAX_TYPE greater than any data type enum value

See Also

▲ [Data Type Support](#).

CHAPTER 2

Data Structure Documentation

NZAE_HANDLE Struct Reference

The Function Handle. An opaque handle used with Function AE functions.

Detailed Description

The Function Handle. An opaque handle used with Function AE functions.

See Also

▲ Function

NZAEAGG_HANDLE Struct Reference

The Aggregate Handle. An opaque handle used with Aggregate AE functions.

Detailed Description

The Aggregate Handle. An opaque handle used with Aggregate AE functions.

See Also

▲ Aggregate

NzaeAggAccumulate Struct Reference

The Accumulate structure.

Public Attributes

input

state

Detailed Description

The Accumulate structure.

See Also

▲ [nzaeAggNext](#)

Member Data Documentation

NzaeAggReadOnlyFieldFunctions input

NzaeAggFieldFunctions state

NzaeAggFieldFunctions Struct Reference

Read and write record functions for Aggregation.

Public Member Functions

NzaeAggRcCode(* [getValue](#))(NZAEOAGG_HANDLE handle, int index, NzudsData **data) Get Value.

NzaeAggRcCode(* [isNull](#))(NZAEOAGG_HANDLE handle, int index, bool *result) Specifieds if the field is NULL.

NzaeAggRcCode(* [setBool](#))(NZAEOAGG_HANDLE handle, int index, bool value) Sets the bool field value.

NzaeAggRcCode(* [setDate](#))(NZAEOAGG_HANDLE handle, int index, int32_t value) Sets the date field value.

NzaeAggRcCode(* [setDouble](#))(NZAEOAGG_HANDLE handle, int index, double value) Sets the double field value.

NzaeAggRcCode(* [setFloat](#))(NZAEOAGG_HANDLE handle, int index, float value) Sets the float field value.

NzaeAggRcCode(* [setInt16](#))(NZAEOAGG_HANDLE handle, int index, int16_t value) Sets the int16 field value.

NzaeAggRcCode(* [setInt32](#))(NZAEOAGG_HANDLE handle, int index, int32_t value) Sets the int32 field value.

NzaeAggRcCode(* [setInt64](#))(NZAEOAGG_HANDLE handle, int index, int64_t value) Sets the int64 field value.

NzaeAggRcCode(* [setInt8](#))(NZAEOAGG_HANDLE handle, int index, int8_t value) Sets the int8 field value.

NzaeAggRcCode(* setInterval)(NZAEGG_HANDLE handle, int index, NzudsInterval *value) Sets the interval field value.

NzaeAggRcCode(* setNull)(NZAEGG_HANDLE handle, int index) Sets the field value to NULL.

NzaeAggRcCode(* setNumeric128)(NZAEGG_HANDLE handle, int index, const NzudsNumeric128 *value)
Sets the numeric128 field value.

NzaeAggRcCode(* setNumeric32)(NZAEGG_HANDLE handle, int index, const NzudsNumeric32 *value) Sets the numeric32 field value.

NzaeAggRcCode(* setNumeric64)(NZAEGG_HANDLE handle, int index, const NzudsNumeric64 *value) Sets the numeric64 field value.

NzaeAggRcCode(* setString)(NZAEGG_HANDLE handle, int index, const char *value) Sets the string field value.

NzaeAggRcCode(* setStringLength)(NZAEGG_HANDLE handle, int index, const char *value, int length) Sets the string field value.

NzaeAggRcCode(* setTime)(NZAEGG_HANDLE handle, int index, int64_t value) Sets the time field value.

NzaeAggRcCode(* setTimeStamp)(NZAEGG_HANDLE handle, int index, int64_t value) Sets the timestamp field value.

NzaeAggRcCode(* setTimeTz)(NZAEGG_HANDLE handle, int index, const NzudsTimeTz *value) Sets the timeTz field value.

NzaeAggRcCode(* setValue)(NZAEGG_HANDLE handle, int index, NzudsData *data) Sets the field value.

Public Attributes

metadata

Detailed Description

Read and write record functions for Aggregation.

See Also

- NzaeAggInitializeState
- NzaeAggAccumulate
- NzaeAggMerge
- NzaeAggFinalResult

Public Member Function Documentation

NzaeAggRcCode(* getValue)(NZAEGG_HANDLE handle, int index, NzudsData **data) Get Value.

Parameters

handle

The aggregate handle.

index

The field index.

data

The Returned Field data.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* isNull)(NZAEGG_HANDLE handle, int index, bool *result) Specifies if the field is NULL.

Parameters

handle

The aggregate handle.

index

The field index.

result

TRUE if NULL.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setBool)(NZAEGG_HANDLE handle, int index, bool value) Sets the bool field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The bool value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setDate)(NZAEGG_HANDLE handle, int index, int32_t value) Sets the date field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The date value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setDouble)(NZAEGG_HANDLE handle, int index, double value) Sets the double field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The double value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setFloat)(NZAEGG_HANDLE handle, int index, float value) Sets the float field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The float value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setInt16)(NZAEGG_HANDLE handle, int index, int16_t value) Sets the int16 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The int16 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setInt32)(NZAEGG_HANDLE handle, int index, int32_t value) Sets the int32 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The int32 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setInt64)(NZAEGG_HANDLE handle, int index, int64_t value) Sets the int64 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The int64 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setInt8)(NZAEGG_HANDLE handle, int index, int8_t value) Sets the int8 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The int8 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setInterval)(NZAEGG_HANDLE handle, int index, NzudsInterval *value) Sets the interval field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The interval value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setNull)(NZAEGG_HANDLE handle, int index) Sets the field value to NULL.

Parameters

handle

The aggregate handle.

index

The field index.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setNumeric128)(NZAEGG_HANDLE handle, int index, const NzudsNumeric128 *value)

Sets the numeric128 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The numeric128 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setNumeric32)(NZAEGG_HANDLE handle, int index, const NzudsNumeric32 *value)

Sets the numeric32 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The numeric32 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setNumeric64)(NZAEGG_HANDLE handle, int index, const NzudsNumeric64 *value)

Sets the numeric64 field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The numeric64 value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setString)(NZAEGG_HANDLE handle, int index, const char *value) Sets the string field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The string value, with length determined by strlen.

Returns

NzaeAggRcCode

The aggregate return code.

The value is expected to be NULL-terminated. A copy of the string value is created.

NzaeAggRcCode(* setStringLength)(NZAEGG_HANDLE handle, int index, const char *value, int length)

Sets the string field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The string value.

length

The length of the string.

Returns

NzaeAggRcCode

The aggregate return code.

The string length is determined by the length argument. A copy of the string value is created. NULL termination does not apply.

