# Implement instrumentation code for the generated code using the SNMP4J-Agent API

This page provides concrete MIB instrumentation sample implementations based on the SNMP4J-AGENT-TUTORIAL-MIB.

## Instrumentation using Virtual Tables

With SNMP4J-AgentX's `BufferedMOTableModel`, you can easily instrument tables based on a virtual table technology. That means, the table is (usually) never hold in with all its rows in memory. Instead, only those rows are fetched from the underlying data source that are actually needed and requested by the currently running requests on the table.

1. Create your own sub-class implementation of `DefaultMOFactory` in order to be able to create/use your own `BufferedMOTableModel` instead of the default table model generated by AgenPro:

```
package org.snmp4j.agent.tutorial;

import org.snmp4j.agent.mo.*;
import org.snmp4j.agent.tutorial.impl.Snmp4JAgentTutorialFileTreeBUModel;
import org.snmp4j.smi.OID;

/**
 * The {@link Snmp4jAgentTutorialFactory} implements the {@link org.snmp4j.agent.mo.MOFactory} interface
to
 * create the instrumentation implementation for the {@link Snmp4jAgentTutorialMib}.
 *
 * @author Frank Fock
 */
public class Snmp4jAgentTutorialFactory extends DefaultMOFactory {

    @Override
    public <R extends MOTableRow, M extends MOTableModel<? extends R>> M
        createTableModel(OID tableOID, MOTableIndex indexDef, MOColumn[] columns) {
        if (Snmp4jAgentTutorialMib.oidSnmp4jAgentTutorialFileTreeBUEntry.equals(tableOID)) {
            return (M) new Snmp4JAgentTutorialFileTreeBUModel(tableOID, indexDef, columns);
        }
        return super.createTableModel(tableOID, indexDef, columns);
    }
}
```

2. In the generated `Agent.java` class file, change the getFactory() method to:

```
protected MOFactory getFactory() {
  return new Snmp4jAgentTutorialFactory();
}
```

3. In the generated `Agent.java` class file, add the following code starting the "link related MIB objects" comment to the `registerMIBs()` method in order to let the virtual table model use the associated scalar to determine the root directory for the file tree browsing done by the model:

```
/**
 * Register your own MIB modules in the specified context of the agent.
 * The {@link MOFactory} provided to the <code>Modules</code> constructor
 * is returned by {@link #getFactory()}.
 */
protected void registerMIBs()
{
  if (modules == null) {
    modules = new Modules(getFactory());
  }
  try {
    modules.registerMOs(server, null);
  }
  catch (DuplicateRegistrationException drex) {
    logger.error("Duplicate registration: "+drex.getMessage()+"."+
                 " MIB object registration may be incomplete!", drex);
  }
  // link related MIB objects
  ((Snmp4JAgentTutorialFileTreeBUModel)modules.getSnmp4jAgentTutorialMib().
          getSnmp4jAgentTutorialFileTreeBUEntry().getModel()).
          setRootPathScalar(modules.getSnmp4jAgentTutorialMib().
getSnmp4jAgentTutorialFileTreeBURootPath());
}
```

4. Implement your subclass of the `BufferedMOTableModel` or `BufferedMOMutableTableModel`. See the attached file Snmp4JAgentTutorialFileTreeBUModel.java for an example.

## Related articles

- How-to create your own SNMP agent with AGENTPP tools?
- Implement instrumentation code for the generated code using the SNMP4J-Agent API