IBM® Netezza® Analytics
Release 11.x

Analytic Executables Language Development Kit API Reference

Part Number 00X6333-00 Rev. 1

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

Contents

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 com-ponents:

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

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

Comments on the Documentation

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

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

We appreciate your comments.

CHAPTER 1
Module Documentation

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 nzaeIsLocal 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 nzaeLoc-protGetApi. If an AE is local, function nzaeIsLocal returns a TRUE value. If an AE is not local it is re-mote.

Functions

int nzaeIsLocal()
Returns TRUE if the AE is local.

int nzaeIsRemote()
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 nzaeLoc-protGetApi. If an AE is local, function nzaeIsLocal returns a TRUE value. If an AE is not local it is re-mote.

Function Documentation

int nzaeIsLocal()
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 nzaeIsRemote()
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 rrror 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 nzaeAggIntialize. Output parameters are handle and errorMessage.

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

struct NzaeShpInitialization
Argument to function nzaeShpIntialize. 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* nzaeconptBuildFileTypeName(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* nzaeconptBuildFileTypeName(NZAECONPT_HANDLE handle)
Builds and returns the file type connection point fully qualified name for file format connec-tion 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, tranaction 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_CON-TROL_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 (NZAE_REMPROT_CMD_STATUS, NZAE_REMPROT_CMD_STOP, NZAE_REMPROT_CMD_SIGNAL, NZAE_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 millseconds.

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 initalization 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 ocurred.

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 re-ceived.

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
caseSens-itive)
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.

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

Enumerations

enum NzaeCorrelationType {
NzaeUnknownCorrelationType= 0, NzaeUncorrelated= 1, NzaeInnerCorrelation= 2, NzaeLeftCorrelation= 3 }

Specialized information about how this AE is being invoked.

enum NzaeRcCode {
NZAE_RC_ERROR= -1, NZAE_RC_NORMAL= 0, NZAE_RC_END=

1 } Return codes from nzae functions.

Detailed Description

Function AEs are called from Scalar or Table SQL Functions.

Function Documentation

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

Parameters

handle
The function handle.

NzaeRcCode nzaeDone(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
NzaeRcCode

The function return code.

NzaeRcCode nzaeGetEnv(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 sys-tem 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 retured 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, NzaeEnvironmen-tEntry *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 sys-tem 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 point-er. 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 data-Type, 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 nzaeShpGetNextEnvironmentEntry(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, Nza-eShpOutputColumnInfo *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_AD-APTER_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_AL-LOWED= 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_CATA-LOG_IS_UPPER= 18, AE_UC_INVALID_SIZER_GET_UDF_RETURN_TYPE= 19, AE_UC_BAD_NEXT_ENVIR-ONMENT= 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)(NZAEAGG_HANDLE handle, int index, NzudsData **data) Get Value.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NzaeAggRcCode(* setValue)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_HANDLE handle, int index, bool *result) Specifieds 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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_HANDLE handle, int index, const NzudsNu-meric32 *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)(NZAEAGG_HANDLE handle, int index, const NzudsNu-meric64 *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)(NZAEAGG_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)(NZAEAGG_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 ter-mination does not apply.

NzaeAggRcCode(* setTime)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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 nzaeAggIntialize. Output parameters are handle and errorMessage.

Public Attributes

errorMessage
handle
hEnv
ldkVersion

Detailed Description

An argument to function nzaeAggIntialize. 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)(NZAEAGG_HANDLE handle, int index, NzudsData **data)

Get Value.

NzaeAggRcCode(* isNull)(NZAEAGG_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)(NZAEAGG_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)(NZAEAGG_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 follwing 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 autoload 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 autoload 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 nzaeShpIntialize. Output parameters are handle and errorMessage.

Public Attributes

errorMessage
handle
hEnv
ldkVersion

Detailed Description

Argument to function nzaeShpIntialize. 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 al-lowed 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 in 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

Trademarks

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 ser-vice the equipment have physical access to it. Install each AC power outlet near the NPS rack that plugs into it, and keep it freely accessible. Provide approved 30A circuit breakers on all power sources.

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

Homologation Statement

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

FCC - Industry Canada Statement

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

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

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

CE Statement (Europe)

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

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

VCCI Statement

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

Index

Index