Building the National Virtual Collaboratory for Earthquake Engineering Research

NEESgrid

Technical Report NEESgrid-2004-25
www.neesgrid.org

Last modified: June 4, 2004

NEESgrid NTCP C Client API

Sam Lang[1]

[1] Argonne National Laboratory

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

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

ntcp bindings

June 4, 2004

Contents

1 NEES NTCP C Client API

The NEES NTCP API is a client side library used to write C programs that can interact with an NTCP server. The client API provides full functionality for each of the NTCP operations defined by the NTCP schema, as well as querying service data and receiving notifications on state changes.

The standard procedures for using the NTCP client API, in order, are:

1. **Acticate the NTCP module.** This step does setup of the underlying libraries and APIs. See Activation/Deactivation for further info.
2. **Initialize Security Attributes [OPTIONAL].** If security is desired, a security attributes handle must be initialized. See Security for further info.
3. **Initialize an NTCP Handle.** An NTCP handle is used to represent the remove NTCP server. The initialized handle is used by all subsequent NTCP operations. See Handle for further info.
4. **Open a Session.** This is the first operation sent to the remote NTCP server. See NTCP Session Operations for further info.
5. **Perform Transaction Operations.** This is the meat and potatoes. Once a session has been opened, transactions can be proposed, executed, or cancelled, as well as other operations requested of the NTCP server. See NTCP Main Operations for further info.
6. **Close a Session.** Once all NTCP operations are completed, the session should be closed. See NTCP Session Operations for further info.
7. **Destroy the NTCP Handle.** See Handle for further info.
8. **Deactivate the NTCP module.** See Activation/Deactivation for further info.

The NTCP functions all handle errors in the same fashion. The *globus_result_t* variable returned from a function contains success/failure info, as well as further information on an error if one occurred. See Errors for further information on error handling.

2 ntcp bindings Module Index

2.1 ntcp bindings Modules

Here is a list of all modules:

3 ntcp bindings Data Structure Index

3.1 ntcp bindings Data Structures

Here are the data structures with brief descriptions:

4 ntcp bindings Module Documentation

4.1 Activation/Deactivation

The NEES NTCP client API uses standard Globus module activation and deactivation.

Defines

- #define NEES_NTCP_MODULE (&nees_l_ntcp_module)

4.1.1 Detailed Description

The NEES NTCP client API uses standard Globus module activation and deactivation.
Before any NTCP calls can be made, the NTCP module must be activated as follows:

```
globus_module_activate(NEES_NTCP_MODULE);
```

To deactivate the module once NTCP calls are complete, do:

```
globus_module_deactivate(NEES_NTCP_MODULE);
```

This function (and all NTCP functions) returns GLOBUS_SUCCESS if the NTCP module was successfully activated. Otherwise an error reference is returned. The error string from this reference can be extracted using Errors. This function may be called multiple times.

4.1.2 Define Documentation

4.1.2.1 #define NEES_NTCP_MODULE (&nees_l_ntcp_module)

Module Descriptor.

4.2 Handle

The NTCP handle is used to represent the NTCP server endpoint on the client side.

Init Handle

- globus_result_t nees_ntcp_init_handle (nees_ntcp_handle_t ∗handle, const char ∗serviceGSH, globus_ogsa_security_auth_attr_t security_attr, int confidential)

Destroy NTCP Handle

- globus_result_t nees_ntcp_destroy_handle (nees_ntcp_handle_t handle)

4.2.1 Detailed Description

The NTCP handle is used to represent the NTCP server endpoint on the client side.

It maintains the GSH (grid service handle) and security contexts for interacting with the server. Each of the other NTCP client functions that interact with the server take a NTCP handle as input. The standard use of the handle is to initialize it, do NTCP operations, and then destroy.

```
nees_ntcp_handle_t              handle;
result = nees_ntcp_init_handle(&handle);
** check result **

** do operations with handle **

result = nees_ntcp_destroy_handle(handle);
** check result **
```

4.2.2 Function Documentation

4.2.2.1 globus_result_t nees_ntcp_init_handle (nees_ntcp_handle_t ∗ *handle*, const char ∗ *serviceGSH*, globus_ogsa_security_auth_attr_t *security_attr*, int *confidential*)

Initialize the NTCP Handle for use with each of the NTCP functions.

