第七周实习报告

5080309684 仇卓

任务概述

继续改进 pc 客户端，改善 UI 界面设备信息和服务的显示方式，完善 Action 调用和输出显示。

实现要点

服务器端部分没有修改

虚拟设备的修改：

之前虚拟设备中的 Action 是没有输出的，修改 Action，加入输出参数：

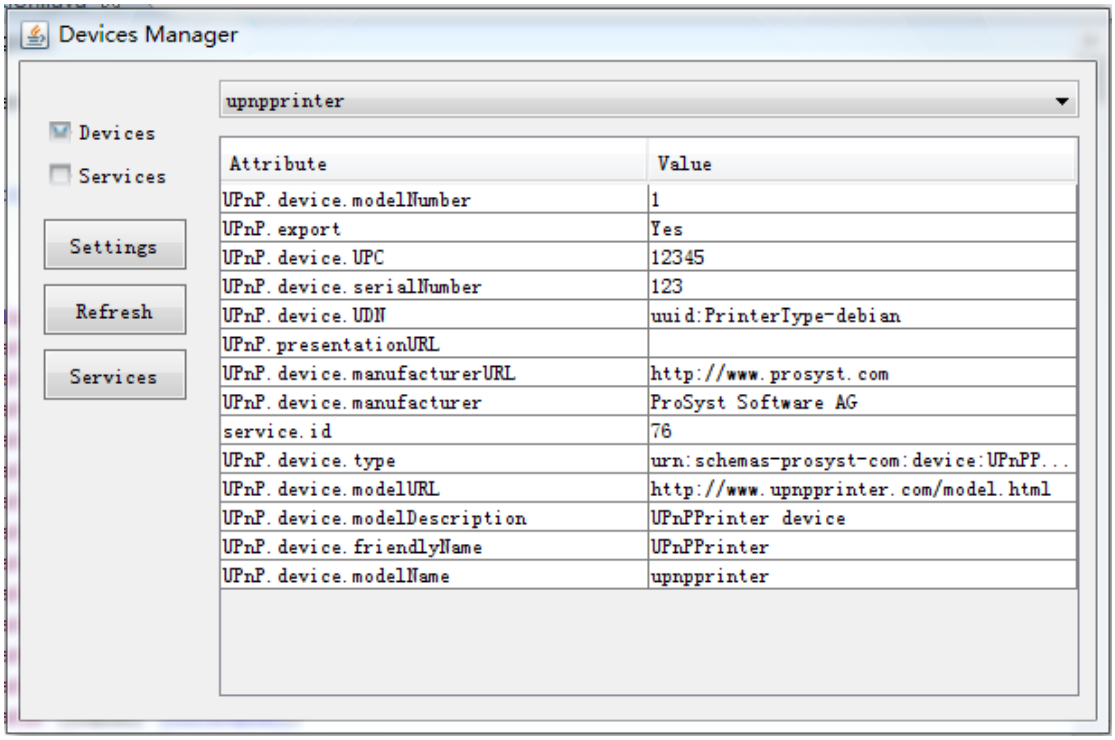
```
package upnpdevicetest;

import java.util.Dictionary;
import java.util.Hashtable;

import org.osgi.service.upnp.UPnPAction;
import org.osgi.service.upnp.UPnPStateVariable;


public class PrintAction implements UPnPAction {
	UPnPPrinterDevice dev;
	String name = "print";
	String[] iaNames;
	String[] oaNames;   // 输出参数
	Hashtable argSSV;

	public PrintAction(String[] iaNames, String[] oaNames, Hashtable argSSV,
			UPnPPrinterDevice dev) {
		this.iaNames = iaNames;
		this.oaNames = oaNames;
		this.argSSV = argSSV;
		this.dev = dev;
	}

	public String getName() {
		return name;
	}
```

