CHAPTER 7

Java SDK for XML PI

E-DI provides Java client API libraries which take Java objects as inputs and return Java objects, so that customers can use these APIs to develop their own applications. They can also use these APIs in their existing applications.

E-DI provides these NetConfSession APIs:

- Connecting to E-DI
- Getting managed devices
- Getting the basic inventory from managed devices
- Getting the configuration of the managed devices
- Configuring the managed devices
- Error/Exception handling classes

All APIs allow per device operations as per the NETCONF protocol.

The APIs also provide object models for the get and set methods (that is, for NETCONF schemas and notifications).

Though connection, set and get methods take Java objects as inputs, they are internally converted to NETCONF complaint XML objects which communicate with devices.

In the current release, Java SDK provides only SSH1 connection APIs.

All classes meet draft-ietf-netconf-prot-07.txt specifications.

Note After opening a session with E-DI using XML session, you should ensure that a Hello request is sent. This should be done before associate-device request is sent.

Accessing the Java APIs

E-DI provides the following sets of NETCONF APIs:

- E-DI Server Management APIs
- Device Management APIs

After you install E-DI, the APIs can be accessed from <*EDI location*>

E-DI Server Management APIs

E-DI provides the following Server Management APIs:

- com.cisco.edi.ncclient.api.NetconfCreateCredentialSet
- com.cisco.edi.ncclient.api.NetconfManageDevice
- com.cisco.edi.ncclient.api.NetconfGetDevices
- com.cisco.edi.ncclient.api.NetconfTransform

com.cisco.edi.ncclient.api.NetconfCreateCredentialSet

com.cisco.edi.ncclient.api.NetconfCreateCredentialSet is used to create a credential set, which can be used to manage devices in E-DI. At present, only Telnet transport type is supported for credential set creation.

The following is the constructor. If the `credentialSetName` is null (empty string) it throws an exception:

```
public NetconfCreateCredentialSet (String credentialSetName, String login, String
password, String enable, String readCom, String writeCom, String transportType) throws
Exception
```

Here,

`credentialSetName` is the credential set to be created.

`login` is the login ID.

`password` is the password.

`enable` is the enable password.

`readCom` is the SNMP read community

`writeCom` is the SNMP read community

`transportType` is the transport type (only Telnet is supported.)

```
public String toNetconfXMLRequestString(String messageId)
```

returns NetConf request for creating the credential set.

com.cisco.edi.ncclient.api.NetconfManageDevice

`com.cisco.edi.ncclient.api.NetconfManageDevice` is used to manage a device in E-DI.

The following is the constructor. If any of the fields are null (empty string) it throws an exception.

```
public NetconfManageDevice (String ip, String credentialSetName) throws Exception
```

Here, `ip` represents the IP address of the device that needs to be managed.

`credentialSetName` represents the credential set for managing the device.

```
public String toNetconfXMLRequestString(String messageId)
```

returns a NetConf request for managing a device.

com.cisco.edi.ncclient.api.NetconfGetDevices

com.cisco.edi.ncclient.api.NetconfGetDevices is used to get the information about the online and offline devices.

The following is the default constructor:

```
public NetconfGetDevices
```

```
public String toNetconfXMLRequestString(String messageId, boolean online)
```

returns NetConf request for getting information about online and offline devices.

If online = true, this function returns request for online devices.

If online =false, this function returns a request for offline devices.

com.cisco.edi.ncclient.api.NetconfTransform

com.cisco.edi.ncclient.api.NetconfTransform is used to transform the configuration or the operational data into XML format without connecting to the device. It takes the configurational or the operational data in the form of a string.

The following is the constructor. If any of the fields are null (empty string) it throws an exception.
`public NetconfTransform( String devicetype, String osversion, String clidata) throws Exception`

User needs to specify device type, OS version exactly from the E-DI supported OS-Device types, in the available constructors, along with the configurational or operational data.

