How to configure SNMPv3 users with same name but different passphrases?

If a SNMPv3 security name is used by different agents (comand responders), it is necessary to **localize the keys** (passphrases) per agent. Key localization needs the authoritative engine ID of the target agent for the computation of the localized keys.

If agents do not use static engine IDs (which is **not** conforming to the SNMPv3 standard), this approach is not feasible because the engine ID might have changed between computation. In this case, a **separate USM instance** is required for each such agent.

The code snippets below illustrate these two approaches:

Key Localization

```
OctetString sharedUserName = new OctetString("sharedUser");
Target[] targets = <some SNMPv3 SecureTarget instances>
DefaultUdpTransportMapping transport = new DefaultUdpTransportMapping();
Snmp snmp = new Snmp(transport);
snmp.listen();

// discover engine IDs and add localized users
SecurityProtocols secProtocols = SecurityProtocols.getInstance();
for (int i=0; i<targets.length; i++) {
  Target t = targets[i];
  byte[] engineID = snmp.discoverAuthoritativeEngineID(t.getAddress(), t.getTimeout());
  OctetString authKey = new OctetString("md5Passphrase");
  authKey = securityProtocols.passwordToKey(AuthMD5.getID(), authKey, engineID.getValue());
  OctetString privKey = new OctetString("desPassphrase");
  privKey = securityProtocols.passwordToKey(PrivDES.getID(), AuthMD5.getID(), privKey, egineID.getValue());
  snmp.getUSM().addLocalizedUser(engineID, sharedUserName,
    AuthMD5.getID(), authKey,
    PrivDES.getID(), privKey);
}
```

USM Separation

```
SecurityProtocols.getInstance().addDefaultProtocols();

DefaultUdpTransportMapping transportCluster1 =
  new DefaultUdpTransportMapping(new UdpAddress("0.0.0.0/0"));
DefaultUdpTransportMapping transportCluster2 =
  new DefaultUdpTransportMapping(new UdpAddress("0.0.0.0/0"));

MessageDispatcher dispCluster1 = new MessageDispatcherImpl();
MessageDispatcher dispCluster2 = new MessageDispatcherImpl();

Snmp snmpCluster1 = new Snmp(dispCluster1, transportCluster1);
Snmp snmpCluster2 = new Snmp(dispCluster2, transportCluster1);

dispCluster1.addMessageProcessingModel(new MPv1());
dispCluster2.addMessageProcessingModel(new MPv2c());
localEngineID1 = new OctetString(
        MPv3.createLocalEngineID(new OctetString("Cluster1"+
                                                 System.currentTimeMillis())));
localEngineID2 = new OctetString(
        MPv3.createLocalEngineID(new OctetString("Cluster2"+
                                                 System.currentTimeMillis())));

USM usmCluster1 = new USM(SecurityProtocols.getInstance(), localEngineID1, 0);
USM usmCluster2 = new USM(SecurityProtocols.getInstance(), localEngineID2, 0);

dispCluster1.addMessageProcessingModel(new MPv3(usmCluster1));
dispCluster2.addMessageProcessingModel(new MPv3(usmCluster2));

snmpCluster1.getUSM().addUser(sharedUserName,
    AuthMD5.getID(), new OctetString("md5Passphrase"),
    PrivDES.getID(), new OctetString("desPassphrase"));
snmpCluster2.getUSM().addUser(sharedUserName,
    AuthMD5.getID(), new OctetString("md5Passphrase"),
    PrivDES.getID(), new OctetString("desPassphrase"));

transportCluster1.listen();
transportCluster2.listen();
```