

How to use SNMP4J-AgentJMX with existing MBeans?

SNMP4J-AgentJMX is an API that works on top of SNMP4J-Agent and SNMP4J to provide support for mapping content of an MBean server to a SNMP4J-Agent based SNMP agent.

To map MBeans of a server to SNMP, the following steps are necessary:

1. Define or download a MIB. *If you need to create one, use [MIB Designer](#) because you will have to redesign your MIB often in an iterative process. Using a visual tool which is capable of undo/redo and moving/refactoring MIB nodes saves many hours of work and bug search.*
2. Create the SNMP4J-Agent object initialization for MOTables & MOScalars to hold the MIB structure. This step is called "creating the MIB stubs". This can be done automated for SNMP4J-Agent by [AgenPro](#) (code generation from the MIB).
3. Create a your own MOFactory (`myJmxMoFactory`) based on [JMXDefaultMOFactory](#) that actually creates SNMP4J-AgentJMX based ManagedObjects (MOScalarJMX and MOTOtableJMX).
Subclass the Mib class generated by AgenPro from the MIB and call (see also the `addJvmManagementMibInstrumentation` method of the sample MIB instrumentation class `JvmManagementMibInst`):

```
JMXDefaultMOFactory jmxFactory =
    new JMXDefaultMOFactory(server, scalarSupport);
// create MOs with factory
createMO(jmxFactory);
```

4. Statically map the OIDs of the MIB (generated in the Mib class by AgenPro) to the MBean names of the MBean server:

```
private static final Object[][] SCALAR_MBEANS_JVM_RUNTIME = {
    { JvmManagementMib.oidJvmRTName, "Name", String.class },
    { JvmManagementMib.oidJvmRTVMName, "VmName", String.class },
    { JvmManagementMib.oidJvmRTVMVendor, "VmVendor", String.class },
    { JvmManagementMib.oidJvmRTVMVersion, "VmVersion", String.class },
    { JvmManagementMib.oidJvmRTSpecName, "SpecName", String.class },
    { JvmManagementMib.oidJvmRTSpecVendor, "SpecVendor", String.class },
    { JvmManagementMib.oidJvmRTSpecVersion, "SpecVersion", String.class },
    { JvmManagementMib.oidJvmRTManagementSpecVersion, "ManagementSpecVersion",
String.class },
    { JvmManagementMib.oidJvmRTBootClassPathSupport,
new InverseBooleanType("BootClassPathSupported") },
    { JvmManagementMib.oidJvmRTInputArgsCount, "InputArguments", Long.class },
    { JvmManagementMib.oidJvmRTUptimeMs, "Uptime", Long.class },
    { JvmManagementMib.oidJvmRTStartTimeMs, "StartTime", Long.class }
};
```

5. Bind the mapping to an MBean:

```
final MBeanServerConnection server =
    ManagementFactory.getPlatformMBeanServer();

final MBeanAttributeMOTableSupport tableSupport =
    new MBeanAttributeMOTableSupport(server);
final MBeanAttributeMOScalarSupport scalarSupport =
    new MBeanAttributeMOScalarSupport(server);

ObjectName onameJvmRT =
    new ObjectName(ManagementFactory.RUNTIME_MXBEAN_NAME);

scalarSupport.addAll(onameJvmRT, SCALAR_MBEANS_JVM_RUNTIME);
```

6. That's all for scalars. For tables there is more to do. Of course, you also need to setup the SNMP agent with its security, listen ports, etc. See the `JMXTestAgent` for a simple example of that.