Parameters:

handle A pointer to the handle type, this parameter gets allocated, so it should be non initialized or NULL when passed in. This is the parameter that gets passeed to successive NTCP function calls in this library.

serviceGSH The full NTCP server URI. This will most likely be a string of the form `http://.../ogsa/services/nees/ntcp/NTCPServer`

security_attr This is an already initialized handle to a set of security attributes. The security attributes defines how the user wants security to be performed (which credentials to use, etc). If no security is desired, this parameter should be NULL.

confidential If you want NTCP messages to be secured only with message protection (the messages are visible but signed), this parameter should be NULL or 0. If you want NTCP messages to be secured with message privacy (the messages are encrypted), this parameter shuld be non-null (1).

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_destroy_handle

4.2.2.2 globus_result_t nees_ntcp_destroy_handle (nees_ntcp_handle_t *handle*)

Destroy the NTCP handle.

Once all NTCP functions are completed the NTCP handle should be destroyed. This function performs cleanup and memory deallocation.

Parameters:

handle The handle to be destroyed

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_init_handle

4.3 Security

The NTCP functions provide full security (either protected or private) using WS Secure Conversation.

Initialize Security Attributes

- globus_result_t globus_ogsa_security_authentication_attr_init (globus_ogsa_security_auth_attr_t ∗security_attributes, gss_cred_id_t cred_handle, char ∗target, int options, char ∗∗port_types, char ∗username)

4.3.1 Detailed Description

The NTCP functions provide full security (either protected or private) using WS Secure Conversation.

The security attributes can be setup by a call to Handle, passing in an instance of the globus_ogsa_security_auth_attr_t type. The security attributes are initialized by a call to Security. For example, a security attributes instance might be initialized as follows:

```
globus_ogsa_security_auth_attr_t security_attributes;
result = globus_ogsa_security_authentication_attr_init(
    &security_attributes,
    NULL, NULL,
    GLOBUS_OGSA_SECURITY_NULL_OPTIONS,
    NULL, NULL);
** check result **
```

and then the resulting security attributes would be passed to Handle:

```
nees_ntcp_handle_t handle;
result = nees_ntcp_init_handle(&handle, gsh, security_attributes, 0);
** check result **
```

Notice that in this case the last parameter to Handle is 0, signalling the desire for protected security (message signatures). If private security (message encryption) is desired, this parameter should be non-null.

See also:

globus_ogsa_security_authentication_attr_init

4.3.2 Function Documentation

4.3.2.1 globus_result_t globus_ogsa_security_authentication_attr_init (globus_ogsa_security_auth_attr_t ∗ *security_attributes*, gss_cred_id_t *cred_handle*, char ∗ *target*, int *options*, char ∗∗ *port_types*, char ∗ *username*)

Initialize the security attributes.

Parameters:

security_attributes Pointer to an uninitialized instance of a globus_ogsa_security_auth_attr_t type. This function initializes the instance pointed to by security_attributes. The security attributes instance is required for a call to globus_ogsa_security_authentication_init, which handles the destruction of the security attributes.

cred_handle The credential used by the client to authenticate to the server. This parameter can be NULL, in which case the credential is searched for on the system in the usual locations (this is most often the user's proxy certificate).

target The DN string of the expected server credential. This parameter can be NULL, in which case the server's DN can be anything.

options The security options can be one of:

- GLOBUS_OGSA_SECURITY_NULL_OPTIONS - Basic authentication is used. No delegation, no grim.
- GLOBUS_OGSA_SECURITY_ENABLE_GRIM - Exepect the server's credential to be a GRIM proxy, and do verification of the appropriate parts of the GRIM credential. Only with this option are the other parameters (port_types and username) used.
- GLOBUS_OGSA_SECURITY_DELEGATE - Request that the server initiates delegation of the client's proxy certificate with the client.
- GLOBUS_OGSA_SECURITY_IGNORE_GRIM - Expect the server's credential to be a GRIM proxy, but don't verify any of the parameters of the GRIM credential (just ignore them).

port_types The valid port types for this user are returned as an array of strings (only used for GRIM credentials, if NULL, the port_types are ignored)

username The username expected to be given to the client on the server (based on the client's DN). The username can be NULL, in which case, any username is acceptable. This parameter is only used when options == GLOBUS_OGSA_SECURITY_GRIM, and can/should be NULL otherwise.

