



NetApp™

Go further, faster™

NetApp Manageability SDK Training

C/C++ Programming



SDK C/C++ Interface – Core Functions

Four sets of Core API Families:

- Start-up and Memory management APIs
 - *na_elem_new()*, *na_elem_free()*, *na_startup()*, *na_shutdown()*
 - These APIs provide functionalities like
 - Setting/freeing system resources for the program
 - Creating/Cleaning resources for program elements

- I/O Element Management APIs
 - *na_child_**(), *na_*_iterator()*, *na_elem_**()
 - The I/O elements are akin to XML containers. They have names and values. They can be nested arbitrarily.
 - These APIs provide functionalities like
 - Setting/getting the contents of the I/O elements
 - Embedding/Extracting the child I/O elements

SDK C/C++ Interface – Core Functions

■ Session Management APIs

- *na_server_**(), *na_server_get_**(), *na_server_set_**(), *na_server_invoke_**()
- These APIs provide functionalities like
 - Open/Close Server connections
 - Set/Get the Server session parameters like server type, transport type, authentication style etc
 - Send commands to the Server

■ Results Checking Routines

- *na_results_status()*, *na_results_reason()*, *na_results_errno()*
- These APIs provide functionalities like
 - Check the status of API requests to the Server
 - Get the error number and error reason string in case of API failure

Configuring Session Parameters

- Server type
 - Indicates the target appliance for API calls
 - It can be set using the function *na_server_set_server_type()*
 - parameter to *na_server_set_server_type()* method can take the following values:
 - *NA_SERVER_TYPE_FILER*
 - To send API calls to NetApp Storage system
 - *NA_SERVER_TYPE_DFM*
 - To send API calls to DataFabric Manager server

Configuring Session Parameters

■ Server style

- Indicates the authentication mechanism to be used for communicating with the Server
- It can be set using the function *na_server_set_style()*
- parameter to *na_server_set_style()* function can take the following values:
 - *NA_STYLE_LOGIN_PASSWORD*
 - Causes the Server to use HTTP simple authentication with a username and password.
 - User needs to provide login/password details with *na_server_adminuser()* method
 - *NA_STYLE_HOSTSEQUIV*
 - Causes the Server to authenticate against the */etc/hosts.equiv* file on the filer
 - *NA_STYLE_RPC*
 - Causes the Server to use native Windows authentication and authorization mechanism

Configuring Session Parameters

- Transport type
 - Indicates the transport mechanism to be used for sending API commands
 - It can be set using the function *na_server_set_transport_type()*
 - Parameter to *na_server_set_transport_type()* method can take the following values:
 - NA_SERVER_TRANSPORT_HTTP
 - Uses HTTP as transport protocol
 - This is the default transport type
 - NA_SERVER_TRANSPORT_HTTPS
 - Uses HTTPS as transport protocol
 - Provides secure communication by encrypting the API requests/responses

Configuring Session Parameters

- Server Port
 - Indicates the port on/to which API requests to be sent
 - Depends on 'Transport type' and 'Server type' settings
 - It can be set using the function *na_server_set_port()*
 - This setting should be done only after setting 'Transport type' and 'Server type' parameters

Configuring Session Parameters

- Server timeout
 - Sets a timeout on the server when making connection or when reading/writing data from/to a socket
 - It can be set using the function *na_server_set_timeout()*
 - The timeout parameter is in number of seconds
 - If timeout is non-zero the server will use non blocking socket I/O otherwise blocking I/O will be used.
 - By default Blocking I/O is used

C/C++ Programming with SDK – Basic Steps

- STEP 1: Create Server Context on Client
 - Load the libraries and Proc addresses
 - *na_startup()*
 - Create Server Object with Server Name / IP, Product API version to be used as inputs
 - *na_server_t *S = na_server_open (Server-Name, majorversion, minorversion)*

- STEP 2: Set Session parameters on Client
 - Set server type : *na_server_set_server_type()*
 - Set Authentication style : *na_server_set_style()*
 - Set Transport type : *na_server_set_transport_type()*
 - Set Server port : *na_server_set_port()*
 - Set Server timeout : *na_server_set_timeout()*

C/C++ Programming with SDK – Basic Steps

■ STEP 3: Send Commands to the Server

- Create a Command element and use the Core API *na_server_invoke_elem()*

E.g.

```
na_element_t *elem = na_elem_new("system-get-version");  
na_server_invoke_elem(S, elem);
```

OR

- Send the Command as a parameter & its arguments as name/value pairs to the Core API *na_server_invoke()*

E.g.

```
na_server_invoke(S, "system-get-version", NULL);
```

- Always terminate the list with NULL (there must always be an even number of arguments to this function).
- Only char data type can be passed in name/value pairs.

