JMX Tools

by SuperBonBon

1. UPNP# Services JMX management interface

We provide an JMX management interface with the *net.sbbi.upnp.jmx.JMXManager* class. The console is an HTTP interface based on the MX4J project. You can try it to see the *net.sbbi.upnp.jmx.UPNPServiceMBean* class in action. You'll need an UPNP# Device on your network to see UPNP# devices exposed as JMX MBean in the http console. The console detects when new devices are joining or leaving the network and automatically register/unregister these devices as MBeans.

We provide a script called *UPNPLib* in the bin distribution directory to start the console on port 7070. You can call this link to access it with username/password *admin/admin* once the script is started. You'll have to edit the *bin/bootstrap.xml* startup config file *JMXConsole* entry to modify user names and server port.

2. UPNP# Services as JMX MBeans

UPNP# Services can be exposed as JMX MBeans with the *net.sbbi.upnp.jmx.UPNPServiceMBean* class. The MBean can be used to call device service operations and state variables transparently. You can also use resource bundles with the MBean to provide informations about operations names, input parameters and state variables. Take a look at the the javadoc for more info about it.

Here is a small example to create and expose an UPNP# device service as an MBean :

```
UPNPRootDevice rootDevice = net.sbbi.upnp.Discovery.discover()[0];
//let's look at a root device specific child devices
String dvURN = "upnp:schemas-upnp-org:device:WANDevice";
UPNPDevice myIGDWANDevice = rootDevice.getChildDevice( dvURN );
if ( myIGDWANDevice != null ) {
  System.out.println( "IGD WAN device " + myIGDWANDevice.getUDN() );
  String srvURN = "upnp:schemas-upnp-org:service:WANIpConnection:1";
  UPNPService WANIpConnectionSrv = myIGDWANDevice.getService( srvURN );
  if ( WANIpConnectionSrv != null ) {
    System.out.println( "IGD WAN device WANIpConnection service " +
                        WANIpConnectionSrv.getServiceId() );
    // creating the MBean
```

```
      UPNPServiceMBean mBean = new UPNPServiceMBean( myIGDWANDevice,
                                                     WANIpConnectionSrv );
      // and creating an MBean Server and registering the bean you can
      // use mBean.getObjectName() to have an unique beans object name
      MBeanServer srv = MBeanServerFactory.createMBeanServer();
      srv.registerMBean( mBean, mBean.getObjectName() );
      // now invoke a method defined in the UPNP device service specs
      // ( upnp:schemas-upnp-org:service:WANIpConnection:1 here ).
      // We call here the GetExternalIPAddress method to retreive
      // the IP from the UPNP device.
      Map result = (Map)srv.invoke( mBean.getObjectName(),
                                    "GetExternalIPAddress",
                                    null, null );
      // result can be null if wrong params or an unknow param
      // name has been provided. Check the device specs if such
      // thing occurs
      if ( result != null ) {
        // the device returns the ip with the NewExternalIPAddress key
        // according to the device specs
        String ip = (String)result.get( "NewExternalIPAddress" );
        System.out.println( "Device IP is " + ip );
      }
    }
  }
```

3. UPNPDiscoveryMBean

This MBean allows you to discover devices on the network and register the devices services as UPNPServiceMBean :

```
import net.sbbi.upnp.jmx.*;
...
String name = "UPNPLib discovery:name=Discovery MBean";
ObjectName discBeanName = new ObjectName( name );
UPNPDiscoveryMBean bean = new UPNPDiscovery( discoveryTimeout, true, true );
server.registerMBean( bean, discBeanName );
```

The code is quite easy, you can also define to only monitor a set of specific devices :

```
import net.sbbi.upnp.jmx.*;
...
String name = "UPNPLib discovery:name=Discovery MBean";
ObjectName discBeanName = new ObjectName( name );
String type = "urn:schemas-upnp-org:device:WANDevice:1";
UPNPDiscoveryMBean bean = new UPNPDiscovery( type, discoveryTimeout, true, true );
server.registerMBean( bean, discBeanName );
```

Here only devices of type *urn:schemas-upnp-org:device:WANDevice:1* will be registered. You can take a look at the API docs for more infor about the class constructors and

behaviour

4. JMX UPNP Connector

You can expose your entire JMX interface as UPNP# devices via the standard JMX connector api. Here is a code sample to register the connector :

```
// simple URL define your host name and port
JMXServiceURL url = new JMXServiceURL( "service:jmx:upnp://localhost:8080 );
Map env = new HashMap();
// define the net.sbbi.upnp.jmx package for connector implementation
env.put( JMXConnectorServerFactory.PROTOCOL_PROVIDER_PACKAGES, "net.sbbi.upnp.jmx" );
// define a custom UPNP Mbeans Builder look at interface
// net.sbbi.upnp.jmx.upnp.UPNPMBeanBuilder javadoc for more info
env.put( UPNPConnectorServer.UPNP_MBEANS_BUILDER, new MyUPNPMBeanBuilderImpl() );
// setting to instruct the connector to lookup for all UPNP evices on the network
// and expose them as MBeans into the MBeans server, default to false
env.put( UPNPConnectorServer.EXPOSE_UPNP_DEVICES_AS_MBEANS, Boolean.TRUE );
// discovery timeout for the previ0ous setting
env.put( UPNPConnectorServer.EXPOSE_UPNP_DEVICES_AS_MBEANS_TIMEOUT, 2500 );
// setting to expose MBeans as UPNP devices
env.put( UPNPConnectorServer.EXPOSE_MBEANS_AS_UPNP_DEVICES, Boolean.TRUE );

// start the connector and look at the magic with an UPNP devices tool ;o)
JMXConnectorServer connectorServer = JMXConnectorServerFactory.newJMXConnectorServer(
upnpConnectorBeanName = "Connectors:protocol=upnp,host=localhost,port=8080;
ObjectName cntorServerName = ObjectName.getInstance( upnpConnectorBeanName );
server.registerMBean( connectorServer, cntorServerName );
connectorServer.start();
```

Once an MBean is registered it will be automatically exposed as an custom UPNP device. The device will be destroyed when the MBeans is unregistered from the MBeans server, take a look at the API docs for more infos.