Technical Report NEESgrid-2004-03

www.neesgrid.org

(Whitepaper Version: 0.4)

Last modified: February 18, 2004

NEES NTCP Control Plugin Documentation

Joseph Bester[1]

Feedback on this document should be directed to bester@mcs.anl.gov

[1] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

Acknowledgment: This work was supported primarily by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Number CMS-0117853.

NEESgrid NTCP Control Plugin

October 17, 2003

Contents

1 NEESgrid NTCP Control Plugin Main Page

NEESgrid NTCP C Plugin Interface

This document describes the C language interface for the NTCP Control Plugin described in the `Control Plugin Description (DRAFT)`. The audience of this document are people who are interested in implementing native c-language control plugins for the NTCP server. This document describes the general requirements of a plugin, the functions which must be implemented by a plugin, and the API available for the plugin to interact with the NEES server.

Plugin Requirements

Plugins will be implemented as a shared object or DLL which must be thread-safe, using the same threading model as the JVM. This software has been tested on the following platforms:

- Linux with glibc 2.1 and 2.3, with a 2.4 kernel, with Sun's JVM version 1.4 and pthreads.
- Windows XP with Sun's JVM version 1.4.2 with windows threads, linked with the multithreaded runtime dll

Plugins must export a function called *initialize_plugin*, of type nees_ntcp_plugin_init_t. This will be called by the NTCP server when a new plugin instance is requested. The plugin implementation must return pointers to functions which implement the rest of the plugin interface as well as an optional `void *` argument which will be passed to each of those, which may be used to point to plugin instance-specific data.

All plugin function types return an integer value: zero for success and non-zero for failure. When a non-zero value is returned, the plugin driver will call the plugin's nees_ntcp_plugin_error_string_t function to translate the error to a string which will be added to the exception which is thrown to the NTCP server. A more detailed description of the plugin function types is available in the plugin functions page.

Plugin Data Type APIs

This interface also defines a number of abstract data types which correspond to the Java classes used by the NTCP server implementation to pass values from the client to the plugin implementation. These are described in the data types page.

Configuring a C Plugin

WSDD Properties The NTCP server is configured via a WSDD file in its installation directory. This file contains (among other things) a sequence of service defintions. In the definition of the *nees/ntcp/NTCPServer* service, the ntcpBackendFactory and the org.nees.ntcp.plugins.gateway.module parameters must be defined to use the NTCP C Gateway Plugin and the native C plugin code.

Parameter Name: ntcpBackendFactory Parameter Value: org.nees.ntcp.plugins.gateway.GatewayPluginFactory

Parameter Name: org.nees.ntcp.plugins.gateway.module Parameter Value:Native Shared Object Name

The shared object name should either contain the absolute path of the object or DLL, or just the base name of an object or DLL in the runtime-linker's search path. For linux, this object must be in a directory contained in the *LD_LIBRARY_PATH* environment variable. For windows the DLL must be in a directory contained in the *PATH* environment variable. If the shared object cannot be found, an error will be generated on linux, but on windows, the NTCP server will fail in native code.

The following is an example WSDD fragment which will configure an NTCP server to use the native plugin located in the shared library "neesgrid_c_plugin.so", with the changes from the default NTCP Server WSDD highlighted.

```
<service name="nees/ntcp/NTCPServer" provider="Handler" style="wrapped">
     <parameter name="name" value="NTCP Server"/>
     <parameter name="schemaPath"
                value="schema/nees/ntcp/ntcp_serve\_service.wsdl"/>
     <parameter name="className"
                value="org.nees.ntcp.ntcpServer.NtcpServer"/>
     <parameter name="baseClassName"
                value="org.nees.ntcp.server.impl.NtcpServerImpl"/>
     <parameter name="operationProviders"
                value="org.globus.ogsa.impl.ogsi.NotificationSourceProvider"/>
     <parameter name="handlerClass"
                value="org.globus.ogsa.handlers.RPCURIProvider"/>
     <parameter name="persistent" value="true"/>
     <parameter name="allowedMethods" value="*"/>
     <parameter name="isSecure" value="true"/>
     <parameter name="isProfiling" value="false"/>
     <parameter name="localPolicyPlugin"
         value="org.nees.ntcp.server.backend.test.DummyLocalPolicyPlugin"/>
     <parameter name="ntcpBackendFactory"
                value="org.nees.ntcp.plugins.gateway.GatewayPluginFactory"/>
     <parameter name="org.nees.ntcp.plugins.gateway.module"
                value="neesgrid_c_plugin.so"/>
 </service>
```

System Properties By default, the Java gateway plugin will load the C gateway plugin code from libnees_ntcp_control_plugin_gcc32dbgpthr.so on linux and nees_ntcp_control_plugin_win32dbgthr.dll on windows. The shared object must be found in one of the directories in LD_LIBRARY_PATH or PATH environment variables in linux and windows respectively. If the plugin is installed with a different name or Globus flavor, then the Java System properties org.nees.ntcp.plugins.gateway.lib and org.nees.ntcp.plugins.gateway.flavor can be used to let the Java Gateway Plugin know what shared object or DLL to load.

org.nees.ntcp.plugins.gateway.lib The base name of the DLL, not containing the prefix ”lib” on linux, the extension ”.so” or ”.dll” on linux or windows respectively, and the flavor name. The default for this is ”nees_ntcp_control_plugin”.

org.nees.ntcp.plugins.gateway.flavor The Globus flavor specifier, which will be appended with a leading underscore to the gateway lib name before the extension is added by the loader.

For example, to use the gateway plugin library ”libnees_ntcp_control_plugin_vendorcc32pthr.so” instead of ”libnees_ntcp_control_plugin_gcc32dbgpthr.so” on linux, the property org.nees.ntcp.plugins.gateway.flavor should be set to ”vendorcc32pthr” and the org.nees.ntcp.plugins.gateway.flavor property can be omitted. System properties are typically set by passing `-D`*`property-name`*`=`**property-value** on the Java command line, so this example would be to add `-Dorg.nees.ntcp.plugins.gateway.flavor=vendorcc32pthr` to the Java command line.