Returns:

GLOBUS_SUCCESS if no error occurred, otherwise the result refers to a globus_object_t instance.

4.4 Errors

Nearly all of the functions in the NTCP client API return a *globus_result_t* type.

Error String

- char ∗ nees_ntcp_error_string (globus_result_t result)

4.4.1 Detailed Description

Nearly all of the functions in the NTCP client API return a *globus_result_t* type.

This type represents the success or failure of the function, and maintains errors that occurred within the function. In general, if the function completed successfully, the value returned should be equal to GLOBUS_SUCCESS. If the value returned is not equal to GLOBUS_SUCCESS, the error that occurred can be extracted as a string from the result using Errors. There aren't currently any error codes defined for the NTCP client API.

4.4.2 Function Documentation

4.4.2.1 char∗ nees_ntcp_error_string (globus_result_t *result*)

Get the error string of the result returned from an NTCP function.

Parameters:

result The result returned from an NTCP function

Returns:

NULL if the result parameter indicates no error. Otherwise, the returned value is an allocated string (char ∗) containing a string representation of the error (including the error chain). NOTE: The returned value needs to be freed by the user with free()

4.5 NTCP Session Operations

The NEES NTCP Session module provides client side functions to interact with an NTCP server.

Open

- globus_result_t nees_ntcp_open_session (nees_ntcp_handle_t handle, ntcp__ParameterTypeArray paramArray)

Close

- globus_result_t nees_ntcp_close_session (nees_ntcp_handle_t handle)

4.5.1 Detailed Description

The NEES NTCP Session module provides client side functions to interact with an NTCP server.

4.5.2 Function Documentation

4.5.2.1 globus_result_t nees_ntcp_open_session (nees_ntcp_handle_t *handle*, ntcp__ParameterTypeArray *paramArray*)

Allows the client to open a session at the NTCP server with a list of parameters.

Parameters:

handle The NTCP handle that gets initiated from Handle.

paramArray An initial set of parameters to open the session with. The array can be created using the NTCP Parameter Arrays functions.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_close_session

4.5.2.2 globus_result_t nees_ntcp_close_session (nees_ntcp_handle_t *handle*)

Close the opened session on the NTCP server.

Parameters:

handle the initialized NTCP handle

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_close_session

4.6 NTCP Main Operations

The NEES NTCP API provides client side functions for proposing and managing a transaction at the NTCP server.

Transaction States

- enum ntcp__TransactionStateType

Possible Axis

- enum ntcp__GeomAxisType

ControlPoint Parameter Names

- enum ntcp__ControlPointParameterNameType

Set Parameter

- globus_result_t nees_ntcp_set_parameter (nees_ntcp_handle_t handle, _xsd__string name, _xsd__string value)

Get Parameter

- globus_result_t nees_ntcp_get_parameter (nees_ntcp_handle_t handle, _xsd__string name, _xsd__string ∗value)

Propose

- globus_result_t nees_ntcp_propose (nees_ntcp_handle_t handle, char ∗transactionName, BIGNUM ∗stepNumber, ntcp__ControlPointTypeArray controlPoints, int proposeTimeout, int transactionTimeout, int transactionRememberedUntil, enum ntcp__TransactionStateType ∗resultState)

Cancel

- globus_result_t nees_ntcp_cancel (nees_ntcp_handle_t handle, char ∗transactionName, int interruptWhileExecuting)

Execute

- globus_result_t nees_ntcp_execute (nees_ntcp_handle_t handle, char ∗transactionName)

Get Transaction

- globus_result_t nees_ntcp_get_transaction (nees_ntcp_handle_t handle, char ∗transactionName, ntcp__TransactionType ∗∗transactionType)

Get Control Point

- globus_result_t nees_ntcp_get_control_point (nees_ntcp_handle_t handle, char ∗name, ntcp__ControlPointType ∗resultControlPoint)

Get Parameters

- globus_result_t nees_ntcp_get_parameters (nees_ntcp_handle_t handle, ntcp__ParameterTypeArray ∗resultParameters)

4.6.1 Detailed Description

The NEES NTCP API provides client side functions for proposing and managing a transaction at the NTCP server.