To transform configurational data, user needs to set the IDU version as well. The following returns the XML request for the configurational data:

```
public String getNetconfC2XRequest(String messageId)
```

To transform operational data, user needs to set the show command as well. The following returns the XML request for the operational data:

```
public String getNetconfShowCmdRequest(String messageId, String Command)
```

For message ID information, see Note.

Device Management APIs

E-DI provides the following Device Management APIs:

- com.cisco.edi.ncclient.api.NetconfHello
- com.cisco.edi.ncclient.api.NetconfGet
- com.cisco.edi.ncclient.api.NetconfGetConfig
- com.cisco.edi.ncclient.api.NetconfCopyConfig
- com.cisco.edi.ncclient.api.NetconfValidate
- com.cisco.edi.ncclient.api.NetconfLock
- com.cisco.edi.ncclient.api.NetconfCommit
- com.cisco.edi.ncclient.api.NetconfDiscardChanges

- com.cisco.edi.ncclient.api.NetconfUnLock
- com.cisco.edi.ncclient.api.NetconfCloseSession
- com.cisco.edi.ncclient.api.NetconfKillSession
- com.cisco.edi.ncclient.api.NetconfDeleteConfig
- com.cisco.edi.ncclient.clienttool.session.XMLSession
- com.cisco.edi.ncclient.api.util.Device

com.cisco.edi.ncclient.api.NetconfHello

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns Hello request.

Hello request must be sent before associating device or server, after opening an XML session. To open an XML session see com.cisco.edi.ncclient.clienttool.session.XMLSession

At times the Hello request may not give a response.

Note If you specify a value for the RPC message ID, then the RPC reply will also have this value. The default message ID is 1.

com.cisco.edi.ncclient.api.NetconfGet

NetconfGet retrieves configuration information. To retrieve configurational information, the NetconfGet is equivalent to NetconfGetConfig with running configuration as the source.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfGet request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfGetShowCmd

NetconfGetShowCmd is used to retrieve operational information. It takes `showcommand` as the input.

If the `showcommand` is null (empty string) it throws an exception.

```
public String getNetconfShowCmdRequest(messageId)
public String getNetconfShowCmdRequest()
```

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfGetConfig

NetconfGetConfig is used to retrieve all or part of the configuration of the device. A filter may be specified to retrieve only a part of the configuration. See com.cisco.edi.ncclient.api.NetconfFilter.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns GetConfig request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfFilter

NetconfFilter is used to prepare a request to get a part of the configuration. NetconfGetConfig uses this filter to specify what configuration to get.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns GetNetconf XML String.

User needs to set device type, OS version and IDU version exactly from the E-DI supported OS-Device types.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfCopyConfig

NetconfCopyConfig creates or replaces an entire configuration datastore with the contents of another complete configuration datastore. If the target datastore exists, it is overwritten. Otherwise, a new one is created.

The source can be any running, startup or candidate configuration datastore.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfCopyConfig request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfValidate

NetconfValidate validates the contents of the specified configuration.

Cisco E-DI supports the validate operation on candidate, running and startup configurations. Currently, Cisco E-DI only does syntax checking.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfValidate request.

com.cisco.edi.ncclient.api.NetconfLock

NetconfLock allows the client to lock a local configuration datastore of a device in Cisco E-DI. The datastore can be any running, startup or candidate configuration datastore. If a datastore is locked, access to it is limited to read-only for any other clients. All locks will be cleared on a normal or abnormal session termination.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfLock request.

com.cisco.edi.ncclient.api.NetconfCommit

When a candidate's configuration content is complete, the configuration data can be committed, publishing the dataset to the device, and requesting the device to conform to the behavior described in the new configuration. NetconfCommit is used to commit the candidate configuration as the new current configuration of the device.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfCommit request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfDiscardChanges

If the client decides that the candidate configuration should not be committed, NetconfDiscardChanges can be used to revert the candidate configuration to the current committed configuration.

If the client decides that the candidate configuration should not be committed, NetconfDiscardChanges can be used to discard the candidate configuration. Cisco E-DI then sets the candidate configuration to an empty store.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfDiscardChanges request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfUnLock

NetconfUnLock releases a configuration lock, previously obtained with NetconfLock.