NzaeAggRcCode(* setTime)(NZAEGG_HANDLE handle, int index, int64_t value) Sets the time field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The time value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setTimeStamp)(NZAEGG_HANDLE handle, int index, int64_t value) Sets the timestamp field value.

Parameters

handle

The aggregate handle.

value

The timestamp value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setTimeTz)(NZAEGG_HANDLE handle, int index, const NzudsTimeTz *value) Sets the timeTz field value.

Parameters

handle

The aggregate handle.

index

The field index.

value

The timeTz value.

Returns

NzaeAggRcCode

The aggregate return code.

NzaeAggRcCode(* setValue)(NZAEGG_HANDLE handle, int index, NzudsData *data) Sets the field value.

Parameters

handle

The aggregate handle.

index

The field index.

data

The value.

Returns

NzaeAggRcCode

The aggregate return code.

Member Data Documentation

NzaeAggMetadata metadata

NzaeAggFinalResult Struct Reference

The Final Result structure.

Public Attributes

inputState
result

Detailed Description

The Final Result structure.

See Also

▲ nzaeAggNext

Member Data Documentation

NzaeAggReadOnlyFieldFunctions inputState

NzaeAggFieldFunctions result

NzaeAggInitialization Struct Reference

An argument to function nzaeAggInitialize. Output parameters are handle and errorMessage.

Public Attributes

errorMessage
handle
hEnv
ldkVersion

Detailed Description

An argument to function nzaeAggInitialize. Output parameters are handle and errorMessage.

See Also

▲ nzaeAggInitialize

Member Data Documentation

char errorMessage[NZAEAGG_ERROR_MESSAGE_LENGTH]

NZAEAGG_HANDLE handle

NZAEENV_HANDLE hEnv

int ldkVersion

NzaeAggInitializeState Struct Reference

The InitializeState structure.

Public Attributes

state

Detailed Description

The InitializeState structure.

See Also

▲ [nzaeAggNext](#)

Member Data Documentation

NzaeAggFieldFunctions state

NzaeAggMerge Struct Reference

The Merge structure.

Public Attributes

inputState

state

Detailed Description

The Merge structure.

See Also

nzaeAggNext

Member Data Documentation

NzaeAggReadOnlyFieldFunctions inputState

NzaeAggFieldFunctions state

NzaeAggMetadata Struct Reference

NzaeAggMetatadata.

Public Attributes

numColumns
scales
sizes
types

Detailed Description

NzaeAggMetatadata.

See Also

NzaeAggReadOnlyFieldFunctions
NzaeAggFieldFunctions

Member Data Documentation

int numColumns

int* scales

int* sizes

NzudsDataType* types

NzaeAggReadOnlyFieldFunctions Struct Reference

Read-only record functions for Aggregation.

Public Member Functions

NzaeAggRcCode(* getValue)(NZAEGG_HANDLE handle, int index, NzudsData **data)

Get Value.

NzaeAggRcCode(* isNull)(NZAEGG_HANDLE handle, int index, bool
*result) Specifies whether the field is NULL.

Public Attributes

metadata

Detailed Description

Read-only record functions for Aggregation.

See Also

NzaeAggAccumulate

NzaeAggMerge

NzaeAggFinalResult

Public Member Function Documentation

**NzaeAggRcCode(* getValue)(NZAEGG_HANDLE handle, int index, NzudsData
data) Get Value.

Parameters

handle

The aggregate handle.

index

The field index.

data

The returned Field data.

Returns

NzaeAggRcCode

The aggregate return code.

**NzaeAggRcCode(* isNull)(NZAEGG_HANDLE handle, int index, bool
*result)** Specifies whether the field is NULL.

Parameters

handle

The aggregate handle.

index

The field index.

result

TRUE if NULL.

Returns

NzaeAggRcCode

The aggregate return code.

Member Data Documentation

NzaeAggMetadata metadata

NzaeApi Struct Reference

Contains a data connection handle.

Public Attributes

```
apiType
union {
    aggregation
    any
    function
    shaper
} handle
```

Detailed Description

Contains a data connection handle.

See Also

```
nzaeRemprotAcceptApi
nzaeRemprotAcceptApiWithTimeout
nzaeLocprotGetApi
```

Member Data Documentation

NZAEAGG_HANDLE aggregation
Aggregation AE Data Connection Handle.

void* any Used
internally.

NzaeApiTypes apiType
Indicates the type of data connection handle stored in the union.

NZAE_HANDLE function

Function AE Data Connection Handle.

```
handle  
union { ... }
```

NZAESHP_HANDLE shaper
Shaper or Sizer AE Data Connection Handle.

NZAECONPT_HANDLE Struct Reference

The ConnectionPoint Handle. An opaque handle used with Connection Point AE functions.

Detailed Description

The ConnectionPoint Handle. An opaque handle used with Connection Point AE functions.

See Also

▲ Remote Initialization.

NZAEENV_HANDLE Struct Reference

NzaeEnvironmentEntry Struct Reference

The environment entry.

Public Attributes

```
name  
value
```

Detailed Description

The environment entry.

See Also

```
nzaeGetFirstEnvironmentEntry  
nzaeGetNextEnvironmentEntry  
nzaeAggGetFirstEnvironmentEntry  
nzaeAggGetNextEnvironmentEntry  
nzaeShpGetFirstEnvironmentEntry  
nzaeShpGetNextEnvironmentEntry
```


Member Data Documentation

const char* name

const char* value

NzaeInitialization Struct Reference

Argument to function nzaeIntialize. Output parameters are handle and errorMessage.

Public Attributes

errorMessage

handle

hEnv

ldkVersion

Detailed Description

Argument to function nzaeIntialize. Output parameters are handle and errorMessage.

See Also

▲ nzaeInitialize

Member Data Documentation

char errorMessage[NZAE_ERROR_MESSAGE_LENGTH]

NZAE_HANDLE handle

NZAEENV_HANDLE hEnv

int ldkVersion

NzaeMetadata Struct Reference

Metadata describing the input and output rows of the AE.

Public Attributes

correlationType

hasOver

hasPartition

hasSort

inputColumnCount
inputIsConstant
inputScales
inputSizes
inputTypes
oneOutputRowRestriction
outputColumnCount
outputScales
outputSizes
outputTypes

Detailed Description

Metadata describing the input and output rows of the AE.

The memory pointed to by inputTypes and outputTypes belongs to the handle and should not be freed by the user.

See Also

▲ [nzaeGetMetadata](#)

Member Data Documentation

NzaeCorrelationType correlationType
Correlation: see definition of NzaeCorrelationType.

bool hasOver
Invoked with OVER.

bool hasPartition
Has partition.

bool hasSort
Invoked with SORT.

int inputColumnCount
The number of input columns.

int* inputIsConstant
Determines if the input type is a constant, 0 or 1.

int* inputScales

The scale of the numeric, otherwise 0.

int* inputSizes

The precision of the numeric or the max size of the string.