2 NEESgrid NTCP Control Plugin Module Index

2.1 NEESgrid NTCP Control Plugin Modules

Here is a list of all modules:

3 NEESgrid NTCP Control Plugin Module Documentation

3.1 NEES NTCP Types

Modules

- Transaction Handle
- Parameter
- Control Point
- Control Point Geometry Parameter

Typedefs

- typedef nees_i_ntcp_transaction_handle_s * nees_ntcp_transaction_handle_t
- typedef nees_i_ntcp_parameter_s * nees_ntcp_parameter_t
- typedef nees_i_ntcp_control_point_s * nees_ntcp_control_point_t
- typedef nees_i_ntcp_geom_parameter_s * nees_ntcp_geom_parameter_t

Enumerations

- enum nees_ntcp_geom_axis_t {
  NEES_NTCP_AXIS_NONE,
  NEES_NTCP_AXIS_X,
  NEES_NTCP_AXIS_Y,
  NEES_NTCP_AXIS_Z }
- enum nees_ntcp_geom_parameter_name_t {
  NEES_NTCP_PARAMETER_NONE,
  NEES_NTCP_PARAMETER_FORCE,
  NEES_NTCP_PARAMETER_MOMENT,
  NEES_NTCP_PARAMETER_DISPLACEMENT,
  NEES_NTCP_PARAMETER_ROTATION }

3.1.1 Typedef Documentation

3.1.1.1 typedef struct nees_i_ntcp_transaction_handle_s* nees_ntcp_transaction_handle_t

Abstract type analog to the java ControlPluginTransactionHandle class.
See the transaction handle documentation for functions related to this type.

3.1.1.2 typedef struct nees_i_ntcp_parameter_s* nees_ntcp_parameter_t

Abstract type analog to the java ParameterType class.
See the parameter documentation for functions related to this type.

3.1.1.3 typedef struct nees_i_ntcp_control_point_s* nees_ntcp_control_point_t

Abstract type analog to the java ControlPointType class.
See the control point documentation for functions related to this type.

3.1.1.4 typedef struct nees_i_ntcp_geom_parameter_s* nees_ntcp_geom_parameter_t

Abstract type analog to the java ControlPointGeomParameterType class.

3.1.2 Enumeration Type Documentation

3.1.2.1 enum nees_ntcp_geom_axis_t

Enumeration of axes used in a geometry parameter.

Enumeration values:

NEES_NTCP_AXIS_NONE An uninitialized axis.

NEES_NTCP_AXIS_X The x axis.

NEES_NTCP_AXIS_Y The y axis.

NEES_NTCP_AXIS_Z The z axis.

3.1.2.2 enum nees_ntcp_geom_parameter_name_t

Enumeration of parameter names.

Enumeration values:

NEES_NTCP_PARAMETER_NONE An uninitialized parameter.

NEES_NTCP_PARAMETER_FORCE A force parameter.

NEES_NTCP_PARAMETER_MOMENT A moment parameter.

NEES_NTCP_PARAMETER_DISPLACEMENT A displacement parameter.

NEES_NTCP_PARAMETER_ROTATION A rotation parameter.

3.2 Required Plugin Functions

Functions signatures for plugin functions.

Typedefs

- typedef int(∗ nees_ntcp_plugin_propose_t)(void ∗arg, const nees_ntcp_transaction_handle_t transaction, globus_bool_t ∗result)
- typedef int(∗ nees_ntcp_plugin_execute_t)(void ∗arg, nees_ntcp_transaction_handle_t transaction)
- typedef int(∗ nees_ntcp_plugin_cancel_t)(void ∗arg, const nees_ntcp_transaction_handle_t transaction)
- typedef int(∗ nees_ntcp_plugin_get_control_point_t)(void ∗arg, const char ∗name, nees_ntcp_control_point_t ∗result_point)
- typedef int(∗ nees_ntcp_plugin_set_parameter_t)(void ∗arg, const nees_ntcp_parameter_t parameter)
- typedef int(∗ nees_ntcp_plugin_get_parameter_t)(void ∗arg, const char ∗name, nees_ntcp_parameter_t result_parameter)
- typedef int(∗ nees_ntcp_plugin_open_session_t)(void ∗arg, const nees_ntcp_parameter_t ∗parameter_array, size_t parameter_array_size)
- typedef int(∗ nees_ntcp_plugin_close_session_t)(void ∗arg)
- typedef int(∗ nees_ntcp_plugin_error_string_t)(void ∗arg, int error_code, char ∗∗string)
- typedef void(∗ nees_ntcp_plugin_destroy_t)(void ∗arg)
- typedef int(∗ nees_ntcp_plugin_init_t)(void ∗∗arg, nees_ntcp_plugin_propose_t ∗propose, nees_ntcp_plugin_execute_t ∗execute, nees_ntcp_plugin_cancel_t ∗cancel, nees_ntcp_plugin_get_control_point_t ∗get_control_point, nees_ntcp_plugin_set_parameter_t ∗set_parameter, nees_ntcp_plugin_get_parameter_t ∗get_parameter, nees_ntcp_plugin_open_session_t ∗open_session, nees_ntcp_plugin_close_session_t ∗close_session, nees_ntcp_plugin_error_string_t ∗error_string, nees_ntcp_plugin_destroy_t ∗destroy)

3.2.1 Detailed Description

Functions signatures for plugin functions.

3.2.2 Typedef Documentation

3.2.2.1 typedef int(* nees_ntcp_plugin_propose_t)(void * arg, const nees_ntcp_transaction_handle_t transaction, globus_bool_t * result)

The plugin does site-specific validation of a transaction (such as verifying that all the control points that appear in the request actually exist, or that the control point parameters specified in the transaction request do not exceed any site-specific maximum values).