A client is not permitted to unlock a configuration datastore that it did not lock. The administrator can use NetconfKillSession to terminate a lock aquired by any user.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfUnLock request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfCloseSession

NetconfCloseSession terminates an active session gracefully. Any pending requests are allowed to complete. Upon termination, any active locks are released.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfCloseSession request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfKillSession

NetconfKillSession is used by one session to kill another session. It can be used by the administrator to terminate any user sessions.

For example, a NETCONF session(A) can kill another NETCONF session(B) provided A has admin privileges or has FULL_CONTROL on SERVER. When a session is killed, any operations pending on it will be removed, and not executed by the Cisco E-DI server. Any locks held by the session will be released.However, if there is an operation currently being executed by Cisco E-DI, for example an edit-config, that operation cannot be stopped.Changes made to the candidate configuration will be removed. Changes to other config stores are always implicitly committed by Cisco E-DI.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetconfKillSession request.

For message ID information, see Note.

com.cisco.edi.ncclient.api.NetconfDeleteConfig

NetconfDeleteConfig deletes a configuration datastore. The <running> configuration datastore cannot be deleted. Only startup configuration can be deleted.

```
public static String getNetconfXMLRequestString()
public static String getNetconfXMLRequestString(String messageId){
```

returns NetConfDeleteConfig request string.

com.cisco.edi.ncclient.clienttool.session.XMLSession

This class provides higher level access to E-DI server to create a netconf session, sends requests, receives responses, and also provides session management methods.

```
public XmlSession()
```

`public XmlSession()` This constructor checks for log4j.properties. You need to set log4j.properties file using System.setProperty("`properties file path`").

```
public String openSession(StringBuffer helloFromEDI, int serverPort, String server, String
login, String password, int transportType , String keyFile , String propertiesFile) throws
ExceptionopenSession
```

method opens an XML session with E-DI.

`serverPort` is the EDI Telnet port. This should be 2323 for both Telnet and SSH.

`server` is E-DI server (IP or DNS name).

`login` is E-DI login.

`password` is E-DI password.

`transportType` 1 stands for Telnet, and 2 stands for SSH. If transportation type is SSH, any client machine can open only one SSH session as opensession does port forwarding. The response from the server listens at the 7777 port on the client side, since the client side port 7777 is open and engaged with the opened session.

`keyFile` Set this .keystore file path: *E-Di Install Location*\edi\resources\server\.keystore

This should set for SSH connection, other it should be null.

`propertiesFile`

`public void closeSession()`
`closeSession` method closes the current session.

`public String getSessionId()`
`getSessionId` method returns the current session ID.

`public boolean isConnected()`
`isConnected` returns the current session status.

`public String sendRequest(String xmlRequest) throws Exception`
`sendRequest` method sends the XML request to the Server and returns response.

`public String associateDevice(com.cisco.edi.ncclient.api.util.Device d) throws Exception`

`associateDevice` method associates the device with the current session.

`public String sendHello(String hello) throws Exception`

`sendHello` method sends Hello request.

For message ID information, see Note.Command Translator

com.cisco.edi.ncclient.api.util.Device

`public Device(IpAddress ipAddress,String login,String password) throws Exception`
`IpAddress` is device IP address.

`login` is the device username.

`password` is the device password.

`public Device(String name,String login,String password) throws Exception`
`name` The only valid value is **`server`**. Any other value will result in an Exception being thrown.

`login` is the device username.

`password` is the device password

`public String getNetconfXMLRequestString(String messageId)`
`getNetconfXMLRequestString` is the associate device request.

`public String getNetconfXMLRequestString()`

`getNetconfXMLRequestString` is the associate device request.
`public String getIPAddress() {`
`getIPAddress` method returns IP address of the device.

`void setIPAddress(String ip) throws Exception`
`setIPAddress` method sets the IP address for the device.

`public void setLogin(String login) throws Exception`
`setLogin` method sets the username for the device.

`public void setPassword(String password) throws Exception`

sets the password for the device.

For message ID information, see Note.