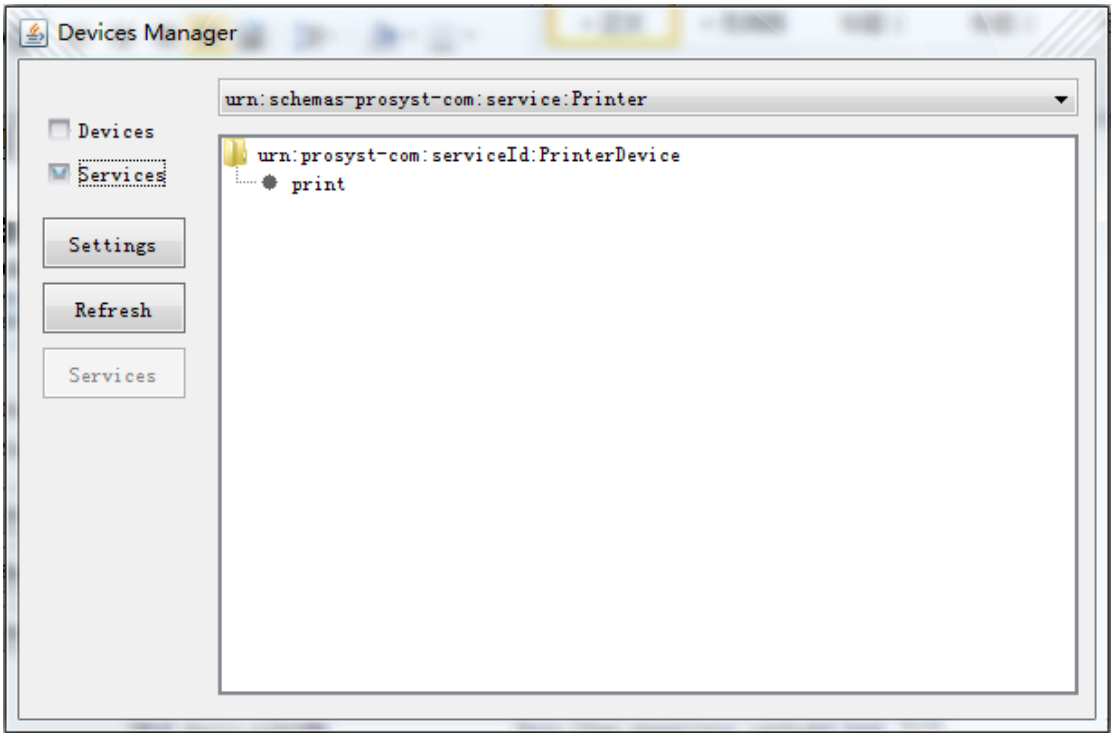
```
    public String getReturnArgumentName() {
        return null;
    }

    public String[] getInputArgumentNames() {
        return iaNames;
    }

    // 返回输出参数
    public String[] getOutputArgumentNames() {
        return oaNames;
    }

    public UPnPStateVariable getStateVariable(String argumentName) {
        return (UPnPStateVariable) argSSV.get(argumentName);
    }

    public Dictionary invoke(Dictionary d) throws Exception {
        String msg = (String) d.get(iaNames[0]);
        Hashtable outputs = new Hashtable();
        if (msg != null) {
            // send printing=true event
            Hashtable events = new Hashtable(2);
            events.put(dev.sv_Printing.getName(), Boolean.TRUE);
            dev.srv_Printer.generateEvent(events);
            // printing
            System.out.println("Printing: \n" + msg);
            try {
                Thread.currentThread();
                Thread.sleep(5000);
            } catch (Exception exc) {}

            // send printing=false event
            events.put(dev.sv_Printing.getName(), Boolean.FALSE);
            dev.srv_Printer.generateEvent(events);

            // 该输入参数 msg 加一个”out ”的前缀，加入输出参数
            outputs.put(oaNames[0], "out " + msg);
        }
        // 返回输出参数
        return outputs;
    }
}
```