NzudsDataType* inputTypes

The input data types. NzudsDataType is defined elsewhere.

bool oneOutputRowRestriction

Row restriction; if TRUE, exactly one output row is required per input row and no output is allowed after the end of the data

int outputColumnCount

The number of output columns.

int* outputScales

The scale of the numeric, otherwise 0.

int* outputSizes

The precision of the numeric or the max size of the string.

NzudsDataType* outputTypes

The output data types. NzudsDataType is defined elsewhere.

NzaeNumeric128BytesBigEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric128)]

NzaeNumeric128BytesLittleEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric128)]

NzaeNumeric32BytesBigEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric32)]

NzaeNumeric32BytesLittleEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric32)]

NzaeNumeric64BytesBigEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric64)]

NzaeNumeric64BytesLittleEndian Struct Reference

Public Attributes

bytes

See Also

▲ Numeric Functions

Member Data Documentation

unsigned char bytes[sizeof(NzudsNumeric64)]

NZAEREMPROT_HANDLE Struct Reference

The Remote Protocol Handle. An opaque handle used with Remote Protocol AE functions.

Detailed Description

The Remote Protocol Handle. An opaque handle used with Remote Protocol AE functions.

See Also

▲ Remote Initialization.

NzaeRemprotCallbackResult Struct Reference

Public Attributes

bFreeData

data

dataLength

returnCode

Detailed Description

Setting a callback allows the Remote AE LDK Application to receive the following messages:

NZAE_REMPROT_CMD_STATUS NZAE_REMPROT_CMD_STOP NZAE_REMPROT_CMD_SIGNAL
NZAE_REM-PROT_CMD_CONTROL_DATA

See Also

▲ NzaeRemprotCallback

Member Data Documentation

int bFreeData

Must be TRUE if data has been allocated via malloc.

char* data

Data. Must be allocated via malloc.

int dataLength

Data length. May be 0.

int returnCode

Return Code. 0 is normal.

NzaeremprotInitialization Struct Reference

Initializes a Remote AE Notification Connection.

Public Attributes

errorMessage

handle

hConpt

ldkVersion

Detailed Description

Initializes a Remote AE Notification Connection.

See Also

▲ nzaeRemprotCreateListener

Member Data Documentation

char errorMessage[NZAEREMPROT_ERROR_MESSAGE_LENGTH]

NZAEREMPROT_HANDLE handle

NZAECONPT_HANDLE hConpt

int ldkVersion

NzaeRuntime Struct Reference

Runtime information.

Public Attributes

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

Detailed Description

Runtime information.

See Also

nzaeGetRuntime
 nzaeAggGetRuntime
 nzaeShpGetRuntime

Member Data Documentation

NzaeAdapterType adapterType

uint64_t aeCallId

uint64_t aeQueryId

int32_t dataSliceId

int32_t hardwareId

NzaeLocus locus

bool loggingEnabled

int logMask

int32_t numberDataSlices

int32_t numberSpus

int32_t sessionId

int64_t suggestedMemoryLimit

int64_t transactionId

char userName[1024]

bool userQuery

NzaeSharedLibraryInfo Struct Reference

Shared library information.

Public Attributes

autoLoad

An array of the autoloader settings.

libraryFullPaths

An array of the library's full paths.

libraryNames

An array of library names.

numLibraries

The number of libraries.

Detailed Description

Shared library information.

See Also

nzaeGetLibraryInfo
 nzaeAggGetLibraryInfo
 nzaeShpGetLibraryInfo

Member Data Documentation

bool* autoLoad
 An array of the autoloading settings.

const char** libraryFullPaths
 An array of the library's full paths.

const char** libraryNames
 An array of library names.

int numLibraries
 The number of libraries.

NZAESHP_HANDLE Struct Reference

The Shaper Handle. An opaque handle used with Shaper and Sizer AE functions.

Detailed Description

The Shaper Handle. An opaque handle used with Shaper and Sizer AE functions.

See Also

▲ [Shaper and Sizer](#)

NzaeShpInitialization Struct Reference

Argument to function nzaeShpInitialize. Output parameters are handle and errorMessage.

Public Attributes

errorMessage
 handle
 hEnv
 ldkVersion

Detailed Description

Argument to function nzaeShpInitialize. Output parameters are handle and errorMessage.

See Also

nzaeShpInitialize

Member Data Documentation

char errorMessage[NZAESHP_ERROR_MESSAGE_LENGTH]

NZAESHP_HANDLE handle

NZAEENV_HANDLE hEnv

int ldkVersion

NzaeShpMetadata Struct Reference

Metadata describing input rows of the AE. The memory pointed to by inputTypes belongs to the handle and should not be freed by the user.

Public Attributes

inputColumnCount

The number of input columns.

inputIsConstant

Determines if the input type is a constant, 0 or 1.

inputScales

The scale of the numeric, otherwise 0.

inputSizes

The precision of the numeric or the max size of string.

inputTypes

The number or input data types, NzudsDataType is defined elsewhere.

oneOutputRowRestriction

Row restriction; if TRUE exactly one output row is required per input row and no output is allowed after the end of the data.

Detailed Description

Metadata describing input rows of the AE. The memory pointed to by inputTypes belongs to the handle and should not be freed by the user.

See Also

▲ [nzaeShpGetMetadata](#)

Member Data Documentation

int inputColumnCount

The number of input columns.

int* inputIsConstant

Determines if the input type is a constant, 0 or 1.

int* inputScales

The scale of the numeric, otherwise 0.

int* inputSizes

The precision of the numeric or the max size of string.

NzudsDataType* inputTypes

The number or input data types, NzudsDataType is defined elsewhere.

bool oneOutputRowRestriction

Row restriction; if TRUE exactly one output row is required per input row and no output is allowed after the end of the data.

NzaeShpOutputColumnInfo Struct Reference

Information about a user-added output column.

Public Attributes

columnName

dataType

precision

scale

size

Detailed Description

Information about a user-added output column.

See Also

▲ [nzaeShpGetOutputColumnInfo](#)

Member Data Documentation

const char* columnName

NzudsDataType dataType

int precision

int scale

int size

NzudsData Struct Reference

Public Attributes

union { pBool

pDate pDouble

pFixedString

pFloat

pGeometryString

pInt16

pInt32

pInt64

pInt8 pInterval

pNationalFixedString

pNationalVariableString

pNumeric128

pNumeric32

pNumeric64

pTime

pTimeStamp

pTimeTz

pVarbinaryString

```

pVariableString
} data
isNull
length
union {
boolVal
dateVal
doubleVal
floatVal
int16Val
int32Val
int64Val
int8Val
interval
numeric128
numeric32
numeric64
timeStampVal
timeTz
timeVal
} privateData
type

```

Detailed Description

field data to serialize / deserialize

The memory that this struct points to belongs to the class.