4.6.2 Enumeration Type Documentation

4.6.2.1 enum ntcp__TransactionStateType

Set of values for possible transaction states.

4.6.2.2 enum ntcp__GeomAxisType

Possible Axis values.
\anchor ntcp__GeomAxisType

4.6.2.3 enum ntcp__ControlPointParameterNameType

Possible values for the ControlPoint parameter name.
\anchor ntcp__ControlPointParameterNameType

4.6.3 Function Documentation

4.6.3.1 globus_result_t nees_ntcp_set_parameter (nees_ntcp_handle_t *handle*, _xsd__string *name*, _xsd__string *value*)

Set the parameter (name/value) for the session at the NTCP server.

Parameters:

handle The NTCP handle referencing the NTCP server endpoint

name The new parameter's name. This is a *char* ∗ type.

value The new parameter's value. This is a *char* ∗ type.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_get_parameter

4.6.3.2 globus_result_t nees_ntcp_get_parameter (nees_ntcp_handle_t *handle*, _xsd__string *name*, _xsd__string ∗ *value*)

Get the parameter with the given name from the NTCP server.

Parameters:

handle The NTCP handle that references the NTCP server

name The name string for the parameter at the server that you want to get the value of.

value The value string of the parameter from the NTCP server with the given name

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_set_parameter

4.6.3.3 globus_result_t nees_ntcp_propose (nees_ntcp_handle_t *handle*, char ∗ *transactionName*, BIGNUM ∗ *stepNumber*, ntcp__ControlPointTypeArray *controlPoints*, int *proposeTimeout*, int *transactionTimeout*, int *transactionRememberedUntil*, enum ntcp__TransactionStateType ∗ *resultState*)

Propose a transaction to the NTCP server.

Parameters:

handle The handle to the NTCP server instance

transactionName The name of the proposed transaction, must be unique from other NTCP transaction

stepNumber optional number indicating what time-step in the simulation this proposal represents. If the client does not wish to provide a stepNumber, this parameter should be NULL.

controlPoints An array of per-control-point request parameters

proposeTimeout the proposal expiration time, represented as the number of seconds from the current time. If the proposal is received by the server after the time specified by this parameter, it will be ignored.

transactionTimeout the transaction expiration time, represented as the number of seconds from the current time. If no execute request has been received for the transaction created by this proposal by the time specified here, the transaction will be terminated. The transactionTimeout must not be earlier than the proposeTimeout

transactionRememberedUntil the transaction state timeout for the resulting transaction, represented as the number of seconds from the current time. Once this time expires, the NTCP is allowed to forget the state of the transaction created by this proposal. the transactionRememberUntil timeout must not be earlier than the transactionTimeout.

resultState Gives the result of this transaction proposal request. See NTCP Main Operations

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_ControlPointTypeArray_destroy , nees_ntcp_ControlPointTypeArray_append , nees_ntcp_controlPointTypeArray_init , nees_ntcp_execute , nees_ntcp_cancel

4.6.3.4 globus_result_t nees_ntcp_cancel (nees_ntcp_handle_t *handle*, char ∗ *transactionName*, int *interruptWhileExecuting*)

Cancel a transaction.

Parameters:

handle the NTCP handle representing the NTCP session at the server

transactionName the name of the transaction to cancel

interruptWhileExecuting if non-null, the transaction will be interrupted during execution. Otherwise (if null) the transaction state will be changed to terminated, but will continue to completion.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_propose , nees_ntcp_execute

4.6.3.5 globus_result_t nees_ntcp_execute (nees_ntcp_handle_t *handle*, char ∗ *transactionName*)

Execute a proposed transaction.

Parameters:

handle the NTCP handle referring to the NTCP session at the server

transactionName the name of the transaction to execute

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_propose , nees_ntcp_cancel

4.6.3.6 globus_result_t nees_ntcp_get_transaction (nees_ntcp_handle_t *handle*, char ∗ *transactionName*, ntcp__TransactionType ∗∗ *transactionType*)

Get the current transaction information from the NTCP server.