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

transaction A transaction handle containing a transaction name and any control point parameters associated with this proposed transaction.

result A pointer to a boolean return value. If this function doesn't encounter an exception while validating the transaction, it must set the boolean's value to either

GLOBUS_TRUE All site-specific conditions for accepting the proposed transaction are met.

GLOBUS_FALSE Any site-specific condition for accepting the proposed transaction was not met.

Returns:

If any error occurs while attempting to validate the proposed transaction, then this function must return a non-zero value. In that case, the NTCP server will ignore the value of the boolean pointed to by the *result* parameter and an exception will be thrown.

3.2.2.2 typedef int(* nees_ntcp_plugin_execute_t)(void * arg, nees_ntcp_transaction_handle_t transaction)

The plugin begins asynchronously processing the proposed transaction. If this transaction completes successfully, the plugin must call nees_ntcp_transaction_handle_set_resulting_control_points() with the resulting control point status. If this transaction completes unsuccessfully, then the plugin must call nees_ntcp_transaction_handle_mark_terminated().

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

transaction A transaction handle containing a transaction name and any control point parameters associated with this transaction. If the plugin instance is able to begin executing the transaction successfully, then it may retain a reference to this transaction handle until the transaction is completed, either by the plugin calling nees_ntcp_transaction_handle_set_resulting_control_points(), nees_ntcp_transaction_handle_mark_terminated(), or via a successful return from the plugin's cancel function. After any of those conditions are met, the plugin must no longer access its reference to the transaction handle.

Returns:

If any error occurs while attempting to validate or begin executing the proposed transaction, then this function must return a non-zero value.

3.2.2.3 typedef int(* nees_ntcp_plugin_cancel_t)(void * arg, const nees_ntcp_transaction_handle_t transaction)

The plugin attempts to cancel transaction described by the transaction handle. This function blocks until the specified transaction is either completely cancelled or something exceptional prevents the cancellation from being processed.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

transaction A transaction handle containing a transaction name and any control point parameters associated with this transaction.

Returns:

If the plugin is able to cancel the transaction, it must return zero from this function; otherwise, it must return a non-zero value.

3.2.2.4 typedef int(* nees_ntcp_plugin_get_control_point_t)(void * arg, const char * name, nees_ntcp_control_point_t * result_point)

The plugin returns the values of the named control point.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

name The name of the control point whose value is being requested.

result_point A pointer to a control point which the plugin must initialize and set to a valid control point if the named point is known by the plugin; otherwise, this pointer must be set to NULL. The plugin must not access or free the result point after this function returns.

Returns:

If the plugin is able to set the value of the *result_point* point successfully, it must return zero; otherwise, it must return a non-zero value.

3.2.2.5 typedef int(* nees_ntcp_plugin_set_parameter_t)(void * arg, const nees_ntcp_parameter_t parameter)

The plugin processes a change in the value of a parameter. The plugin does any required site-specific validation and action based on the new value of the parameter.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

parameter The new value of the parameter. The plugin must not access the parameter outside of this function scope. If the plugin wants to retain a copy of this value, it must make its own copy using nees_ntcp_parameter_copy().

Returns:

If the new parameter value is accepted by the plugin, the plugin must return zero; otherwise, it must return a non-zero value.

3.2.2.6 typedef int(* nees_ntcp_plugin_get_parameter_t)(void * arg, const char * name, nees_ntcp_parameter_t result_parameter)

The plugin processes a request to find the current value of the requested parameter.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

name The name of the parameter whose value is being requested.

result_parameter A parameter which the plugin should modify with new geometry parameter values. The plugin must not access the result_parameter value outside of this function scope. If the plugin wants to retain a copy of this value, it must make its own copy using nees_ntcp_parameter_copy().

Returns:

If the new parameter value is accepted by the plugin, the plugin must return zero; otherwise, it must return a non-zero value.

3.2.2.7 typedef int(* nees_ntcp_plugin_open_session_t)(void * arg, const nees_ntcp_parameter_t * parameter_array, size_t parameter_array_size)

The plugin prepares to run an experiment by setting initial state as necessary.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

parameter_array An array of parameters. Which the plugin may use to initialize the experiment. If the plugin wishes to access any of these parameters after this function returns, it must call nees_ntcp_control_plugin_parameter_copy() to create a copy of that parameter. Any copies made by the plugin must be freed by the plugin.

parameter_array_size The number of entries in the parameter array.

Returns:

If the session state is initialized successfully, the plugin must return zero; otherwise, it must return a non-zero value.

3.2.2.8 typedef int(* nees_ntcp_plugin_close_session_t)(void * arg)

The plugin prepares to end an experiment by doing any site-specific shutdown procedure.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called.

Returns:

If the session state is shut down successfully, the plugin must return zero; otherwise, it must return a non-zero value.

3.2.2.9 typedef int(* nees_ntcp_plugin_error_string_t)(void * arg, int error_code, char ** string)

The plugin converts an error code returned from one of its functions to an error string.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called. If the *error_code* value was generated by the initialize_plugin function, then this parameter will be *NULL*.

error_code A non-zero value returned previously from one of the plugin functions.

string A pointer to a `char *` which the plugin will allocate and populate with an error description string. If this is set to a NULL value, a default unhelpful string will be provided by the plugin interface. The string will be freed by the plugin interface.

Returns:

If the plugin generates a string successfully then this function must return zero; otherwise, it must return a non-zero value and the *string* value will be ignored (and not freed by the plugin interface).

3.2.2.10 typedef void(* nees_ntcp_plugin_destroy_t)(void * arg)

The plugin destroys any state associated with the plugin instance. This will only be called after all transactions executing in the plugin have been successfully terminated or cancelled. The plugin interface will no longer make any calls into the plugin until the initialize_plugin is called again.