The data in this struct is valid until it is used in another `nzudsReadNext`, any write operation is performed, or the handle is closed.

string types always have an extra null terminator not included in the length

See Also

▲ [Data Type Support](#).

Member Data Documentation

`int8_t boolVal`

`data`

union { ... }

int32_t dateVal

double doubleVal

float floatVal

int16_t int16Val

int32_t int32Val

int64_t int64Val

int8_t int8Val

NzudsInterval interval

int8_t isNull
if value is true then data column is SQL null

int32_t length
length of data

NzudsNumeric128 numeric128

NzudsNumeric32 numeric32

NzudsNumeric64 numeric64

const int8_t* pBool

const int32_t* pDate

const double* pDouble

const char* pFixedString

const float* pFloat

const char* pGeometryString

const int16_t* pInt16

const int32_t* pInt32

const int64_t* pInt64

const int8_t* pInt8

const NzudsInterval* pInterval

const char* pNationalFixedString

const char* pNationalVariableString

const NzudsNumeric128* pNumeric128

const NzudsNumeric32* pNumeric32

const NzudsNumeric64* pNumeric64

privateData

union { ... }

This union should be considered private

const int64_t* pTime

const int64_t* pTimeStamp

const NzudsTimeTz* pTimeTz

const char* pVarbinaryString

const char* pVariableString

int64_t timeStampVal

NzudsTimeTz timeTz

int64_t timeVal

NzudsDataType type

Data Type of this struct

Determines correct pointer in data union if data is not null.

▲ See Also

► NzudsDataType

NzudsInterval Struct Reference

Public Attributes

month

time

Detailed Description

Interval data type definition

It has microsecond resolution and ranges from +/- 178000000 years. The time part represents everything but months and years (microseconds) and the month part represents months and years.

See Also

▲ Data Type Support.

Member Data Documentation

int32_t month

int64_t time

NzudsNumeric128 Struct Reference

Public Attributes

digit

Detailed Description

Numeric 128 data type definition

See Also

▲ Data Type Support.

Member Data Documentation

NzudsNumericDigit digit[4]

NzudsNumeric32 Struct Reference

Public Attributes

digit

Detailed Description

Numeric 32 data type definition

See Also

▲ Data Type Support.

Member Data Documentation

NzudsNumericDigit digit[1]

NzudsNumeric64 Struct Reference

Public Attributes

digit

Detailed Description

Numeric 64 data type definition

See Also

▲ Data Type Support.

Member Data Documentation

NzudsNumericDigit digit[2]

NzudsTimeTz Struct Reference

Public Attributes

time
zone

Detailed Description

Time Zone data type definition

Uses the int64 time value and adds an int32 time zone as well. The time zone is represented in seconds.

See Also

▲ [Data Type Support](#).

Member Data Documentation

int64_t time

int32_t zone

Notices and Trademarks

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
26 Forest Street
Marlborough, MA 01752 U.S.A.*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement

or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only. This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. (enter the year or years). All rights reserved.

Trademarks

IBM, the IBM logo, ibm.com and Netezza are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trade-mark information" at ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

NEC is a registered trademark of NEC Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and/or other countries.

D-CC, D-C++, Diab+, FastJ, pSOS+, SingleStep, Tornado, VxWorks, Wind River, and the Wind River logo are trademarks, registered trademarks, or service marks of Wind River Systems, Inc. Tornado patent pending.

APC and the APC logo are trademarks or registered trademarks of American Power Conversion Corporation.

Other company, product or service names may be trademarks or service marks of others.



Regulatory and Compliance

Regulatory Notices

Install the NPS system in a restricted-access location. Ensure that only those trained to operate or service the equipment have physical access to it. Install each AC power outlet near the NPS rack that plugs into it, and keep it freely accessible. Provide approved 30A circuit breakers on all power sources.

Product may be powered by redundant power sources. Disconnect ALL power sources before servicing. High leakage current. Earth connection essential before connecting supply. Courant de fuite élevé. Raccordement à la terre indispensable avant le raccordement au réseau.

Homologation Statement

This product may not be certified in your country for connection by any means whatsoever to interfaces of public telecommunications networks. Further certification may be required by law prior to making any such connection. Contact an IBM representative or reseller for any questions.

FCC - Industry Canada Statement

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

CE Statement (Europe)

This product complies with the European Low Voltage Directive 73/23/EEC and EMC Directive 89/336/EEC as amended by European Directive 93/68/EEC.

Warning: This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

VCCI Statement

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づきクラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

Index

A

- adapterType
 - NzaeRuntime,87
- AE Manager Functionality,53
- aeCallId
 - NzaeRuntime,87
- aeQueryId
 - NzaeRuntime,87
- AeUserCode
 - User Codes,61
- Aggregate,35
 - nzaeAggClose,37
 - nzaeAggGetEnv,37
 - nzaeAggGetFirstEnvironmentEntry,38
 - nzaeAggGetLastErrorCode,38
 - nzaeAggGetLastErrorText,38
 - nzaeAggGetLibraryFullPath,38
 - nzaeAggGetLibraryInfo,39
 - nzaeAggGetLibraryProcessInfo,39
 - nzaeAggGetNextEnvironmentEntry,39
 - nzaeAggGetNumberOfParameters,40
 - nzaeAggGetParameter,40
 - nzaeAggGetRuntime,40
 - nzaeAggGetSystemLogFileName,40
 - nzaeAggGetType,40
 - nzaeAggLog,41
 - NzaeAggMessageType,42
 - nzaeAggNext,41
 - nzaeAggPing,41
 - NzaeAggRcCode,43
 - NzaeAggType,43
 - nzaeAggUpdate,42
 - nzaeAggUserError,42
- aggregation
 - NzaeApi,79
- any
 - NzaeApi,79
- apiType
 - NzaeApi,79
- autoLoad
 - NzaeSharedLibraryInfo,89

B

- bFreeData
 - NzaeRemprotCallbackResult,86
- boolVal
 - NzudsData,93
- bytes
 - NzaeNumeric128BytesBigEndian,83
 - NzaeNumeric128BytesLittleEndian,84
 - NzaeNumeric32BytesBigEndian,84
 - NzaeNumeric32BytesLittleEndian,84
 - NzaeNumeric64BytesBigEndian,85
 - NzaeNumeric64BytesLittleEndian,85

