The main goal is to prove interoperability among several NETCONF Client and Server implementations. Implementations should be based on the following internet drafts:

- `draft-ietf-netconf-prot-06`
- `draft-ietf-netconf-ssh-04`
- `draft-ietf-netconf-beep-05`
- `draft-ietf-netconf-soap-05`

The testing itself is based on predefined scenarios and allows running ad-hoc test among different attendees.

**Test Environment**

The NETCONF protocol defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. In general different test scenarios are to be tested, having NETCONF Clients and multiple Servers connected to each other. The NETCONF managers send configuration data to the servers using different NETCONF functions. The Client which is referred as Manager and the Server as Agent are located on the same link as shown in the [Figure 1](#) below.

![Figure 1: NETCONF Test Scenario](image)

The NETCONF application may configure both devices (Agent1, Agent2 in the **Figure 1**) using different scenarios of NETCONF protocol. An additional FTP server
is connected to the network which contains pre-defined NETCONF XML files to be downloaded for testing by the NETCONF managers.

**Common NETCONF model to be used**

In order to have a common NETCONF data-model supported by all vendors to use NETCONF specific protocol operations and simulate different types of errors during the interoperability event, it is assumed that all vendors should support the following simple NETCONF data-model defined by the XML schema for network interface data configuration that is given below:

```xml
<xsd:attribute name="operation">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="merge"/>
      <xsd:enumeration value="create"/>
      <xsd:enumeration value="replace"/>
      <xsd:enumeration value="delete"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>

<xsd:element name="config">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="interfaces" minOccurs="0"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>

<!-- *************INTERFACE CONFIGURATION START************* -->
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="address" type="xsd:string"/>
<xsd:element name="netmask" type="xsd:string"/>
<xsd:element name="mtu" type="xsd:string"/>
<xsd:element name="bcast" type="xsd:string"/>
<xsd:element name="interfaces">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="iface" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="iface">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="address" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="netmask" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="bcast" minOccurs="0" maxOccurs="1"/>
      <xsd:element ref="mtu" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```xml
...and so on...
```
In case when the data-model defined above is not supported then empty NETCONF messages are to be used during testing as defined in the file “interop_xml.zip”. The test cases are also defined in “InteropEvent.xls” file.

**Basic Test Scenarios**

The basic test scenarios consist of:

1. SSH connection setup between the Manager and Agent. (SSH is preferred at this event as an application layer protocol. One can use BEEP or SOAP as well if supported by other vendors).

2. NETCONF Capability exchange.

3. `<get-config>`,
4. `<edit-config>`,
5. `<copy-config>`,
6. `<delete-config>`,
7. `<lock>`,
8. `<unlock>`,
9. `<get>`,
10. `<close-session>`
11. `<kill-session>`

12. Configuration of data stores such as “running”, “startup”, “candidate” upon availability.

13. malformed requests
14. RPC message before the hello message
15. manager m1 locks a configuration (running)
16. manager m2 tries to lock the same configuration (without success)
17. manager m1 unlocks the config.
18. manager m2 locks the config.
19. close-session should release the lock.

**Advanced Test Scenarios**

The advanced test scenarios include:
**Test Scenario 1.** Sub tree filtering for the network interface data-model defined above.

**Test Scenario 2.** Testing the NETCONF rollback capability on error for different types of simulated errors.

**Test Scenario 3.** This scenario is intended for testing the behavior of a single Agent which is configured by two Managers at the same time with different NETCONF protocol operations.

If you have any suggestion or test scenario please sent them to: cristian.cadar@netlab.nec.de