How and why to migrate from SNMP4J-Agent DefaultMOPersistenceProvider to MOXodusPersitenceProvider?

Why Migrating to MOXodusPersitenceProvider?

- No data loss, any change to a `RandomAccessManagedObject` is written to disk immediately during COMMIT phase.
- Faster agent shutdown.
- Less disk-space usage (save more than 50%).

Migration Tasks

In your main agent code, first add the member moXodusPersistenceProvider and initialise it as follows:

```
public class SampleAgent implements AgentStateListener<AgentConfigManager> { // <--- UPDATE THIS LINE
...
    private MOXodusPersistence moXodusPersistence;   // <--- ADD THIS LINE
    protected AgentConfigManager agent;
...
}
```

```
// Enclose the following code block (most likely part of the constructor) that loads the default configuration
with the following if-statement:

        // load initial configuration from properties file only if there is no persistent data for the default
context:
        moXodusPersistence = new MOXodusPersistence(moServers, Environments.newInstance(configFile));  // <---
ADD THIS LINE
        if (!moXodusPersistence.isContextLoadable(null)) {                                              // <---
ADD THIS LINE
            InputStream configInputStream =
                    SampleAgent.class.getResourceAsStream("SampleAgentConfig.properties");
            if (args.containsKey("cfg")) {
                String configFilename = (String) ArgumentParser.getValue(args, "cfg", 0);
                try {
                    configInputStream = new FileInputStream(configFilename);
                } catch (FileNotFoundException ex1) {
                    logger.error("Config file '" + configFilename + "' not found: " + ex1.getMessage(), ex1);
                    throw new RuntimeException(ex1);
                }
            }
            final Properties props = new Properties();
            try {
                props.load(configInputStream);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            configurationFactory = () -> new PropertyMOInput(props, agent);
        }
```

The AgentConfigManager call will be the same as before except that the `moXodusPersistenceProvider` is used now:

```
        MOXodusPersistenceProvider moXodusPersistenceProvider = new MOXodusPersistenceProvider
(moXodusPersistence); // <--- ADD THIS LINE
        OctetString defaultEngineID = new OctetString(MPv3.createLocalEngineID());
        OctetString engineID = moXodusPersistenceProvider.getEngineId(defaultEngineID);
        SnmpConfigurator snmpConfigurator = new SnmpConfigurator(true);
        agent = new AgentConfigManager(
                engineID,
                messageDispatcher,
                null,
                moServers,
                ThreadPool.create("SampleAgent", 3),
                configurationFactory,
                moXodusPersistenceProvider, moXodusPersistenceProvider, null, dhKickstartParameters) { // <---
UPDATE THIS LINE

            @Override
            protected Session createSnmpSession(MessageDispatcher dispatcher) {
                Session session = super.createSnmpSession(dispatcher);
                snmpConfigurator.configure(session, getUsm(), messageDispatcher, args);
                return session;
            }
        };
        agent.addAgentStateListener(this);
```

The following callback method needs to be added to the main class in order to be able to compile the last code line from the above snippet:

```
    /**
     * The agent state has changed to the new state as provided.
     *
     * @param agentConfigManager
     *         the agent's configuration manager. Use this object to access all agent resources, if needed to
process this
     *         event.
     * @param newState
     *         the new state of the agent. Although the listener may advance to agent state further, it is not
recommended
     *         to do so, because the {@link AgentConfigManager} will do it anyway.
     */
    @Override
    public void agentStateChanged(AgentConfigManager agentConfigManager, AgentState newState) {
        switch (newState.getState()) {
            case AgentState.STATE_INITIALIZED:
                moXodusPersistence.registerChangeListenersWithServer(server);
                break;
            case AgentState.STATE_SHUTDOWN:
                moXodusPersistence.unregisterChangeListenersWithServer(server);
                break;
        }
    }
```

That's all.

Related articles

- How and why to migrate from SNMP4J-Agent DefaultMOPersistenceProvider to MOXodusPersitenceProvider?
- How to migrate from SNMP4J-Agent 2.x to 3.x?