C

- columnName
 - NzaeShpOutputColumnInfo,92
- correlationType
 - NzaeMetadata,82

D

- data
 - NzaeRemprotCallbackResult,86
 - NzudsData,93
- Data Connection APIs,27
- Data Type Support.,62
 - NzudsDataType,63
 - NzudsNumericDigit,63
- dataLength
 - NzaeRemprotCallbackResult,86
- dataSliceld
 - NzaeRuntime,87
- dataType
 - NzaeShpOutputColumnInfo,92
- Date and Time Functions,53
 - nzaeIntervalToMilliseconds,54
 - nzaeIntervalToSeconds,55
 - nzaeMillisecondsToInterval,55
 - nzaeMillisecondsToNzTime,55
 - nzaeMinutesToNzTimeTzOffset,55
 - nzaeNzDateToPosixTimeMilliseconds,55
 - nzaeNzDateToPosixTimeSeconds,56
 - nzaeNzTimestampToPosixTimeMilliseconds,56

Index

- nzaeNzTimestampToPosixTimeSeconds,56
- nzaeNzTimeToMilliseconds,56
- nzaeNzTimeToSeconds,56
- nzaeNzTimeTzOffsetToMinutes,57
- nzaePosixTimeMillisecondsToNzDate,57
- nzaePosixTimeMillisecondsToNzTimestamp,57
- nzaePosixTimeSecondsToNzDate,57
- nzaePosixTimeSecondsToNzTimestamp,57
- nzaeSecondsToInterval,58
- nzaeSecondsToNzTime,58

- dateVal
 - NzudsData,94

- digit
 - NzudsNumeric128,97
 - NzudsNumeric32,97
 - NzudsNumeric64,97

- doubleVal
 - NzudsData,94

E

- errorMessage
 - NzaeAggInitialization,76
 - NzaeInitialization,81
 - NzaeremprotInitialization,86
 - NzaeShpInitialization,90

F

- floatVal
 - NzudsData,94
- function
 - NzaeApi,79
- Function,27
 - nzaeClose,29
 - NzaeCorrelationType,35
 - nzaeDone,29
 - nzaeGetEnv,29
 - nzaeGetFirstEnvironmentEntry,30
 - nzaeGetInputColumn,30
 - nzaeGetLastErrorCode,30
 - nzaeGetLastErrorText,31
 - nzaeGetLibraryFullPath,31
 - nzaeGetLibraryInfo,31
 - nzaeGetLibraryProcessInfo,31

- nzaeGetMetadata,32
- nzaeGetNext,32
- nzaeGetNextEnvironmentEntry,32
- nzaeGetNextPartition,32
- nzaeGetNumberOfParameters,33
- nzaeGetParameter,33
- nzaeGetRuntime,33
- nzaeLog,33
- nzaeOutputResult,34
- nzaePing,34
- NzaeRcCode,35
- nzaeUserError,34

G

- getValue
 - NzaeAggFieldFunctions,67
 - NzaeAggReadOnlyFieldFunctions,78

H

- handle
 - NzaeAggInitialization,76
 - NzaeApi,80
 - NzaeInitialization,81
 - NzaeremprotInitialization,86
 - NzaeShpInitialization,90
- hardwareId
 - NzaeRuntime,87
- hasOver
 - NzaeMetadata,82
- hasPartition
 - NzaeMetadata,82
- hasSort
 - NzaeMetadata,82
- hConpt
 - NzaeremprotInitialization,87
- hEnv
 - NzaeAggInitialization,76
 - NzaeInitialization,81
 - NzaeShpInitialization,90

I

- Initialization APIs,11

- NzaeApiTypes,12
- Initialize from an AE Environment.,13
 - nzaeAggInitialize,14
 - nzaeInitialize,14
 - nzaeShpInitialize,14
- input
 - NzaeAggAccumulate,66
- inputColumnCount
 - NzaeMetadata,82
 - NzaeShpMetadata,91
- inputIsConstant
 - NzaeMetadata,82
 - NzaeShpMetadata,91
- inputScales
 - NzaeMetadata,82
 - NzaeShpMetadata,91
- inputSizes
 - NzaeMetadata,83
 - NzaeShpMetadata,91
- inputState
 - NzaeAggFinalResult,75
 - NzaeAggMerge,77
- inputTypes
 - NzaeMetadata,83
 - NzaeShpMetadata,91
- int16Val
 - NzudsData,94
- int32Val
 - NzudsData,94
- int64Val
 - NzudsData,94
- int8Val
 - NzudsData,94
- interval
 - NzudsData,94
- isNull
 - NzaeAggFieldFunctions,68
 - NzaeAggReadOnlyFieldFunctions,78
 - NzudsData,94

L

- ldkVersion
 - NzaeAggInitialization,76
 - NzaeInitialization,81

- NzaeremprotInitialization,87
- NzaeShpInitialization,90
- length
 - NzudsData,94
- libraryFullPaths
 - NzaeSharedLibraryInfo,89
- libraryNames
 - NzaeSharedLibraryInfo,89
- Local Initialization,12
 - nzaelsLocal,13
 - nzaelsRemote,13
 - nzaeLocprotGetApi,13
- locus
 - NzaeRuntime,88
- loggingEnabled
 - NzaeRuntime,88
- logMask
 - NzaeRuntime,88

M

- metadata
 - NzaeAggFieldFunctions,75
 - NzaeAggReadOnlyFieldFunctions,79
- month
 - NzudsInterval,96

N

- name
 - NzaeEnvironmentEntry,81
- numberDataSlices
 - NzaeRuntime,88
- numberSpus
 - NzaeRuntime,88
- numColumns
 - NzaeAggMetadata,77
- Numeric Functions,58
 - nzaeGetDoubleFromNumeric128,59
 - nzaeGetDoubleFromNumeric32,59
 - nzaeGetDoubleFromNumeric64,59
- numeric128
 - NzudsData,94
- numeric32
 - NzudsData,94