Parameters:

handle the NTCP handle referring to the NTCP session at the server

transactionName the name of the transaction to get information for

transactionType the resulting transaction information from the server. This instance should be freed by the caller with nees_ntcp__TransactionType_destroy.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_execute , nees_ntcp_cancel

4.6.3.7 globus_result_t nees_ntcp_get_control_point (nees_ntcp_handle_t *handle*, char ∗ *name*, ntcp__ControlPointType ∗ *resultControlPoint*)

Get a control point from the NTCP session.

Parameters:

handle the NTCP handle referencing the NTCP session at the server

name the name of the control point to get

resultControlPoint The resulting controlPoint received from the NTCP server. This should be a pointer to an uninitialized instance of ntcp__ControlPointType. The user is required to free the returned instance with NTCP ControlPointType.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_ControlPointType_destroy , nees_ntcp_ControlPointType_copy

4.6.3.8 globus_result_t nees_ntcp_get_parameters (nees_ntcp_handle_t *handle*, ntcp__ParameterTypeArray ∗ *resultParameters*)

Get the list of parameters from the NTCP server.

Parameters:

handle The NTCP handle representing the NTCP session at the server

resultParameters All the parameters currently maintained at the NTCP server. This is an array of parameters which must be freed by the user with NTCP Parameter Arrays. Other array manipulation functions are described in NTCP Parameter Arrays.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_ParameterTypeArray_length , nees_ntcp_ParameterTypeArray_at_index , nees_ntcp_ParameterTypeArray_destroy

4.7 NTCP Parameter Arrays

The NEES NTCP API provides functions to create, fill, and destroy an array of parameters.

Initialize

- globus_result_t nees_ntcp_ParameterTypeArray_init (ntcp__ParameterTypeArray ∗array)

Append

- globus_result_t nees_ntcp_ParameterTypeArray_append (ntcp__ParameterTypeArray array, char ∗name, char ∗value)

At Index

- globus_result_t nees_ntcp_ParameterTypeArray_at_index (ntcp__ParameterTypeArray array, int index, char ∗∗name, char ∗∗value)

Length

- int nees_ntcp_ParameterTypeArray_length (ntcp__ParameterTypeArray array)

Find

- globus_result_t nees_ntcp_ParameterTypeArray_find (ntcp__ParameterTypeArray array, char ∗name, char ∗∗value)

Destroy

- globus_result_t nees_ntcp_ParameterTypeArray_destroy (ntcp__ParameterTypeArray array)

4.7.1 Detailed Description

The NEES NTCP API provides functions to create, fill, and destroy an array of parameters.

4.7.2 Function Documentation

4.7.2.1 globus_result_t nees_ntcp_ParameterTypeArray_init (ntcp__ParameterTypeArray ∗ *array*)

Initialize a ParameterTypeArray.

The ParameterTypeArray is used to pass parameters to the server and receive parameters from the server. Functions that use parameter arrays are: NTCP Session Operations, NTCP Main Operations.

Parameters:

array A pointer to the to-be-created parameter array.

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_ParameterTypeArray_destroy , nees_ntcp_ParameterTypeArray_append , nees_ntcp_ParameterTypeArray_find , nees_ntcp_ParameterTypeArray_at_index

4.7.2.2 globus_result_t nees_ntcp_ParameterTypeArray_append (ntcp__ParameterTypeArray *array*, char ∗ *name*, char ∗ *value*)

Add a new ParameterType (name/value pair) to the end of the array.

Parameters:

array The array to append a new element to

name The name of the parameter to be appended

value The value of the parameter to be appended

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_ParameterTypeArray_destroy , nees_ntcp_ParameterTypeArray_init , nees_ntcp_ParameterTypeArray_find , nees_ntcp_ParameterTypeArray_at_index

4.7.2.3 globus_result_t nees_ntcp_ParameterTypeArray_at_index (ntcp__ParameterTypeArray *array*, int *index*, char ∗∗ *name*, char ∗∗ *value*)

Get the Parameter from the Parameter array at the given index.

Parameters:

array The array to get the parameter from

index The index of the desired parameter. If this index is out-of-bounds of the array, an error is returned.

name The name of the parameter found at the given index. This string must be freed by the caller with free().

value The value of the parameter found at the given index. This string must be freed by the caller with free().

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_ParameterTypeArray_destroy , nees_ntcp_ParameterTypeArray_init , nees_ntcp_ParameterTypeArray_find , nees_ntcp_ParameterTypeArray_append

4.7.2.4 int nees_ntcp_ParameterTypeArray_length (ntcp__ParameterTypeArray *array*)

Get the length of the parameter array.