Parameters:

arg The plugin instance-specific data value returned when the plugin's initialize_plugin function was called. This value may be freed by the plugin, if appropriate, as the plugin interface will no longer use it.

Returns:

void

3.2.2.11 typedef int(* nees_ntcp_plugin_init_t)(void ** arg, nees_ntcp_plugin_propose_t * propose, nees_ntcp_plugin_execute_t * execute, nees_ntcp_plugin_cancel_t * cancel, nees_ntcp_plugin_get_control_point_t * get_control_point, nees_ntcp_plugin_set_parameter_t * set_parameter, nees_ntcp_plugin_get_parameter_t * get_parameter, nees_ntcp_plugin_open_session_t * open_session, nees_ntcp_plugin_close_session_t * close_session, nees_ntcp_plugin_error_string_t * error_string, nees_ntcp_plugin_destroy_t * destroy)

The plugin implementation initializes a new plugin instance by returning a pointer to a plugin instance-specific value and pointers to function pointers used to implement the plugin.

Parameters:

arg A pointer to a `void *` value which the plugin may use to point to a plugin instance-specific data structure. This value will be passed to all of the plugin function pointers.

propose A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to propose a new transaction to the plugin.

execute A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin begin executing a transaction.

cancel A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin cancel a transaction.

get_control_point A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin return the current value of a control point.

set_parameter A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin set the value of a named parameter.

get_parameter A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin get the value of a named parameter.

open_session A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin do any necessary local initialization.

close_session A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin do any necessary local deactivation.

error_string A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to have the plugin translate an integer error code to an error string.

destroy A pointer to a function pointer. This value must be modified by the plugin to point to a function which the plugin interface will call to destroy this plugin instance.

Returns:

If the plugin is able to initialize the plugin instance, it must return zero from this function; otherwise, it must return a non-zero value.

3.3 Transaction Handle

Functions

- int nees_ntcp_transaction_handle_get_name (const nees_ntcp_transaction_handle_t handle, char **result_name)
- int nees_ntcp_transaction_handle_get_requested_control_points (const nees_ntcp_transaction_handle_t handle, nees_ntcp_control_point_t **result_points, size_t *result_number_of_points)
- int nees_ntcp_transaction_handle_mark_terminated (nees_ntcp_transaction_handle_t handle)
- int nees_ntcp_transaction_handle_set_resulting_control_points (nees_ntcp_transaction_handle_t handle, nees_ntcp_control_point_t *control_points, size_t control_points_length)

3.3.1 Detailed Description

This section describes the functions which may be executed to access the values of a transaction handle and to indicate completion of a transaction.

3.3.2 Function Documentation

3.3.2.1 int nees_ntcp_transaction_handle_get_name (const nees_ntcp_transaction_handle_t *handle*, char ** *result_name*)

Get the name of a transaction.

Parameters:

handle A transaction handle which the plugin is interested in the name of. This handle must be one obtained by the plugin as passed to its propose or execute functions, subject to the scoping rules described in those function pointer descriptions.

result_name A pointer to a `char *`. This function will modify the value pointed to by this to be a copy of the transaction's name. The plugin instance must free this copy of the name using free().

Returns:

If the plugin is valid and the name pointer is valid, then this function returns zero; otherwise, a non-zero error code is returned and *result_name* will be unmodified.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The name was copied successfully.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the transaction's name into result_name

3.3.2.2 int nees_ntcp_transaction_handle_get_requested_control_points (const nees_ntcp_transaction_handle_t *handle*, nees_ntcp_control_point_t ** *result_points*, size_t * *result_number_of_points*)

Get the value of the transaction's control points.

Parameters:

handle A transaction handle which the plugin is interested in the name of. This handle must be one obtained by the plugin as passed to its propose or execute functions, subject to the scoping rules described in those function pointer descriptions.

result_points A pointer to a `nees_ntcp_control_point_t` array. This function will modify the value pointed to by this to point to a copy of the transaction's control points. The plugin must free this array by calling nees_ntcp_control_point_array_free()

result_number_of_points A pointer to a `size_t`. This function will modify the value pointed to by this to be the number of elements in the new *points* array.

Returns:

If the parameters to this function are valid and it is able to copy the control points, then this function returns zero; otherwise, it returns a non-zero error code.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control points were copied successfully.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the transaction's control points into *result_points*.

3.3.2.3 int nees_ntcp_transaction_handle_mark_terminated (nees_ntcp_transaction_handle_t *handle*)

Indicate that transaction execution finished unsuccessfully.

Parameters:

handle A transaction handle which the plugin is terminating. This handle must be one obtained by the plugin as passed to its execute function. After this function returns, the plugin must not access the handle.

Returns:

If the handle is valid and the transaction state was set as terminated, then this function returns zero; otherwise a non-zero error code is returned.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The transaction termination was completed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the transaction's control points into *result_points*.

NEES_NTCP_CONTROL_PLUGIN_JVM_ERROR Internal error accessing the Java Virtual Machine.

NEES_NTCP_CONTROL_PLUGIN_CLASS_FORMAT_ERROR,Internal error loading a Java class. Java class file corrupted.

NEES_NTCP_CONTROL_PLUGIN_CLASS_CIRCULARITY_ERROR Internal error loading a Java class. A Java class file contains a broken object implementation.

NEES_NTCP_CONTROL_PLUGIN_CLASS_NOT_FOUND Internal error loading a Java class. Java CLASSPATH not set properly.

NEES_NTCP_CONTROL_PLUGIN_EXCEPTION_IN_INITIALIZER Internal erorr loading a Java class.

NEES_NTCP_CONTROL_PLUGIN_NO_SUCH_METHOD Internal error locating a Java method. Likely caused by a Java interface change.

NEES_NTCP_CONTROL_PLUGIN_NO_SUCH_FIELD Internal error locating a Java field. Likely caused by a Java interface change.

