



Go further, faster™

NetApp Manageability SDK

Technical Overview





NM SDK Overview

- Provides infrastructure to develop applications to **monitor** and **manage** NetApp® storage system.
- Bundle of open interfaces (APIs) to enable easy **development of interoperable** management solution
- Enables **integration** with third party management tool, orchestration tool, or home-grown tool in a customer environment.
- Product Offering includes:
 - SDK API library bindings in C/C++, .NET, Java and Perl
 - Supported on Windows, Linux (RHEL/SLES/OEL), Solaris, HP-UX, AIX, VMware ESX
 - Detailed documentation of APIs in Data ONTAP
 - Detailed documentation of APIs in DataFabric Manager suite:
 - APIs for Operations Manager, Performance Advisor, Provisioning & Protection Manager
 - **Web Services** support
 - Sample codes, Developer Tools, Design Guides
- Offered **free** of cost to customers and partners since June 2003
- More than 350 participating companies – both partners and customers
- Supported through:
 - [Forum](#) (Preferred)
 - Mailing List : dl-sdk-answers@netapp.com
 - NGS



Features

- Provides access to [Data ONTAP](#) features through APIs.
- Provides access to all features of [DataFabric Manager Suite](#) of products through APIs.
 - Operations Manager
 - Performance Advisor
 - Provisioning Manager
 - Protection Manager
- Supported on multiple platforms
 - Windows, Linux (RHEL/SLES/OEL), Solaris, HP-UX, AIX, VMware ESX
- Supports multiple language interfaces
 - C/C++, C#, VB.NET, Java, and Perl
 - [Web Services](#) support for DataFabric Manager suite
- Supports multiple transport protocols
 - HTTP, HTTPS and Windows RPC
- Supports different authentication mechanisms
 - login/password, host.equiv and Windows RPC authentication
- Dynamic Library feature for C and C++

Features (contd ..)

- SDK core APIs are a small set of immutable Core APIs
 - Easy to learn API programming
 - The existing core APIs ideally do not get changed, there may be new additions though.
 - Easy to add new ONTAP API commands
- Simple to use APIs
 - XML based request/response mechanism
 - XML type name-value pair arguments
- RBAC support
 - Make use of RBAC in Data ONTAP and Operations Manager
- Multithreading
 - SDK APIs are Thread-safe
- Faster integration and storage abstraction by leveraging policy based management tools as point of integration
 - Policy-based efficient storage provisioning
 - Policy-based data protection
 - Operational Monitoring

Comparison of different Interfaces

Interface	Performance	Features	Ease-Of-Use	Standards Based	TCO
API	High	All features of Storage system	High	Partial (XML based)	Medium
SNMP	High	Only get (data) features	High	Yes	Low
SMI-S	Medium (2-level of indirection)	Mainly standard SMI-S profiles	Medium	Yes	Low
CLI (RSH /SSH)	Medium (Considering time for parsing output)	All features of Storage system	Medium	No	High

Configurable Session Parameters

- SDK provides flexibility for the clients to configure the session parameters
- Following session parameters are configurable
 - Server type
 - Server style
 - Transport type
 - Server port
- These session parameters are persistent till the session is closed



Configurable Session Parameters

- Server type
 - Indicates the target appliance for API calls
 - It can be set to
 - FILER
 - API calls to NetApp Storage system
 - DFM
 - API calls to NetApp DataFabric Manager server

Configurable Session Parameters

■ Server style

- Indicates the authentication mechanism to be used for communicating with the Server
- It can be set to
 - RPC
 - Available only on Windows
 - Uses native Windows authentication and authorization
 - hosts.equiv
 - Authenticates against the /etc/hosts.equiv file on the filer
 - hosts.equiv file entries are in the following format
 - *Hostname [User name]*
 - Requests coming from the named host and the named user are allowed in as root
 - login-password
 - Authenticates against the user provided login/password details

Configurable Session Parameters

■ Transport type

- Indicates the transport mechanism to be used for sending API commands
- It can be set to
 - HTTP
 - Default transport type
 - Must use login-password/hosts.equiv mechanism for authentication
 - HTTPS
 - Similar to HTTP connection
 - Provides secure communication by encrypting the API requests/responses