Parameters:

array the array to get the length of

Returns:

the length of the array passed in

4.7.2.5 globus_result_t nees_ntcp_ParameterTypeArray_find (ntcp__ParameterTypeArray *array*, char $*$ *name*, char $**$ *value*)

Find the value of the parameter keyed on the given name.

Parameters:

array The array containing the parameter with the given name

name The name of the parameter to find

value The value of the parameter found with name <name>

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, the parameter with the given name could not be found in the array. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_ParameterTypeArray_destroy , nees_ntcp_ParameterTypeArray_init , nees_ntcp_ParameterTypeArray_at_index , nees_ntcp_ParameterTypeArray_append

4.7.2.6 globus_result_t nees_ntcp_ParameterTypeArray_destroy (ntcp__ParameterTypeArray *array*)

Destroy the parameter array.

This function should be called once the parameter array is no longer needed by the program. It does cleanup and deallocation of the array and its members.

Parameters:

array the array to destroy

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, the parameter with the given name could not be found in the array. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_ParameterTypeArray_find , nees_ntcp_ParameterTypeArray_init , nees_ntcp_ParameterTypeArray_at_index , nees_ntcp_ParameterTypeArray_append

4.8 NTCP ControlPointType

Functions to manage ControlPointTypes returned from NTCP operations such as NTCP Main Operations.

Data Structures

- struct ntcp__ControlPointGeomParameterType_s

  Represents the Geometry info of a control point.

- struct ntcp__ControlPointType_s

  The type that represents a control point at the NTCP server.

Control Point Geometry

- typedef ntcp__ControlPointGeomParameterType_s ntcp__ControlPointGeomParameterType

Control Point

- typedef ntcp__ControlPointType_s ntcp__ControlPointType

Copy

- globus_result_t nees_ntcp_ControlPointType_copy (ntcp__ControlPointType from, ntcp__ControlPointType ∗to)

Destroy

- void nees_ntcp_ControlPointType_destroy (ntcp__ControlPointType ∗cp)

4.8.1 Detailed Description

Functions to manage ControlPointTypes returned from NTCP operations such as NTCP Main Operations.

4.8.2 Typedef Documentation

4.8.2.1 typedef struct ntcp__ControlPointGeomParameterType_s ntcp__ControlPointGeomParameterType

Represents the Geometry info of a control point.

4.8.2.2 typedef struct ntcp__ControlPointType_s ntcp__ControlPointType

The type that represents a control point at the NTCP server.

4.8.3 Function Documentation

4.8.3.1 globus_result_t nees_ntcp_ControlPointType_copy (ntcp__ControlPointType *from*, ntcp__ControlPointType ∗ *to*)

Copy the ControlPointType element.

Parameters:

from the ControlPointType instance to be copied

to the resulting copied ControlPointType instance. This should be a pointer to an uninitialized instance of a ControlPointType. The instance returned must be freed by a call to NTCP ControlPointType

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_ControlPointType_destroy , nees_ntcp_propose

4.8.3.2 void nees_ntcp_ControlPointType_destroy (ntcp__ControlPointType ∗ *cp*)

Destroy the ControlPointType instance.

Parameters:

cp the control point type to destroy

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_ControlPointType_copy , nees_ntcp_propose

4.9 NTCP TransactionType

Functions to manage TransactionTypes returned from NTCP operations such as NTCP Main Operations.

Data Structures

- struct ntcp__TransactionType_s

  The type that represents a transaction at the NTCP server.

Transaction

- typedef ntcp__TransactionType_s ntcp__TransactionType

Copy

- globus_result_t nees_ntcp_TransactionType_copy (ntcp__TransactionType ∗from, ntcp__TransactionType ∗∗to)

Destroy

- void nees_ntcp_TransactionType_destroy (ntcp__TransactionType ∗trans)

4.9.1 Detailed Description

Functions to manage TransactionTypes returned from NTCP operations such as NTCP Main Operations.

4.9.2 Typedef Documentation

4.9.2.1 typedef struct ntcp__TransactionType_s ntcp__TransactionType

The type that represents a transaction at the NTCP server.

