OpenArm Transaction SnmpMediator

1. SnmpMediator

This class outputs the information for a given transaction as SNMP v.2c traps.

2. SnmpMediator Properties

Regardless of **how** it is configured, the SnmpMediator has the following properties, which must be set:

- *managementHost* : the managment host that traps should be sent to
- *managementHostTrapListenPort* : the port that the managment host listens for traps on
- *enterpriseOID* : the enterprise OID that define your company's/application's traps
- *applicationTrapOIDValue* : the application trap OID that defines traps sent by your application
- *transactionLengthInterval* : the time interval that you want as a threshold that must be exceeded before a trap is sent
- *updateLengthInterval* : the same sort of threshold for updates
- *isBlockedEventTrap* : a boolean flag that determines whether or not blocking events should be forwarded as traps

3. Dependency Injection Configuration

This class is accompanied by a so-called Mediator Configuration class -- SnmpMediatorConfiguration -- which you can use to configure the Mediator programmatically. The properties of the configuration class correspond to the properties of the Mediator. Here is an example:

```
return new OpenArmConfiguration() {
    public Map getMediatorConfigurations() {
        final Map result = new HashMap();
        result.put("net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator
                new SnmpMediatorConfiguration("127.0.0.1", //management host
                162, // management host trap listen port
                "1.3.6.1.2.1.2.0", // enterprise OID
                "1.3.6.1.2.1.2.0.0.0.42", // application trap OID
                2000, // transaction length interval
                10000, // update length interval
                true)); // blocked event trap flag
```

Copyright © 2005 M2 Technologies All rights reserved.

```
        return result;
    }
};
```

The sample above also shows the default values that this class will use for each of these attributes, if you do not provide a properties file, or if you omit one or more of the attributes.

4. Declarative Configuration

If you need to use the (deprecated) file-based configuration mechanism, you should provide a properties file for this class within its classpath/classloader scope. The name of the properties file must be the value of the PROPERTIES_FILE_NAME constant defined in this class. Within this properties file, you should define the 7 properties listed above.

These properties must take the following form: *[standard prefix].[attribute name]=[value]* , where the *[standard prefix]* element must be **net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator**

A sample properties file might therefore look like the following:

```
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.man
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.man
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.ent
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.app
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.tra
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.upd
                net.m2technologies.open_arm.transport.transaction.snmp.SnmpMediator.isB
```

The sample above also shows the default values that this class will use for each of these attributes, if you do not provide a properties file, or if you omit one or more of the attributes.

Copyright © 2005 M2 Technologies All rights reserved.