NEES_NTCP_CONTROL_PLUGIN_JAVA_EXCEPTION A Java method called while handling this threw some other exception.

3.3.2.4 int nees_ntcp_transaction_handle_set_resulting_control_points (nees_ntcp_transaction_handle_t *handle*, nees_ntcp_control_point_t * *control_points*, size_t *control_points_length*)

Indicate that transaction execution finished successfully, setting the resulting value of control points of the transaction.

Parameters:

handle A transaction handle which the plugin is terminating. This handle must be one obtained by the plugin as passed to its execute function. After this function returns, the plugin must not access the handle.

control_points An array of control points which the plugin has allocated by calling nees_ntcp_control_point_array_alloc(). The values of the control points are the results of the transaction. The individual points in the array should represent measured or computed values of the control points at the time that execution completed. The plugin is responsible for freeing this array.

control_points_length The number of control points in the *control_points* array.

Returns:

If the parameters to this function are valid and the transaction state was set as terminated, then this function returns zero; otherwise a non-zero error code is returned.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The transaction termination was completed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the transaction's control points into *result_points*.

NEES_NTCP_CONTROL_PLUGIN_JVM_ERROR Internal error accessing the Java Virtual Machine.

NEES_NTCP_CONTROL_PLUGIN_CLASS_FORMAT_ERROR,Internal error loading a Java class. Java class file corrupted.

NEES_NTCP_CONTROL_PLUGIN_CLASS_CIRCULARITY_ERROR Internal error loading a Java class. A Java class file contains a broken object implementation.

NEES_NTCP_CONTROL_PLUGIN_CLASS_NOT_FOUND Internal error loading a Java class. Java CLASSPATH not set properly.

NEES_NTCP_CONTROL_PLUGIN_EXCEPTION_IN_INITIALIZER Internal erorr loading a Java class.

NEES_NTCP_CONTROL_PLUGIN_NO_SUCH_METHOD Internal error locating a Java method. Likely caused by a Java interface change.

NEES_NTCP_CONTROL_PLUGIN_NO_SUCH_FIELD Internal error locating a Java field. Likely caused by a Java interface change.

NEES_NTCP_CONTROL_PLUGIN_JAVA_EXCEPTION A Java method called while handling this threw some other exception.

3.4 Parameter

Functions

- int nees_ntcp_parameter_init (nees_ntcp_parameter_t *parameter)
- int nees_ntcp_parameter_copy (const nees_ntcp_parameter_t original, nees_ntcp_parameter_t *copy)
- int nees_ntcp_parameter_destroy (nees_ntcp_parameter_t parameter)
- int nees_ntcp_parameter_get_name (const nees_ntcp_parameter_t parameter, char **return_name)
- int nees_ntcp_parameter_get_value (const nees_ntcp_parameter_t parameter, char **return_value)
- int nees_ntcp_parameter_set_name (nees_ntcp_parameter_t parameter, const char *name)
- int nees_ntcp_parameter_set_value (nees_ntcp_parameter_t parameter, const char *value)

3.4.1 Detailed Description

This section describes the functions which may be executed to access the values of a NTCP parameter. Parameters are used to express experimental parameters as name-value pairs.

3.4.2 Function Documentation

3.4.2.1 int nees_ntcp_parameter_init (nees_ntcp_parameter_t * *parameter*)

Construct a new parameter.

Parameters:

parameter A pointer to a `nees_ntcp_parameter_t`. This function will allocated a new parameter and modify the value pointed to by this to point to the new parameter. The plugin must destroy this parameter when no longer needed by calling nees_ntcp_parameter_destroy().

Returns:

This function returns 0 if the parameter was allocated successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter was successfully initialized.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to initialize the parameter.

3.4.2.2 int nees_ntcp_parameter_copy (const nees_ntcp_parameter_t *original*, nees_ntcp_parameter_t * *copy*)

Copy a parameter.

Parameters:

original The original parameter to copy.

copy A pointer to a `nees_ntcp_parameter_t`. This function will allocated a new parameter and modify the value pointed to by this to point to the new parameter. The name and value of the copy will match that of the *original* parameter. The plugin must destroy this parameter when no longer needed by calling nees_ntcp_parameter_destroy().

Returns:

This function returns 0 if the arguments to this function are valid and the parameter was copied successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter was successfully copied.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the parameter into *copy*.

3.4.2.3 int nees_ntcp_parameter_destroy (nees_ntcp_parameter_t *parameter*)

Destroy a parameter.

Parameters:

parameter The parameter to destroy. After this function returns, the plugin must no longer access this parameter. A plugin should only pass parameters to this function which it had created by calling nees_ntcp_parameter_init() or nees_ntcp_parameter_copy().

Returns:

This function returns 0 if the arguments to this function are valid and the parameter was destroyed successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter was successfully destroyed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *parameter* passed to this function was NULL.

3.4.2.4 int nees_ntcp_parameter_get_name (const nees_ntcp_parameter_t *parameter*, char ** *return_name*)

Get a parameter's name.

Parameters:

parameter The parameter to query.

return_name A pointer to a `char *`. This function will modify the value of this to point to a newly allocated string containing the parameter name. The plugin is responsible for freeing this parameter by calling free() on this value.

Returns:

This function returns 0 if the arguments to it are valid and the parameter has a name; it returns a non-zero value otherwise. When an error value is returned, the value pointed to by return_name is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter name was successfully copied.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the parameter's name into *return_name*.

3.4.2.5 int nees_ntcp_parameter_get_value (const nees_ntcp_parameter_t *parameter*, char ** *return_value*)

Get a parameter's value.

Parameters:

parameter The parameter to query.

return_value A pointer to a `char *`. This function will modify the value of this to point to a newly allocated string containing the parameter value. The plugin is responsible for freeing this parameter by calling free() on this value.