Index

numeric64
 NzudsData,94
numLibraries
 NzaeSharedLibraryInfo,89
NZAЕ_HANDLE,65
NzaeAdapterType
 Runtime and Environment Information,60
NZAЕAGG_HANDLE,65
NzaeAggAccumulate,65
 input,66
 state,66
nzaeAggClose
 Aggregate,37
NzaeAggFieldFunctions,66
 getValue,67
 isNull,68
 metadata,75
 setBool,68
 setDate,68
 setDouble,69
 setFloat,69
 setInt16,69
 setInt32,70
 setInt64,70
 setInt8,70
 setInterval,71
 setNull,71
 setNumeric128,71
 setNumeric32,72
 setNumeric64,72
 setString,72
 setStringLength,73
 setTime,73
 setTimeStamp,74
 setTimeTz,74
 setValue,74
NzaeAggFinalResult,75
 inputState,75
 result,75
nzaeAggGetEnv
 Aggregate,37
nzaeAggGetFirstEnvironmentEntry
 Aggregate,38
nzaeAggGetLastErrorCode
 Aggregate,38
nzaeAggGetLastErrorText
 Aggregate,38
nzaeAggGetLibraryFullPath
 Aggregate,38
nzaeAggGetLibraryInfo
 Aggregate,39
nzaeAggGetLibraryProcessInfo
 Aggregate,39
nzaeAggGetNextEnvironmentEntry
 Aggregate,39
nzaeAggGetNumberOfParameters
 Aggregate,40
nzaeAggGetParameter
 Aggregate,40
nzaeAggGetRuntime
 Aggregate,40
nzaeAggGetSystemLogFileName
 Aggregate,40
nzaeAggGetType
 Aggregate,40
NzaeAggInitialization,75
 errorMessage,76
 handle,76
 hEnv,76
 ldkVersion,76
nzaeAggInitialize
 Initialize from an AE Environment.,14
NzaeAggInitializeState,76
 state,76
nzaeAggLog
 Aggregate,41
NzaeAggMerge,76
 inputState,77
 state,77
NzaeAggMessageType
 Aggregate,42
NzaeAggMetadata,77
 numColumns,77
 scales,77
 sizes,77
 types,77
nzaeAggNext
 Aggregate,41

- nzaeAggPing
 - Aggregate,41
- NzaeAggRcCode
 - Aggregate,43
- NzaeAggReadOnlyFieldFunctions,77
 - getValue,78
 - isNull,78
 - metadata,79
- NzaeAggType
 - Aggregate,43
- nzaeAggUpdate
 - Aggregate,42
- nzaeAggUserError
 - Aggregate,42
- NzaeApi,79
 - aggregation,79
 - any,79
 - apiType,79
 - function,79
 - handle,80
 - shaper,80
- NzaeApiTypes
 - Initialization APIs,12
- nzaeClose
 - Function,29
- NZAECONPT_HANDLE,80
- nzaeconptBuildFileName
 - Remote Connection Point.,16
- nzaeconptClose
 - Remote Connection Point.,16
- nzaeconptCreate
 - Remote Connection Point.,16
- nzaeconptGetDataSliceId
 - Remote Connection Point.,16
- nzaeconptGetName
 - Remote Connection Point.,16
- nzaeconptGetSessionId
 - Remote Connection Point.,17
- nzaeconptGetTransactionId
 - Remote Connection Point.,17
- nzaeconptGetType
 - Remote Connection Point.,17
- nzaeconptSetDataSliceId
 - Remote Connection Point.,17
- nzaeconptSetName
 - Remote Connection Point.,17
- nzaeconptSetSessionId
 - Remote Connection Point.,18
- nzaeconptSetTransactionId
 - Remote Connection Point.,18
- nzaeconptSetType
 - Remote Connection Point.,18
- NzaeConptType
 - Remote Connection Point.,18
- NzaeCorrelationType
 - Function,35
- nzaeDone
 - Function,29
- NZAEENV_HANDLE,80
- NzaeEnvironmentEntry,80
 - name,81
 - value,81
- nzaeGetDoubleFromNumeric128
 - Numeric Functions,59
- nzaeGetDoubleFromNumeric32
 - Numeric Functions,59
- nzaeGetDoubleFromNumeric64
 - Numeric Functions,59
- nzaeGetEnv
 - Function,29
- nzaeGetFirstEnvironmentEntry
 - Function,30
- nzaeGetInputColumn
 - Function,30
- nzaeGetLastErrorCode
 - Function,30
- nzaeGetLastErrorText
 - Function,31
- nzaeGetLibraryFullPath
 - Function,31
- nzaeGetLibraryInfo
 - Function,31
- nzaeGetLibraryProcessInfo
 - Function,31
- nzaeGetMetadata
 - Function,32
- nzaeGetNext
 - Function,32

Index

- nzaeGetNextEnvironmentEntry
 - Function,32
- nzaeGetNextPartition
 - Function,32
- nzaeGetNumberOfParameters
 - Function,33
- nzaeGetParameter
 - Function,33
- nzaeGetRuntime
 - Function,33
- NzaeInitialization,81
 - errorMessage,81
 - handle,81
 - hEnv,81
 - ldkVersion,81
- nzaeInitialize
 - Initialize from an AE Environment.,14
- nzaeIntervalToMilliseconds
 - Date and Time Functions,54
- nzaeIntervalToSeconds
 - Date and Time Functions,55
- nzaelsLocal
 - Local Initialization,13
- nzaelsRemote
 - Local Initialization,13
- nzaeLocprotGetApi
 - Local Initialization,13
- NzaeLocus
 - Runtime and Environment Information,60
- nzaeLog
 - Function,33
- NzaeLogLevel
 - Runtime and Environment Information,61
- NzaeMetadata,81
 - correlationType,82
 - hasOver,82
 - hasPartition,82
 - hasSort,82
 - inputColumnCount,82
 - inputIsConstant,82
 - inputScales,82
 - inputSizes,83
 - inputTypes,83
 - oneOutputRowRestriction,83
 - outputColumnCount,83
 - outputScales,83
 - outputSizes,83
 - outputTypes,83
- nzaeMillisecondsToInterval
 - Date and Time Functions,55
- nzaeMillisecondsToNzTime
 - Date and Time Functions,55
- nzaeMinutesToNzTimeTzOffset
 - Date and Time Functions,55
- NzaeNumeric128BytesBigEndian,83
 - bytes,83
- NzaeNumeric128BytesLittleEndian,83
 - bytes,84
- NzaeNumeric32BytesBigEndian,84
 - bytes,84
- NzaeNumeric32BytesLittleEndian,84
 - bytes,84
- NzaeNumeric64BytesBigEndian,84
 - bytes,85
- NzaeNumeric64BytesLittleEndian,85
 - bytes,85
- nzaeNzDateToPosixTimeMilliseconds
 - Date and Time Functions,55
- nzaeNzDateToPosixTimeSeconds
 - Date and Time Functions,56
- nzaeNzTimestampToPosixTimeMilliseconds
 - Date and Time Functions,56
- nzaeNzTimestampToPosixTimeSeconds
 - Date and Time Functions,56
- nzaeNzTimeToMilliseconds
 - Date and Time Functions,56
- nzaeNzTimeToSeconds
 - Date and Time Functions,56
- nzaeNzTimeTzOffsetToMinutes
 - Date and Time Functions,57
- nzaeOutputResult
 - Function,34
- nzaePing
 - Function,34
- nzaePosixTimeMillisecondsToNzDate
 - Date and Time Functions,57
- nzaePosixTimeMillisecondsToNzTimestamp
 - Date and Time Functions,57

