When and how to map a MBean to a SNMP table?

Generally, every MBean that is not a singleton, should be mapped (or modeled) to (as) a SNMP table. MBeans that contain lists of other MBeans or Java objects should be mapped to several SNMP tables that share a common base index (i.e., the index of the master MBean/table).

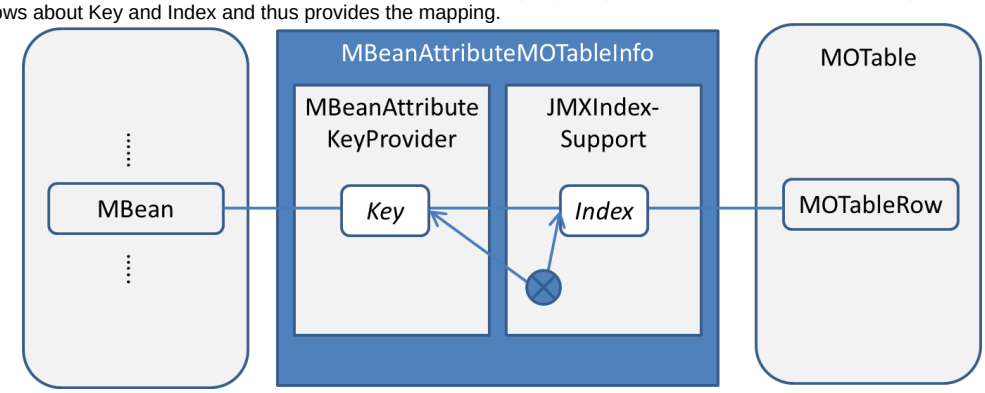
The following steps describe how a MBean can be descriptively mapped to a table:

1. First of all, you need a JMXTableSupport instance which is registered with the MBean server:

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

final MBeanAttributeMOTableSupport tableSupport =
    new MBeanAttributeMOTableSupport(server);
```

2. Next you need to define how the index of the SNMP table is mapped from MBean data or autogenerated. The mapping is done bidirectional by mapping a MBean to a key uniquely identifying the MBean and an Index (OID) uniquely defining the SNMP table row(MOTableRow). The JMXIndexSupport knows about Key and Index and thus provides the mapping.



```
tableSupport.add(oidJvmThreadInstanceEntry,
          new MBeanAttributeMOTableInfo(onameJvmThreading,

// The key provider retrieves all available keys (thus all available MBean instances:
          new MBeanInvokationKeyProvider(onameJvmThreading,
                                         new TypedAttribute("AllThreadIds", long.class),
                                         "getThreadInfo", true),
...
          new String[] { "ThreadId" },

// The JMX index support maps from rowIdentidier (= Key) to MBean objectname and ...
          new JMXIndexSupport() {
        public ObjectName mapToRowMBean(Object rowIdentifier) {
          return null;
        }

// from rowIdentfier to index OID and ...
        public OID mapToIndex(Object rowIdentifier) {
          Long l = (Long)rowIdentifier;
          return OctetString.fromHexString(Long.toHexString(l)).toSubIndex(true);
        }

// maps native row ID (key used by MBean) or index to rowIdentifier ...
        public Object getRowIdentifier(Object nativeRowId, int nativeIndex) {
          return nativeRowId;
        }

// maps index OID to rowIdentifier (= Key)
        public Object mapToRowIdentifier(OID rowIndex) {
          if (rowIndex == null) {
            return null;
          }
          OctetString os = new OctetString();
          os.fromSubIndex(rowIndex, true);
          String hexString = os.toHexString();
          return Long.parseLong(hexString, 16);
        }
      }));
```

3. Next, each column of the SNMP table needs to be mapped to a MBean attribute or action:

```
tableSupport.add(oidJvmThreadInstanceEntry,
          new MBeanAttributeMOTableInfo(onameJvmThreading,
          new MBeanInvokationKeyProvider(onameJvmThreading,
                                         new TypedAttribute("AllThreadIds", long.class),
                                         "getThreadInfo", true),
          new TypedAttribute[] {
          new TypedCompositeDataAttribute(new TypedAttribute("threadId", Long.class)),
          new CombinedBitsType(new TypedAttribute[] {
                               new EnumBitsType("threadState", Thread.State.class, Thread.State.values(),
3),
                               new BooleanBitsType("inNative", 1),
                               new BooleanBitsType("suspended", 2)}),
          new TypedCompositeDataAttribute(new TypedAttribute("blockedCount", Long.class)),
          new TypedCompositeDataAttribute(new TypedAttribute("blockedTime", Long.class)),
          new TypedCompositeDataAttribute(new TypedAttribute("waitedCount", Long.class)),
          new TypedCompositeDataAttribute(new TypedAttribute("waitedTime", Long.class)),
          new MBeanProxyType(server, onameJvmThreading, Long.class,
                             "getThreadUserTime",
                             new TypedCompositeDataAttribute(new TypedAttribute("threadId", long.
class))) {
            public Object transformFromNative(Object nativeValue, ObjectName objectName) {
              Long result = (Long) super.transformFromNative(nativeValue, objectName);
              if ((result == null) || (result < 0)) {
                return 0L;
              }
              return result;
            }
          },
          new TypedCompositeDataAttribute(new TypedAttribute("threadName", String.class)),
          new TypedCompositeDataAttribute(new TypedAttribute("lockOwnerName", String.class)),
          new TypedCompositeDataAttribute(new TypedAttribute("lockOwnerId", Long.class)) {
            public Object transformFromNative(Object nativeValue, ObjectName objectName) {
              Long result = (Long)super.transformFromNative(nativeValue, objectName);
              if ((result == null) || (result < 0)) {
                return "0.0";
              }
              OID rowPointer = new OID(JvmManagementMib.oidJvmThreadInstanceEntry);
              rowPointer.append(JvmManagementMib.colJvmThreadInstId);
              String index = Long.toHexString(result);
              OctetString os = OctetString.fromHexString(index);
              rowPointer.append(os.toSubIndex(true));
              return rowPointer.toString();
            }
          }},
          new String[] { "ThreadId" },
          new JMXIndexSupport() {
        public ObjectName mapToRowMBean(Object rowIdentifier) {
          return null;
        }

        public Object getRowIdentifier(Object nativeRowId, int nativeIndex) {
          return nativeRowId;
        }

        public OID mapToIndex(Object rowIdentifier) {
          Long l = (Long)rowIdentifier;
          return OctetString.fromHexString(Long.toHexString(l)).toSubIndex(true);
        }

        public Object mapToRowIdentifier(OID rowIndex) {
          if (rowIndex == null) {
            return null;
          }
          OctetString os = new OctetString();
          os.fromSubIndex(rowIndex, true);
          String hexString = os.toHexString();
          return Long.parseLong(hexString, 16);
        }
}));
```

4. Finally, the table model of the MOTable has been set to the JMXTableModel bound to the table support and table OID created in the previous steps:

```
JMXTableModel jvmMemMgrPoolRelEntryModel =
    JMXTableModel.getDefaultInstance(oidJvmMemMgrPoolRelEntry,
                          tableSupport,
                          super.getJvmMemMgrPoolRelEntry().getColumns());
((MOTableJMX)super.getJvmMemMgrPoolRelEntry()).
    setModel(jvmMemMgrPoolRelEntryModel);
```