Configurable Session Parameters

- Server Port
 - Indicates the port on/to which API requests to be sent
 - Depends on 'Transport type' and 'Server type' settings
 - Filer default port settings
 - HTTP port – 80
 - HTTPS port – 443
 - DFM default port settings
 - HTTP port – 8088
 - HTTPS port – 8488
 - 'Server Port' setting should be done only after setting 'Transport type' and 'Server type' values



Default Session Parameters

- Server Type
 - FILER
- Transport Type
 - HTTP
- Server Style (authentication mechanism)
 - Login-Password
- Server Port
 - 80
- User
 - root



Manage ONTAP SDK APIs

SDK has two sets of APIs

- Core APIs
 - Provide the client side interfaces
 - Used for setting Server session on the client and sending/receiving the commands to the Server

- Product APIs (Called ZAPIs)
 - Data ONTAP APIs and DataFabric Manager APIs
 - Expose the features of the Server through API commands
 - Used as command parameters for the Core APIs

SDK Core APIs

- Core APIs in C

- na_elem_*
- na_server_*
- na_child_*
- na_results_*

- Core classes in Java

- NaServer
- NaElement
- NaAPIFailedException

- Core classes in PERL

- NaElement
- NaServer

- Core APIs in .NET

- NaServer
- NaElement
- NaException
- NaApiFailedException



Data ONTAP APIs

- Provide APIs that correspond to ONTAP CLI commands

- Provide RBAC support
 - ONTAP API access can be restricted as per the user
 - '*useradmin*' command of Data ONTAP is used to control the API access privileges
 - This feature is supported from Data ONTAP 7G



DataFabric Manager APIs

- Provide APIs for all applications in DataFabric Manager server
 - Operations Manager
 - Performance Advisor
 - Provisioning Manager
 - Protection Manager

- Provide RBAC support
 - Make use of RBAC available in Operations Manager

SDK Programming – Basic Steps

Step 1 : Create Server Context on Client

- Create Server Context object
- Specify the Server details (server name, IP)
- Specify the Product API version to be used

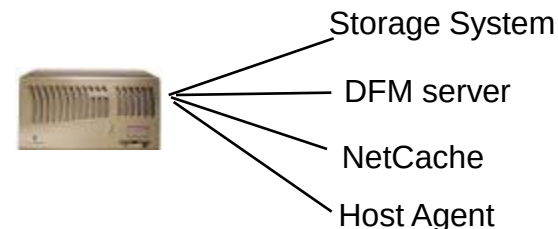


Client System

E.g.: In C: `na_startup();`
`na_server_open(filename, 1, 0);`

Step 2 : Set Session Parameters on Client

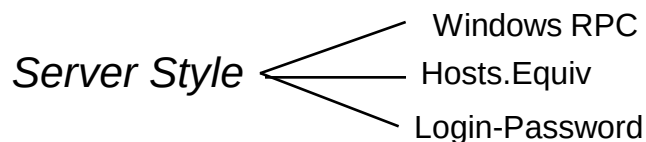
- Select Server type



E.g.: In C: `na_server_set_server_type(NA_SERVER_TYPE_FILER);`

SDK Programming – Basic Steps

- Select Authentication mechanism



E.g.: Setting Server Style in C:

```
na_server_set_style(NA_STYLE_HOSTSEQUIV);
```

- Select Transport mechanism



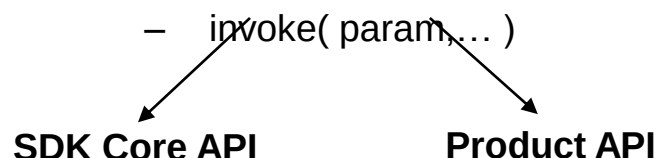
E.g.: Setting Transport Type in C:

```
na_server_set_transport_type(NA_SERVER_TRANSPORT_HTTPS);
```

SDK Programming – Basic Steps

Step 3: Send commands to the server

- There are 2 approaches to send the command
 - Commands are sent to the server by calling the `invoke()` Core API with Product API as a parameter, and its arguments as name/value pairs