4.9.3 Function Documentation

4.9.3.1 globus_result_t nees_ntcp_TransactionType_copy (ntcp__TransactionType ∗ *from*, ntcp__TransactionType ∗∗ *to*)

Copy the TransactionType element.

Parameters:

from the TransactionType instance to be copied

to the resulting copied TransactionType instance. This should be a pointer to an uninitialized instance of a TransactionType. The instance returned must be freed by a call to NTCP TransactionType

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_TransactionType_destroy , nees_ntcp_get_transaction

4.9.3.2 void nees_ntcp_TransactionType_destroy (ntcp__TransactionType ∗ *trans*)

Destroy an instance of ntcp__TransactionType.

Parameters:

trans the instance to destroy

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_TransactionType_copy , nees_ntcp_get_transaction

4.10 NTCP State Notifi cations

The NTCP client API provides functions to subscribe to state changes and event updates from an NTCP server.

Control Point

- globus_result_t nees_ntcp_control_point_notify_register (nees_ntcp_handle_t handle, char ∗∗sink_gsh, globus_abstime_t expire, nees_ntcp_control_point_notify_callback_t control_point_callback, void ∗callback_args)

Transaction

- globus_result_t nees_ntcp_transaction_notify_register (nees_ntcp_handle_t handle, char ∗∗sink_gsh, globus_abstime_t expire, nees_ntcp_transaction_notify_callback_t transaction_callback, void ∗callback_args)

Parameters

- globus_result_t nees_ntcp_parameters_notify_register (nees_ntcp_handle_t handle, char ∗∗sink_gsh, globus_abstime_t expire, nees_ntcp_parameters_notify_callback_t parameters_callback, void ∗callback_args)

Unregister

- globus_result_t nees_ntcp_notifications_unregister (nees_ntcp_handle_t handle, char ∗sink_gsh)

4.10.1 Detailed Description

The NTCP client API provides functions to subscribe to state changes and event updates from an NTCP server.

The possible notifications for the NTCP client to receive are based on the NTCP WSDL schema. The following are functions that allow callbacks to be registered to receive these notifications.

4.10.2 Function Documentation

4.10.2.1 globus_result_t nees_ntcp_control_point_notify_register (nees_ntcp_handle_t *handle*, char ∗∗ *sink_gsh*, globus_abstime_t *expire*, nees_ntcp_control_point_notify_callback_t *control_point_callback*, void ∗ *callback_args*)

Register a callback to receive notification events of changes to control points at the NTCP server.

Parameters:

handle NTCP handle representing the NTCP session at the server

sink_gsh The resulting sink handle that is used to receive notification events. This handle is needed to unsubscribe.

expire lifetime of the notification subscription

control_point_callback Function pointer defined by the user that gets called when notification events are received

callback_args a pointer of user data that gets passed to the callback function when called

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_notifications_unregister

4.10.2.2 globus_result_t nees_ntcp_transaction_notify_register (nees_ntcp_handle_t *handle*, char ∗∗ *sink_gsh*, globus_abstime_t *expire*, nees_ntcp_transaction_notify_callback_t *transaction_callback*, void ∗ *callback_args*)

Register to receive notification events of updates to transaction state changes at the NTCP server.

Parameters:

handle NTCP handle reprenting the NTCP server

sink_gsh The resulting sink handle that is used to receive notification events. This handle is needed to unsubscribe.

expire lifetime of the notification subscription

transaction_callback Function pointer defined by the user that gets called when notification events are received

callback_args a pointer of user data that gets passed to the callback function when called

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_notifications_unregister

4.10.2.3 globus_result_t nees_ntcp_parameters_notify_register (nees_ntcp_handle_t *handle*, char ∗∗ *sink_gsh*, globus_abstime_t *expire*, nees_ntcp_parameters_notify_callback_t *parameters_callback*, void ∗ *callback_args*)

Register to receive notification events of state changes for the Parameters held by the NTCP server.

Parameters:

sink_gsh The resulting sink handle that is used to receive notification events. This handle is needed to unsubscribe.

expire lifetime of the notification subscription

parameters_callback Function pointer defined by the user that gets called when notification events are received

callback_args a pointer of user data that gets passed to the callback function when called

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string , nees_ntcp_notifications_unregister

4.10.2.4 globus_result_t nees_ntcp_notifications_unregister (nees_ntcp_handle_t *handle*, char ∗ *sink_gsh*)

Unregister a notification subscription.

