How to use SNMP4J-SMI with SNMP4J?

The integration of SNMP4J-SMI in SNMP4J is very easy and complete. If you create an `OID` or `Variable` value from a `String` value or converting them using the `toString()` or `format()` methods, SNMP4J-SMI will parse and format the values as human readable object names and value combinations on-the-fly.

SNMP4J Maven Repository

To include SNMP4J-SMI-PRO in your Maven build, you need to add the SNMP4J Maven repository to your .m2/settings.xml file as shown below.

Maven Repository Definition

```
      <repositories>
        <repository>
          <id>snmp4j</id>
          <name>SNMP4J Releases</name>
          <releases>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
            <checksumPolicy>warn</checksumPolicy>
          </releases>
          <url>https://snmp.app/dist/release</url>
          <layout>default</layout>
        </repository>
      </repositories>
```

Maven Dependency for pom.xml

```
<dependency>
   <groupId>org.snmp4j.smi</groupId>
   <artifactId>snmp4j-smi-pro</artifactId>
   <version>1.9.13</version>
   <classifier>jar-with-dependencies</classifier>
 </dependency>
```

Integrating SNMP4J-SMI into SNMP4J

The code needed at minimum to use SNMP4J with SNMP4J-SMI OID and Variable value formatting and parsing support is:

```
static {
    SmiManager smiManager = new SmiManager("myLicenseKey", new File("myEmptyDirectory"));
    SNMP4JSettings.setOIDTextFormat(smiManager);
    SNMP4JSettings.setVariableTextFormat(smiManager);
    // If you need to disable full index formatting, then choose a different format below and uncomment the
line:
    // smiManager.setOidFormat(OIDFormat.ObjectNameAndDecodedIndex4RoundTrip);
 }
```

Compiling MIB specification text files

Sample code to compile textual MIB specifications into the SNMP4J-SMI repository directory:

```
File f = new File(fname);
if (f.exists() && f.canRead()) {
  File[] files;
  if (f.isDirectory()) {
    files = f.listFiles();
  }
  else {
    files = new File[1];
    files[0] = f;
  }
  // Compile MIB files without monitor (null), load them directly into the MIB repository in memory,
  // update existant MIB modules, and do not ignore syntax errors:
  List<CompilationResult> results = smiManager.compile(files, null, true, true, false);
  int ok = 0;
  for (CompilationResult result : results) {
    if (result.getSmiErrorList() == null) {
      ok += result.getModuleNames().size();
    }
  }
  out.println("" + f + " contains ");
  out.println(" " + ok + " syntactically correct MIB modules and");
  out.println(" " + (results.size() - ok) +
      " MIB modules with errors.");
  String lastFile = null;
  for (CompilationResult result : results) {
    List<SmiError> smiErrors = result.getSmiErrorList();
    String n = result.getFileName();
    n = n.substring(n.lastIndexOf('/') + 1);
    if ((lastFile == null) || (!lastFile.equals(n))) {
      out.println("------ " + n + " ------");
      lastFile = n;
    }
    if (smiErrors != null) {
      for (int j = 0; j < smiErrors.size(); j++) {
        SmiError error = smiErrors.get(j);
        String txt = n + " #" + (j + 1) + ": " + error.getMessage();
        out.println(txt);
      }
    }
    else {
      String txt = n + ": " + "OK";
      out.println(txt);
    }
  }
}
else {
  out.println("Cannot access MIB file '" + fname + "'");
}
```

If you specify more than one MIB file, the MIB modules in those files will be automatically processed in the order of their import dependencies This avoids manual sorting or syntax errors because of unresolved IMPORTs.

Loading MIB Modules

Sample code to load (already compiled) a MIB module from the MIB repository directory:

```
smiManager.loadModule("SNMP-VIEW-BASED-ACM-MIB");
smiManager.loadModule("SNMPv2-SMI");
```

The SmiManager will also load all MIB objects that are imported by the specified modules! You do not have to implement this logic yourself

Formatting and Parsing Examples for Object Identifiers (OIDs)

```
// The classic use case, formatting OIDs using (last) object name and numeric (index) suffix:
OID testPrivOID = new OID("1.3.6.1.4.1.4976");
assertEquals("enterprises.4976", testPrivOID.toString());

OID vacmAccessContextMatch = new OID("1.3.6.1.6.3.16.1.4.1.4.7.118.51.103.114.111.117.112.0.3.1");
assertEquals("vacmAccessContextMatch.\"v3group\".\"\".3.'noAuthNoPriv(1)'", vacmAccessContextMatch.toString());

// If the selected OID format supports round-trip processing like this SmiManager.OIDFormat.
ObjectNameAndDecodedIndex4RoundTrip format
// OID.toString() and OID.format() will return the same value. Otherwise, OID.toString() will render the OID in
the next less formatting
// OID format that supports parsing:
assertEquals("vacmAccessContextMatch.\"v3group\".\"\".3.'noAuthNoPriv(1)'", vacmAccessContextMatch.format());

// Parsing of textual OIDs is supported out-of-the-box:
assertEquals(vacmAccessContextMatch, new OID("vacmAccessContextMatch.\"v3group\".\"\".3.'noAuthNoPriv(1)'"));
```

Formatting and Parsing Examples for Variable Bindings (VBs)

```
// store current format
SmiManager.OctetStringDefaultFormat defaultFormat = smiManager.getOctetStringDisplayHint();
// set the format to use MIB defined DISPLAY-HINT for OCTET STRING types
smiManager.setOctetStringDisplayHint(SmiManager.OctetStringDefaultFormat.MIB);

VariableBinding vbEnum = new VariableBinding(new OID("ifAdminStatus.4"), "down(2)");
assertEquals(new VariableBinding(new OID(new int[] { 1,3,6,1,2,1,2,2,1,7,4 }), new Integer32(2)), vbEnum);
assertEquals("down(2)", vbEnum.toValueString());

VariableBinding vbDateAndTime = new VariableBinding(new OID("nlmLogDateAndTime.1"),"2015-10-13,12:45:53.8,+2:
0");
assertEquals(new VariableBinding(new OID(new int[] { 1,3,6,1,2,1,92,1,3,1,1,3,1 }), OctetString.fromHexString
("07:df:0a:0d:0c:2d:35:08:2b:02:00")), vbDateAndTime);

// restore previous format
smiManager.setOctetStringDisplayHint(defaultFormat);
```