- Example:
 - Invoking Data ONTAP API “system-get-version”
 - C: `na_server_invoke(“system-get-version”)`
 - Invoking Operations Manager API “dfm-about”
 - C: `na_server_invoke(“dfm-about”)`
- Commands can also be sent to the server by building a Command Object and encapsulating the Product API & its arguments in it. Then pass the Command Object through the `invokeElem()` Core API
 - Example:
 - Invoking Data ONTAP API “system-get-version”
Java: `elem = new NaElement (“system-get-version”);
NaServer.invokeElem(elem);`
 - Invoking Operations Manager API “dfm-about”
Java: `elem = new NaElement (“dfm-about”);
NaServer.invokeElem(elem)`

SDK Programming – 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 can 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 invoking the command" } }
```

SDK Programming – 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, "system-get-version", NULL);
```

Return from `invoke*()` Core API

```
na_child_get_string(out, "version")
```

Result of the Product API,
embedded as a child element



Useful Tools for SDK Programmers

- Tools bundled with NM SDK
 - ZEXPLORE
 - GUI based utility to try out Data ONTAP APIs
 - apitest.exe
 - Command Line utility to try out both Data ONTAP and DataFabric Manager server APIs
- SIMULATOR
 - Simulates Data ONTAP
 - Can be used to experiment with and develop applications that use Data ONTAP APIs



Limitation

- **SDK will support only UTF-8 character encoding format**



Other NetApp Management Interfaces

- **SNMP**
 - Is a well-known standard for network monitoring
 - Can be used for monitoring NetApp storage
 - Only 'GET' and 'Trap' operations of SNMP are supported by NetApp Storage
 - SDK bundles Data ONTAP SNMP MIB

- **SMI-S Provider**
 - SMI-S is a storage management interface standard developed by the Storage Networking Industry Association (SNIA)
 - NetApp SMI-S Provider can be used for monitoring and managing NetApp Storage
 - The Provider uses ZAPI calls to talk to the Storage

- **RSH/SSH/Telnet (CLI)**
 - These interfaces can be used to invoke the CLI commands on Storage
 - Main issue in using these interfaces is CLI changes



Other NetApp Management Interfaces

■ NDMP

- Standard for controlling network-based backups of file servers to tape servers and jukeboxes.
- The only protocol supported by NetApp for writing backup applications.
- No ONTAPI interfaces for performing backup tasks are presently included in the SDK.

■ Win32

- NetApp Storage systems emulate Windows servers by implementing underlying management RPCs.
- The CIFS protocol code on NetApp Storage systems handle the Win32 API calls translated into SMB requests by CIFS clients on Windows servers
- The SDK includes a list of Win32 APIs supported by NetApp Storage system.

Troubleshooting

Troubleshooting Help

■ Tools

- apitest (Bundled with SDK, CLI tool to test APIs)
- ZExplore (Bundled with SDK, GUI tool to test APIs)
- Evt2Txt (<http://now.netapp.com/NOW/download/tools/evt2text>)
- coregdb (/usr/local/bin/coregdb, core dump analysis tool)
- Ethereal (<http://www.ethereal.com/>, network analysis tool)

■ Documents (Part of SDK Help)

- Version Matrix
- Release History
- Known issues
- API documentation
- Troubleshooting Guidelines
- Frequently Asked Questions

■ Logs

- Audit Logs (On Storage system: /etc/log/auditlog, Provides API invocation history)
- EMS Logs (On Storage system: /etc/log/ems, provides API error messages)
- Syslog (On Storage system: /etc/messages, provides API error messages)
- DFM server Logs (On DataFabric Manager server: <dfm-server-install-directory>/log/. Provides error logging information about API failures)

Troubleshooting Example - 1

- Scenario: Client Application is not receiving complete data from Data ONTAP API (especially when requested large data) though no error is received by the application
 - Possible Cause:
 - Server taking lot of time to process the request as huge data is requested. This might trigger server to reset the connection.
 - The reset packets are received at the transport layer of the Client host which is not handled at the API layer of the Client application. So client may not be getting proper error code.
 - Troubleshooting Method:
 - Get the TCP packet trace using tools like *Ethereal* and check for any possible error messages from the server
 - Possible Work around:
 - Increase the connection timeout
 - Use the iterative APIs when requesting huge data (like thousands of qtrees)