Returns:

This function returns 0 if the arguments to it are valid and the parameter has a value; it returns a non-zero value otherwise. When an error value is returned, the value pointed to by return_value is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter value was successfully copied.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the parameter's value into *return_value*.

3.4.2.6 int nees_ntcp_parameter_set_name (nees_ntcp_parameter_t *parameter*, const char * *name*)

Set a parameter's name.

Parameters:

parameter The parameter to query.

name The new value of the name. This name string will be copied into the parameter structure. The plugin is responsible for any deallocation needed for the value it passes to this function.

Returns:

This function returns 0 if the arguments to it are valid and the parameter name is modified; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter's name was successfully set.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to set the parameter's name.

3.4.2.7 int nees_ntcp_parameter_set_value (nees_ntcp_parameter_t *parameter*, const char * *value*)

Set a parameter's value.

Parameters:

parameter The parameter to query.

value The new value of the parameter. This value string will be copied into the parameter structure. The plugin is responsible for any deallocation needed for the value it passes to this function.

Returns:

This function returns 0 if the arguments to it are valid and the parameter value is modified; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The parameter's value was successfully set.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to set the parameter's value.

3.5 Control Point

Functions

- int nees_ntcp_control_point_init (nees_ntcp_control_point_t *control_point)
- int nees_ntcp_control_point_copy (const nees_ntcp_control_point_t original, nees_ntcp_control_point_t *copy)
- int nees_ntcp_control_point_destroy (nees_ntcp_control_point_t control_point)

- int nees_ntcp_control_point_array_alloc (size_t array_size, nees_ntcp_control_point_t **control_points)
- int nees_ntcp_control_point_array_free (nees_ntcp_control_point_t *control_points, size_t array_size)
- int nees_ntcp_control_point_get_name (const nees_ntcp_control_point_t control_point, char **result_name)
- int nees_ntcp_control_point_get_geometry_parameter_count (const nees_ntcp_control_point_t control_point, size_t *result_count)
- int nees_ntcp_control_point_get_geometry_parameter (const nees_ntcp_control_point_t control_point, size_t i, nees_ntcp_geom_parameter_t *result_geom_parameter)
- int nees_ntcp_control_point_set_name (nees_ntcp_control_point_t control_point, const char *name)
- int nees_ntcp_control_point_set_geometry_parameter (nees_ntcp_control_point_t control_point, size_t i, const nees_ntcp_geom_parameter_t geom_parameter)

3.5.1 Detailed Description

This section describes the functions which may be executed to access the values of a NTCP control point. Control points are used to name a set of force, movement, displacement, and rotations along an axis.

3.5.2 Function Documentation

3.5.2.1 int nees_ntcp_control_point_init (nees_ntcp_control_point_t * *control_point*)

Construct a new control point.

Parameters:

control_point A pointer to a `nees_ntcp_control_point_t`. This function will allocated a new control point and modify the value pointed to by this to point to the new control point. The plugin must destroy this control point when it is no longer needed by calling nees_ntcp_control_point_destroy().

Returns:

This function returns 0 if the control point was allocated successfully; otherwise, it returns a non-zero value and the value of *control_point* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point was successfully initialized.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *control_point* parameter to this function was NULL.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to initialize the control point.

3.5.2.2 int nees_ntcp_control_point_copy (const nees_ntcp_control_point_t *original*, nees_ntcp_control_point_t * *copy*)

Copy a control_point.

Parameters:

original The original control_point to copy.

copy A pointer to a `nees_ntcp_control_point_t`. This function will allocated a new control_point and modify the value pointed to by this to point to the new control point. The name and geometry parameters associated with this control point will be modified to match that of the *original* control point. The plugin must destroy this control_point when no longer needed by calling nees_ntcp_control_point_destroy().

Returns:

This function returns 0 if the arguments to this function are valid and the control point was copied successfully; otherwise, it returns a non-zero value and the value of *copy* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point was successfully copied to *copy*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was NULL.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the control point.

3.5.2.3 int nees_ntcp_control_point_destroy (nees_ntcp_control_point_t *control_point*)

Destroy a control point.

Parameters:

control_point A control point to destroy. After this function returns, the plugin must no longer access this control point. A plugin should only pass control points to this function which it had created by calling nees_ntcp_control_point_init() or nees_ntcp_control_point_copy().

Returns:

This function returns 0 if the arguments to this function are valid and the control point was destroyed successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point was successfully destroyed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *control_point* passed to this function was NULL.

3.5.2.4 int nees_ntcp_control_point_array_alloc (size_t *array_size*, nees_ntcp_control_point_t ** *control_points*)

Alocate an array of control points.

Parameters:

array_size The number of control points to be allocated in the *control_points* array.

control_points A pointer to an array of control_points. This function will modify the value pointed to by this pointer to a newly allocated array of length *array_size* control points. Each control point in the array will be initialized. The plugin is responsible for freeing this array by calling nees_ntcp_control_point_array_free(). The

Returns:

This function returns 0 if the arguments to this function are valid and *control_points* array could be allocated successfully; otherwise it returns a non-zero value and the value of *control_points* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point array was successfully allocated.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters passed to this function was NULL.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to allocate the control point array.

3.5.2.5 int nees_ntcp_control_point_array_free (nees_ntcp_control_point_t * *control_points*, size_t *array_size*)

Free an array of control points.

Parameters:

control_points A pointer to an array of control points to free. This function will free each individual control point in this array as well as the memory associated with the array.

array_size The size of the *control_points* array.

Returns:

This function returns 0 if the arguments to this function are valid and it is able to free the control point array; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point array was successfully freed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *control_points* array passed to this function was NULL.

3.5.2.6 int nees_ntcp_control_point_get_name (const nees_ntcp_control_point_t *control_point*, char ** *result_name*)

Get the name of a control point.