- nzaePosixTimeSecondsToNzDate
 - Date and Time Functions,57
- nzaePosixTimeSecondsToNzTimestamp
 - Date and Time Functions,57
- NzaeRcCode
 - Function,35
- NZAEREMPROT_HANDLE,85
- nzaeRemprotAcceptApi
 - Remote Initialization.,21
- nzaeRemprotAcceptApiWithTimeout
 - Remote Initialization.,21
- nzaeRemprotAcceptEnvironment
 - Remote Initialization.,22
- nzaeRemprotAcceptEnvironmentWithTimeout
 - Remote Initialization.,22
- NzaeRemprotCallback
 - Remote Initialization.,21
- NzaeRemprotCallbackResult,85
 - bFreeData,86
 - data,86
 - dataLength,86
 - returnCode,86
- nzaeRemprotClose
 - Remote Initialization.,23
- NzaeRemprotCmd
 - Remote Initialization.,26
- nzaeRemprotCreateListener
 - Remote Initialization.,23
- nzaeRemprotFreeResources
 - Remote Initialization.,23
- nzaeRemprotGetAcceptSocket
 - Remote Initialization.,23
- nzaeRemprotGetCallback
 - Remote Initialization.,24
- nzaeRemprotGetEnvironmentApiType
 - Remote Initialization.,24
- nzaeRemprotGetLastErrorText
 - Remote Initialization.,24
- nzaeRemprotGetRemoteDataSliceld
 - Remote Initialization.,24
- nzaeRemprotGetRemoteName
 - Remote Initialization.,24
- nzaeRemprotGetRemoteSessionId
 - Remote Initialization.,25
- nzaeRemprotGetRemoteTransactionId
 - Remote Initialization.,25
- NzaeremprotInitialization,86
 - errorMessage,86
 - handle,86
 - hConpt,87
 - ldkVersion,87
- nzaeRemprotIsError
 - Remote Initialization.,25
- NzaeRemprotRcCode
 - Remote Initialization.,27
- nzaeRemprotSetCallback
 - Remote Initialization.,25
- nzaeRemprotWaitForPingOrStop
 - Remote Initialization.,25
- NzaeRuntime,87
 - adapterType,87
 - aeCallId,87
 - aeQueryId,87
 - dataSliceld,87
 - hardwareId,87
 - locus,88
 - loggingEnabled,88
 - logMask,88
 - numberDataSlices,88
 - numberSpus,88
 - sessionId,88
 - suggestedMemoryLimit,88
 - transactionId,88
 - userName,88
 - userQuery,88
- nzaeSecondsToInterval
 - Date and Time Functions,58
- nzaeSecondsToNzTime
 - Date and Time Functions,58
- NzaeSharedLibraryInfo,88
 - autoLoad,89
 - libraryFullPaths,89
 - libraryNames,89
 - numLibraries,89
- NZAESHP_HANDLE,89
- nzaeShpAddOutputColumn
 - Shaper and Sizer,45
- nzaeShpAddOutputColumnNumeric

Index

- Shaper and Sizer,45
- nzaeShpAddOutputColumnString
 - Shaper and Sizer,46
- nzaeShpClose
 - Shaper and Sizer,46
- nzaeShpGetEnv
 - Shaper and Sizer,46
- nzaeShpGetFirstEnvironmentEntry
 - Shaper and Sizer,47
- nzaeShpGetInputColumn
 - Shaper and Sizer,47
- nzaeShpGetLastErrorCode
 - Shaper and Sizer,47
- nzaeShpGetLastErrorText
 - Shaper and Sizer,47
- nzaeShpGetLibraryFullPath
 - Shaper and Sizer,48
- nzaeShpGetLibraryInfo
 - Shaper and Sizer,48
- nzaeShpGetLibraryProcessInfo
 - Shaper and Sizer,48
- nzaeShpGetMetadata
 - Shaper and Sizer,49
- nzaeShpGetNextEnvironmentEntry
 - Shaper and Sizer,49
- nzaeShpGetNumberOfParameters
 - Shaper and Sizer,49
- nzaeShpGetNumOutputColumns
 - Shaper and Sizer,49
- nzaeShpGetOutputColumnInfo
 - Shaper and Sizer,50
- nzaeShpGetParameter
 - Shaper and Sizer,50
- nzaeShpGetRuntime
 - Shaper and Sizer,50
- nzaeShpGetSystemLogFileName
 - Shaper and Sizer,51
- nzaeShpGetUdfReturnType
 - Shaper and Sizer,51
- NzaeShpInitialization,89
 - errorMessage,90
 - handle,90
 - hEnv,90
 - ldkVersion,90
- nzaeShpInitialize
 - Initialize from an AE Environment.,14
- nzaeShpLog
 - Shaper and Sizer,51
- NzaeShpMetadata,90
 - inputColumnCount,91
 - inputIsConstant,91
 - inputScales,91
 - inputSizes,91
 - inputTypes,91
 - oneOutputRowRestriction,91
- NzaeShpOutputColumnInfo,91
 - columnName,92
 - dataType,92
 - precision,92
 - scale,92
 - size,92
- nzaeShpPing
 - Shaper and Sizer,51
- NzaeShpRcCode
 - Shaper and Sizer,52
- nzaeShpSystemCatalogIsUpper
 - Shaper and Sizer,52
- nzaeShpUpdate
 - Shaper and Sizer,52
- nzaeShpUserError
 - Shaper and Sizer,52
- nzaeUserError
 - Function,34
- NzudsData,92
 - boolVal,93
 - data,93
 - dateVal,94
 - doubleVal,94
 - floatVal,94
 - int16Val,94
 - int32Val,94
 - int64Val,94
 - int8Val,94
 - interval,94
 - isNull,94
 - length,94
 - numeric128,94
 - numeric32,94

- numeric64,94
- pBool,94
- pDate,94
- pDouble,94
- pFixedString,94
- pFloat,95
- pGeometryString,95
- pInt16,95
- pInt32,95
- pInt64,95
- pInt8,95
- pInterval,95
- pNationalFixedString,95
- pNationalVariableString,95
- pNumeric128,95
- pNumeric32,95
- pNumeric64,95
- privateData,95
- pTime,95
- pTimeStamp,95
- pTimeTz,95
- pVarbinaryString,95
- pVariableString,95
- timestampVal,95
- timeTz,96
- timeVal,96
- type,96
- NzudsDataType
 - Data Type Support.,63
- NzudsInterval,96
 - month,96
 - time,96
- NzudsNumeric128,96
 - digit,97
- NzudsNumeric32,97
 - digit,97
- NzudsNumeric64,97
 - digit,97
- NzudsNumericDigit
 - Data Type Support.,63
- NzudsTimeTz,97
 - time,98
 - zone,98