Parameters:

handle The NTCP handle representing the NTCP server

sink_gsh The sink subscribed to receive notifications

Returns:

If the returned value is not equal to GLOBUS_SUCCESS, an error occurred. See Errors for further information on extracting the error string from the result.

See also:

nees_ntcp_error_string

5 ntcp bindings Data Structure Documentation

5.1 ntcp__ControlPointGeomParameterType_s Struct Reference

Represents the Geometry info of a control point.

Data Fields

- enum ntcp__GeomAxisType axis
- enum ntcp__ControlPointParameterNameType name
- float value

5.1.1 Detailed Description

Represents the Geometry info of a control point.

5.1.2 Field Documentation

5.1.2.1 enum ntcp__GeomAxisType ntcp__ControlPointGeomParameterType_s::axis

Geometry Axis.

5.1.2.2 enum ntcp__ControlPointParameterNameType ntcp__ControlPointGeomParameterType_s::name

Type of control point.

5.1.2.3 float ntcp__ControlPointGeomParameterType_s::value

value

5.2 ntcp__ControlPointType_s Struct Reference

The type that represents a control point at the NTCP server.

Data Fields

- _xsd__string controlPointName
- ntcp__ControlPointGeomParameterType geom
- _ogsi__ExtensibilityType extension

5.2.1 Detailed Description

The type that represents a control point at the NTCP server.

5.2.2 Field Documentation

5.2.2.1 _xsd__string ntcp__ControlPointType_s::controlPointName

The name of the control point.
This should be equivalent to a *char* ∗

5.2.2.2 ntcp__ControlPointGeomParameterType ntcp__ControlPointType_s::geom

control point geometry info

5.2.2.3 _ogsi__ExtensibilityType ntcp__ControlPointType_s::extension

An extension to allow for other application specific parameters.

5.3 ntcp__TransactionType_s Struct Reference

The type that represents a transaction at the NTCP server.

Data Fields

- _xsd__string name
- enum ntcp__TransactionStateType state
- int __sizeRequestedControlPoints
- ntcp__ControlPointType ∗ requestedControlPoints
- int __sizeResultingControlPoints

- ntcp__ControlPointType ∗ resultingControlPoints
- _xsd__string transactionProposerName
- _ogsi__ExtendedDateTimeType transactionTimeout
- _ogsi__ExtendedDateTimeType transactionRememberedUntil
- _ogsi__ExtendedDateTimeType transactionExecutionBeginTime
- _ogsi__ExtendedDateTimeType transactionTerminationTime

5.3.1 Detailed Description

The type that represents a transaction at the NTCP server.

5.3.2 Field Documentation

5.3.2.1 _xsd__string ntcp__TransactionType_s::name

Name of the Transaction.

5.3.2.2 enum ntcp__TransactionStateType ntcp__TransactionType_s::state

Current Transaction State.

5.3.2.3 int ntcp__TransactionType_s::__sizeRequestedControlPoints

size of the requested control points array

5.3.2.4 ntcp__ControlPointType∗ ntcp__TransactionType_s::requestedControlPoints

Array of control points requested for the transaction.

5.3.2.5 int ntcp__TransactionType_s::__sizeResultingControlPoints

size of the resulting control points array

5.3.2.6 ntcp__ControlPointType∗ ntcp__TransactionType_s::resultingControlPoints

Array of control points resulting from transaction execution.

5.3.2.7 _xsd__string ntcp__TransactionType_s::transactionProposerName

name of the transaction proposer (a *char* ∗)

5.3.2.8 _ogsi__ExtendedDateTimeType ntcp__TransactionType_s::transactionTimeout

timeout for the transaction

5.3.2.9 _ogsi__ExtendedDateTimeType ntcp__TransactionType_s::transactionRememberedUntil

time before transaction state is forgotten

5.3.2.10 _ogsi__ExtendedDateTimeType ntcp__TransactionType_s::transactionExecutionBeginTime

timestamp of transaction start

5.3.2.11 _ogsi__ExtendedDateTimeType ntcp__TransactionType_s::transactionTerminationTime

timestamp of transaction termination

Index