Troubleshooting Example - 2

- Scenario: Client Application is getting 'Unsupported API' error
 - Possible Cause:
 - Older version of Data ONTAP being used with newer SDK having new Data ONTAP API support
 - SDK documentation erroneously included an unsupported API
 - Troubleshooting Method:
 - Check the Data ONTAP Version used by the customer and verify whether that API is supported in that Data ONTAP version – '*SDK release history*' section provides which APIs are supported in which version of Data ONTAP
 - Try using that API with *apitest* tool to check if there are any SDK documentation errors

Troubleshooting Example - 3

- Scenario: Client Application is getting 'No such file or directory' error (errno=13001)
 - Possible Cause
 - Bug in the API handler code on Data ONTAP
 - Application is sending wrong input (example wrong iteration tag input to iterative APIs)
 - Race condition in iterative APIs
 - Troubleshooting Method:
 - Check the *Audit Log* (/etc/log/auditlog) for API invocation history. It will give hints on possible concurrent calls of APIs leading to race conditions
 - Check *EMS Log* (/etc/log/ems) for any errors occurred during processing the API requests (like wrong iteration tag etc)
 - Possible Work around:
 - Upgrade to Data ONTAP 7.3. Race condition of iterative APIs issue is fixed in Data ONTAP 7.3

Troubleshooting Example - 4

- Scenario: Storage system crashed when Client Application sends a request to it
 - Possible Cause:
 - API request is taking long time to process
 - API bug got exposed due to improper input by the Application
 - Troubleshooting Method:
 - Check the core file at /etc/crash/core*
 - Use coregdb tool (/usr/local/bin/coregdb) to load the core
 - ```
$ coregdb core.118048850.2007-09-10.13_43_46.nz
```

```
> cd /x/eng/rlse47/filer/nightly/RironcityN_070906_2230/prod/build
```

```
> amd64-netapp-gdb -nx -x
```

```
#0 dumpcore (
```

```
 pmsg=0x3b064a0 "process on cpu1 hung (api_mpool_07) for 5001 milliseconds!
```

```
 in process api_mpool_07 on release NetApp Release 7.2.3")
```
    - Check if the message has any 'api\_\*' string. If so the crash is caused due to API
    - If the message says it is crashed because API took long time then the API was requesting lot of Data
  - Possible Work around:
    - Use iterative APIs while requesting possible huge data (like thousands of qtrees)
    - Long running process issue has been resolved in 7.3 and 7.2.3P1

# Troubleshooting Example - 5

- Scenario: Client Application hangs
  - Possible Cause:
    - Storage system rebooted during the API processing and application is using blocking connection APIs.
    - Network failure
  - Troubleshooting Method:
    - Check uptime of the NetApp storage system (use '*uptime*' command). If the uptime indicates storage system is booted recently then
      - Check if there is a recent crash: /etc/crash/core-\* file
        - Load the core file if there is a recent one (use coregdb tool as explained in Scenario 4.)
        - Check if the message string has any 'API' related info. If there is any API related message then it could be potential API/Data ONTAP bug
    - If there is no crash or no API related message in the crash stack trace then there was some other issue like power failure or network problem
  - Possible workaround:
    - Restart the Client Application
    - Use non-blocking calls in C (use set-time-out() function for changing blocking to non-blocking)

# Tutorial Exercises

# Tutorial Exercises

1. **Install ONTAP simulator and get API version on the simulator using *zexplore* tool**
  - ▶ **Hint:** Use the command '*system-get-ontapi-version*'
  
2. **Create a Flexvol using *apitest.exe* tool and then check the volume using *FilerView* tool**
  - ▶ **Hint:**
    - ▶ Get the list of aggregates on the NetApp storage system with '*aggr-list-info*' command
    - ▶ Create the flex-volume on one of the aggregates with the command '*volume-create*'



# Thank You