O

- oneOutputRowRestriction
 - NzaeMetadata,83
 - NzaeShpMetadata,91
- outputColumnCount
 - NzaeMetadata,83
- outputScales
 - NzaeMetadata,83
- outputSizes
 - NzaeMetadata,83
- outputTypes
 - NzaeMetadata,83

P

- pBool
 - NzudsData,94
- pDate
 - NzudsData,94
- pDouble
 - NzudsData,94
- pFixedString
 - NzudsData,94
- pFloat
 - NzudsData,95
- pGeometryString
 - NzudsData,95
- pInt16
 - NzudsData,95
- pInt32
 - NzudsData,95
- pInt64
 - NzudsData,95
- pInt8
 - NzudsData,95
- pInterval
 - NzudsData,95
- pNationalFixedString
 - NzudsData,95
- pNationalVariableString
 - NzudsData,95
- pNumeric128
 - NzudsData,95
- pNumeric32

Index

- NzudsData,95
- pNumeric64
 - NzudsData,95
- precision
 - NzaeShpOutputColumnInfo,92
- privateData
 - NzudsData,95
- pTime
 - NzudsData,95
- pTimeStamp
 - NzudsData,95
- pTimeTz
 - NzudsData,95
- pVarbinaryString
 - NzudsData,95
- pVariableString
 - NzudsData,95

R

- Remote Connection Point.,15
 - nzaeconptBuildFileName,16
 - nzaeconptClose,16
 - nzaeconptCreate,16
 - nzaeconptGetDataSlicId,16
 - nzaeconptGetName,16
 - nzaeconptGetSessionId,17
 - nzaeconptGetTransactionId,17
 - nzaeconptGetType,17
 - nzaeconptSetDataSlicId,17
 - nzaeconptSetName,17
 - nzaeconptSetSessionId,18
 - nzaeconptSetTransactionId,18
 - nzaeconptSetType,18
 - NzaeConptType,18
- Remote Initialization.,19
 - nzaeRemprotAcceptApi,21
 - nzaeRemprotAcceptApiWithTimeout,21
 - nzaeRemprotAcceptEnvironment,22
 - nzaeRemprotAcceptEnvironmentWithTimeout,22
 - NzaeRemprotCallback,21
 - nzaeRemprotClose,23
 - NzaeRemprotCmd,26
 - nzaeRemprotCreateListener,23
 - nzaeRemprotFreeResources,23

- nzaeRemprotGetAcceptSocket,23
- nzaeRemprotGetCallback,24
- nzaeRemprotGetEnvironmentApiType,24
- nzaeRemprotGetLastErrorText,24
- nzaeRemprotGetRemoteDataSlicId,24
- nzaeRemprotGetRemoteName,24
- nzaeRemprotGetRemoteSessionId,25
- nzaeRemprotGetRemoteTransactionId,25
- nzaeRemprotIsError,25
- NzaeRemprotRcCode,27
- nzaeRemprotSetCallback,25
- nzaeRemprotWaitForPingOrStop,25
- result
 - NzaeAggFinalResult,75
- returnCode
 - NzaeRemprotCallbackResult,86
- Runtime and Environment Information,59
 - NzaeAdapterType,60
 - NzaeLocus,60
 - NzaeLogLevel,61

S

- scale
 - NzaeShpOutputColumnInfo,92
- scales
 - NzaeAggMetadata,77
- sessionId
 - NzaeRuntime,88
- setBool
 - NzaeAggFieldFunctions,68
- setDate
 - NzaeAggFieldFunctions,68
- setDouble
 - NzaeAggFieldFunctions,69
- setFloat
 - NzaeAggFieldFunctions,69
- setInt16
 - NzaeAggFieldFunctions,69
- setInt32
 - NzaeAggFieldFunctions,70
- setInt64
 - NzaeAggFieldFunctions,70
- setInt8
 - NzaeAggFieldFunctions,70

- setInterval
 - NzaeAggFieldFunctions,71
- setNull
 - NzaeAggFieldFunctions,71
- setNumeric128
 - NzaeAggFieldFunctions,71
- setNumeric32
 - NzaeAggFieldFunctions,72
- setNumeric64
 - NzaeAggFieldFunctions,72
- setString
 - NzaeAggFieldFunctions,72
- setStringLength
 - NzaeAggFieldFunctions,73
- setTime
 - NzaeAggFieldFunctions,73
- setTimeStamp
 - NzaeAggFieldFunctions,74
- setTimeTz
 - NzaeAggFieldFunctions,74
- setValue
 - NzaeAggFieldFunctions,74
- shaper
 - NzaeApi,80
- Shaper and Sizer,43
 - nzaeShpAddOutputColumn,45
 - nzaeShpAddOutputColumnNumeric,45
 - nzaeShpAddOutputColumnString,46
 - nzaeShpClose,46
 - nzaeShpGetEnv,46
 - nzaeShpGetFirstEnvironmentEntry,47
 - nzaeShpGetInputColumn,47
 - nzaeShpGetLastErrorCode,47
 - nzaeShpGetLastErrorText,47
 - nzaeShpGetLibraryFullPath,48
 - nzaeShpGetLibraryInfo,48
 - nzaeShpGetLibraryProcessInfo,48
 - nzaeShpGetMetadata,49
 - nzaeShpGetNextEnvironmentEntry,49
 - nzaeShpGetNumberOfParameters,49
 - nzaeShpGetNumOutputColumns,49
 - nzaeShpGetOutputColumnInfo,50
 - nzaeShpGetParameter,50
 - nzaeShpGetRuntime,50

- nzaeShpGetSystemLogFileName,51
- nzaeShpGetUdfReturnType,51
- nzaeShpLog,51
- nzaeShpPing,51
- NzaeShpRcCode,52
- nzaeShpSystemCatalogIsUpper,52
- nzaeShpUpdate,52
- nzaeShpUserError,52
- size
 - NzaeShpOutputColumnInfo,92
- sizes
 - NzaeAggMetadata,77
- state
 - NzaeAggAccumulate,66
 - NzaeAggInitializeState,76
 - NzaeAggMerge,77
- suggestedMemoryLimit
 - NzaeRuntime,88
- Support APIs,53

T

- time
 - NzudsInterval,96
 - NzudsTimeTz,98
- timeStampVal
 - NzudsData,95
- timeTz
 - NzudsData,96
- timeVal
 - NzudsData,96
- transactionId
 - NzaeRuntime,88
- type
 - NzudsData,96
- types
 - NzaeAggMetadata,77

U

- User Codes,61
 - AeUserCode,61
- userName
 - NzaeRuntime,88
- userQuery

Index

NzaeRuntime,88

V

value

NzaeEnvironmentEntry,81

Z

zone

NzudsTimeTz,98