Parameters:

control_point A control point to query.

result_name A pointer to a `char *`. This function will modify the value pointed to by this to point to a copy of the name of the control point. The plugin is responsible for freeing this name by calling free() on the value.

Returns:

This function returns 0 if the arguments to this function are valid and it is able to return a copy of the control point's name; it returns a non-zero value otherwise, and the value pointed to by *result_name* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point name was successfully copied into *result_name*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the control points's name into *return_name*.

3.5.2.7 int nees_ntcp_control_point_get_geometry_parameter_count (const nees_ntcp_control_point_t *control_point*, size_t * *result_count*)

Get the number of number of geometry parameters associated with a control point.

Parameters:

control_point The control point to query.

result_count A pointer to a `size_t`. The value pointed to by this will be modified to contain the number of geometry parameters associated with this control point.

Returns:

This function returns 0 if the arguments to this function are valid and it is able to return the count of geometry parameters associated with a control point; it returns a non-zero value otherwise, and the value pointed to by *result_count* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The control point parameter count was successfully copied into *result_count*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

3.5.2.8 int nees_ntcp_control_point_get_geometry_parameter (const nees_ntcp_control_point_t *control_point*, size_t *i*, nees_ntcp_geom_parameter_t * *result_geom_parameter*)

Get one geometry parameter associated with a control point.

Parameters:

control_point The control point to query.

i The index of the geometry parameter to retrieve.

result_geom_parameter A pointer to a `nees_ntcp_geom_parameter_t`. This function will modify the value pointed to by this to point to a copy of the geometry parameter in the slot indicated by the *i* parameter. The plugin must free this geometry parameter by calling nees_ntcp_control_point_geom_parameter_free() when done accessing this parameter.

Returns:

This function returns 0 if the arguments to it are valid and it is able to generate a copy of the geometry parameter; otherwise, it returns a non-zero value and the value pointed to by *result_geom_parameter* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was successfully copied into *result_geom_parameter*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null or *i* is outside the acceptable range of geometry parameters for this control point.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the geometry parameter value into *return_geom_parameter*.

3.5.2.9 int nees_ntcp_control_point_set_name (nees_ntcp_control_point_t *control_point*, const char * *name*)

Set the name of a control point.

Parameters:

control_point The control point to modify.

name The new value of the control point's name. This name string will be copied into the control point structure. The plugin is responsible for any deallocation needed for the value it passes to this function.

Returns:

This function returns 0 if the arguments to it are valid and it is able to set the name of the control point; otherwise, it returns a non-zero value and the *control_point* is unmodified..

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The name of the control point was set successfully.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the name into control point.

3.5.2.10 int nees_ntcp_control_point_set_geometry_parameter (nees_ntcp_control_point_t *control_point*, size_t *i*, const nees_ntcp_geom_parameter_t *geom_parameter*)

Set the value of one geometry parameter in a control point.

Parameters:

control_point The control point to modify.

i The index of the geometry parameter to set. Valid values are in the range 0 to 11, inclusive.

geom_parameter The new value of the geometry parameter. The value of this parameter will be copied into the *i*th parameter in the control point. The caller is responsable for freeing *geom_parameter* after this function returns.

Returns:

This function returns 0 if the arguments to it are valid and it is able to set the indicated parameter; otherwise, it returns a non-zero value and the *control_point* is unmodified.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The *i*th geometry parameter of the control point was set successfully.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null or *i* is outside the acceptable range of geometry parameters for this control point.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the geometry parameter into control point.

3.6 Control Point Geometry Parameter

Functions

- int nees_ntcp_geom_parameter_init (nees_ntcp_geom_parameter_t ∗geom_parameter)
- int nees_ntcp_geom_parameter_copy (const nees_ntcp_geom_parameter_t original, nees_ntcp_geom_parameter_t ∗copy)
- int nees_ntcp_geom_parameter_destroy (nees_ntcp_geom_parameter_t geom_parameter)
- int nees_ntcp_geom_parameter_get_axis (const nees_ntcp_geom_parameter_t geom_parameter, nees_ntcp_geom_axis_t ∗result_axis)
- int nees_ntcp_geom_parameter_get_parameter_name (nees_ntcp_geom_parameter_t geom_parameter, nees_ntcp_geom_parameter_name_t ∗result_name)
- int nees_ntcp_geom_parameter_get_value (nees_ntcp_geom_parameter_t geom_parameter, double ∗result_value)
- int nees_ntcp_geom_parameter_set_all (nees_ntcp_geom_parameter_t geom_parameter, nees_ntcp_geom_axis_t axis, nees_ntcp_geom_parameter_name_t name, double value)
- int nees_ntcp_geom_parameter_set_axis (nees_ntcp_geom_parameter_t geom_parameter, nees_ntcp_geom_axis_t axis)
- int nees_ntcp_geom_parameter_set_parameter_name (nees_ntcp_geom_parameter_t geom_parameter, nees_ntcp_geom_parameter_name_t parameter_name)
- int nees_ntcp_geom_parameter_set_value (nees_ntcp_geom_parameter_t geom_parameter, double value)

3.6.1 Detailed Description

This section describes the functions which may be executed to access the values of a NTCP geometry parameter associated with a control point.

3.6.2 Function Documentation

3.6.2.1 int nees_ntcp_geom_parameter_init (nees_ntcp_geom_parameter_t * *geom_parameter*)

Construct a new geometry parameter for a control point.

Parameters:

geom_parameter A pointer to a `nees_ntcp_geom_parameter_t`. This function will allocated a new geometry parameter and modify the value pointed to by this to point to the new geometry parameter. The plugin must destroy this geometry parameter when it is no longer needed by calling nees_ntcp_control_point_geom_parameter_destroy().

Returns:

This function returns 0 if the geometry parameter was allocated successfully; otherwise, it returns a non-zero value and the value of *control_point* is undefined.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was successfully initialized.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_parameter* passed to this function was NULL.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to initialize the geometry parameter.