C/C++ Programming with SDK – Basic Steps

- STEP 4: Check for API Failures
 - SDK APIs can have two types of failures:
 - Core API failure
 - The return value of the core API should be checked to determine the error
 - E.g.;

```
na_elem_t* out = na_child_add();  
if (out == NULL) {  
    print {"error adding element" } }
```
 - Product API failure
 - *na_results_*()* APIs should be used to determine the Product API error
 - E.g.;

```
na_elem_t* out = na_server_invoke();  
if (na_results_status(out) != NA_OK) {  
    print { "Error %s, error code %d", na_results_reason(out),  
        na_results_errno(out) } }
```

C/C++ Programming with SDK – Basic Steps

■ STEP 5: Parse the command results

- The `invoke*()` Core APIs always return `na_element` type
- The result of Product API is embedded as a child `na_element` in the `na_element` object returned from `invoke*()` Core APIs

E.g.:

```
na_elem_t* out = na_server_invoke(s, "quota-list-entries", NULL);  
quotas = na_elem_child(out, "quota-entries")
```

- To iterate over the list of child elements in the resulted `na_element` object, use the iterate functions

E.g.:

```
for (iter=na_child_iterator(quotas); (ss=na_iterator_next(&iter)) != NULL; )  
{ "Do some processing" }
```

- Call the "`na_iterator_next()`" function till all the child elements are extracted

Compiling SDK C/C++ Program

- Add '\$ONTAPI_ROOT/include' in the SDK Application's include path.
- Add '\$ONTAPI_ROOT/lib/<architecture>' in the SDK Application's library path.
- Link the libraries 'libnetapp.lib, libxml.lib, libadt.lib, libpthread.lib, ntapadmin.lib' into the SDK Application .
- When using RPC on Windows, users should
 - Install ntapadmin.dll, pthread.dll into the %systemroot%/system32 directory on the machine that will be running the SDK Application.
- When using HTTPS transport in the SDK Application, include the following in Library Path/PATH
 - libcrypto.so, libssl.so (Solaris)
 - libcrypto.a, libssl.a (AIX)
 - libeay32.dll, ssleay32.dll (Windows)
- Check SDK documents to know the details of the libraries for Multithreaded applications and 64 bit platforms.



SDK C/C++ Interface – Differences from other interfaces

- Provides an option to set timeout on Server
- Supports RPC style transport/authentication mechanism on Windows
- Needs iterator functions *na_iterator_**() to run through the list of child elements
- Needs to call start up function *na_startup()* to load the API libraries

Tutorial Exercises

Tutorial Exercises

1. Monitor volumes on a NetApp storage system and print message when space usage crosses a threshold

▶ Hint:

- ▶ Get list of volumes with the command '*volume-list-info*'
- ▶ Extract values of '*size_total*' and '*size_used*' fields from '*volume_info*' return element

Tutorial Exercises Contd..

2. Retrieve the quota information of a NetApp storage system in a secure way and print quota information : quota-target,vol name,quota-type
- ▶ Hint:
 - ▶ Set the transport type as 'TRANSPORT_TYPE_HTTPS'
 - ▶ Get the quota list through the command '*quota-list-entries*'

Tutorial Exercises Contd..

3. Enable NFS server on NetApp storage system and check the status of the NFS server. Get the list of the export rules on the storage system.

▶ Hint:

- ▶ Use the command '*nfs-enable*' for enabling NFS server
- ▶ Use the command '*nfs-status*' to check the status
- ▶ Use the command '*nfs-exportfs-list-rules*' to get the list of the NFS export rules



Thank You