3.6.2.2 int nees_ntcp_geom_parameter_copy (const nees_ntcp_geom_parameter_t *original*, nees_ntcp_geom_parameter_t * *copy*)

Copy a geometry parameter.

Parameters:

original The original geometry parameter to copy.

copy A pointer to a `nees_ntcp_geom_parameter_t`. This function will allocated a new geometry parameter and modify the value pointed to by this to point to the new parameter. The name and values of the copy will match that of the *original* parameter. The plugin must destroy this geometry parameter when no longer needed by calling nees_ntcp_control_point_geom_parameter_destroy().

Returns:

This function returns 0 if the arguments to this function are valid and the parameter was copied successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was successfully copied to *copy*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters to this function was null.

NEES_NTCP_CONTROL_PLUGIN_OUT_OF_MEMORY There is not enough free memory to copy the geometry parameter.

3.6.2.3 int nees_ntcp_geom_parameter_destroy (nees_ntcp_geom_parameter_t *geom_parameter*)

Destroy a geometry parameter.

Parameters:

geom_parameter The geometry parameter to destroy. After this function returns, the plugin must no longer access this geometry parameter. A plugin should only pass geometry parameters to this function which it had created by calling nees_ntcp_geom_parameter_init(), nees_ntcp_geom_parameter_copy(), or nees_ntcp_control_point_get_geometry_parameter().

Returns:

This function returns 0 if the arguments to this function are valid and the geometry parameter was destroyed successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was successfully destroyed.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_parameter* passed to this function was null.

3.6.2.4 int nees_ntcp_geom_parameter_get_axis (const nees_ntcp_geom_parameter_t *geom_parameter*, nees_ntcp_geom_axis_t * *result_axis*)

Get the axis of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to query.

result_axis A pointer to a `nees_ntcp_geom_axis_t`. This function will modify the value pointed to by this geometry parameter to contain the current axis of the geometry parameter passed as the *geom_parameter* argument.

Returns:

This function returns 0 if the arguments to this function are valid and the *result_axis* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter's axis was successfully copied to *result_axis*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters passed to this function was null.

See also:

nees_ntcp_geom_axis_t

3.6.2.5 int nees_ntcp_geom_parameter_get_parameter_name (nees_ntcp_geom_parameter_t *geom_parameter*, nees_ntcp_geom_parameter_name_t * *result_name*)

Get the name of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to query.

result_name A pointer to a `nees_ntcp_geom_parameter_name_t`. This function will modify the value pointed to by this geometry parameter to contain the current parameter name of the geometry parameter passed as the *geom_parameter* argument.

Returns:

This function returns 0 if the arguments to this function are valid and the *result_parameter_name* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter's name was successfully copied to *result_name*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters passed to this function was null.

See also:

nees_ntcp_geom_parameter_name_t

3.6.2.6 int nees_ntcp_geom_parameter_get_value (nees_ntcp_geom_parameter_t *geom_parameter*, double * *result_value*)

Get the value of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to query.

result_value A pointer to a `double`. This function will modify the value pointed to by this geometry parameter to contain the current value of the geometry parameter passed as the *geom_parameter* argument.

Returns:

This function returns 0 if the arguments to this function are valid and the *result_value* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter's value was successfully copied to *result_value*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER One of the parameters passed to this function was null.

3.6.2.7 int nees_ntcp_geom_parameter_set_all (nees_ntcp_geom_parameter_t *geom_parameter*, nees_ntcp_geom_axis_t *axis*, nees_ntcp_geom_parameter_name_t *name*, double *value*)

Set all data values of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to modify.

axis The new value of the axis. If this is NEES_NTCP_AXIS_NONE then the value of the axis in the geometry parameter will not be modified.

name The new value of the axis. If this is NEES_NTCP_PARAMETER_NONE then the value of the axis in the geometry parameter will not be modified.

value The new value of the geometry parameter.

Returns:

This function returns 0 if the arguments to this function are valid and the *geom_parameter* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was modified to contain the passed values.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_paramter* passed to this function was null, or one of the *axis* or *name* values contains an illegal value.

3.6.2.8 int nees_ntcp_geom_parameter_set_axis (nees_ntcp_geom_parameter_t *geom_parameter*, nees_ntcp_geom_axis_t *axis*)

Set the axis of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to modify.

axis The new value of the axis. If this is NEES_NTCP_AXIS_NONE then the value of the axis in the geometry parameter will not be modified.

Returns:

This function returns 0 if the arguments to this function are valid and the *geom_parameter* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was modified to have the new value of *axis*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_paramter* passed to this function was null, or the *axis* value is invalid.

3.6.2.9 int nees_ntcp_geom_parameter_set_parameter_name (nees_ntcp_geom_parameter_t *geom_parameter*, nees_ntcp_geom_parameter_name_t *parameter_name*)

Set the parameter name of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to modify.

parameter_name The new value of the axis. If this is NEES_NTCP_PARAMETER_NONE then the value of the axis in the geometry parameter will not be modified.

Returns:

This function returns 0 if the arguments to this function are valid and the *geom_parameter* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was modified to have the new value of *parameter_name*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_paramter* passed to this function was null, or the *parameter_name* value is invalid.

3.6.2.10 int nees_ntcp_geom_parameter_set_value (nees_ntcp_geom_parameter_t *geom_parameter*, double *value*)

Set the value of a geometry parameter.

Parameters:

geom_parameter The geometry parameter to modify.

value The new value of the geometry parameter.

Returns:

This function returns 0 if the arguments to this function are valid and the *geom_parameter* was modified successfully; it returns a non-zero value otherwise.

Return values:

NEES_NTCP_CONTROL_PLUGIN_OK The geometry parameter was modified to have the new value of *value*.

NEES_NTCP_CONTROL_PLUGIN_INVALID_PARAMETER The *geom_paramter* passed to this function was null.

Index