修改模拟设备的代码，加入 action 的输出参数：

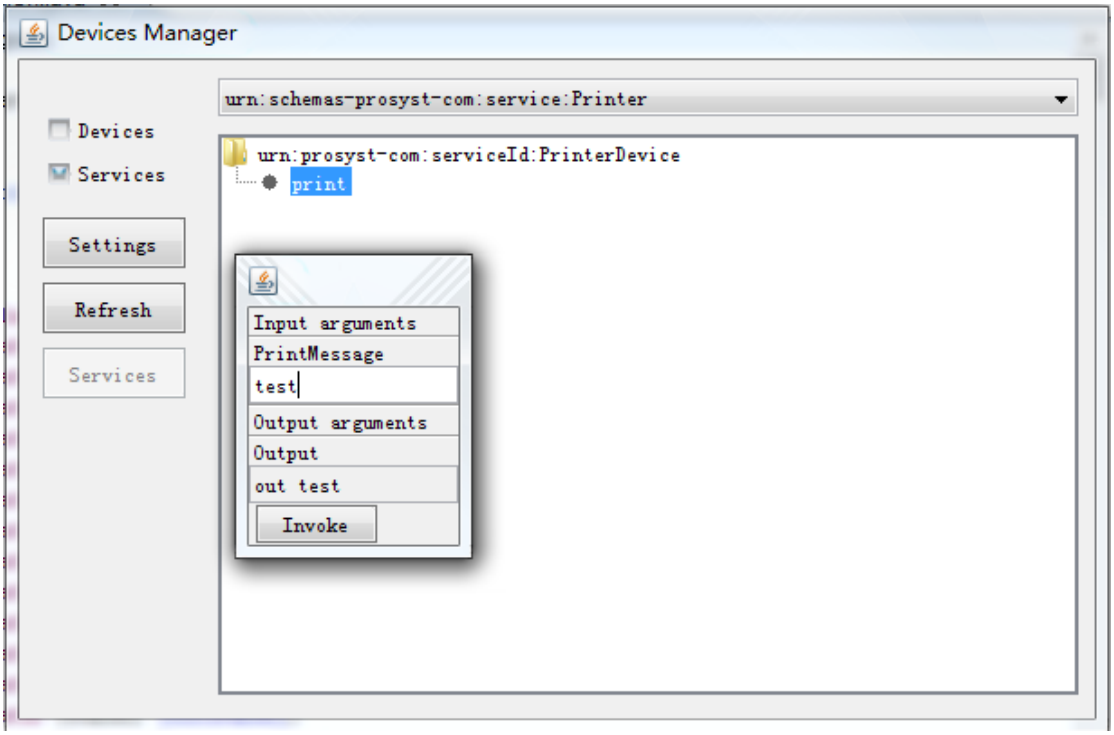
```
UPnPStateVariable sv_Output;   // 定义输出参数
sv_Output = new StateVarImpl("Output", String.class,
            UPnPStateVariable.TYPE_STRING, "", false);
// 将输出参数加入参数列表
upnpStateVars = new UPnPStateVariable[] {sv_Printing, sv_PrintMessage, sv_Output};

// init action Print
Hashtable nameVar = new Hashtable(2);
nameVar.put(sv_PrintMessage.getName(), sv_PrintMessage);
// 新建一个 action
ac_Print = new PrintAction(new String[]{sv_PrintMessage.getName()},
            new String[]{sv_Output.getName()},
            nameVar, this);
upnpActions = new UPnPAction[] {ac_Print};

// init Printer service
srv_Printer = new PrinterService(this);
upnpServices = new UPnPService[] {srv_Printer};
```

PC 端 GUI 的修改：





主界面设备信息的显示和服务层次的显示修改为使用下拉框来选择不同的设备或服务类型，避免了之前使用标签页显示时由于标签名字过长而导致的界面不美观以及使用上的不方便。其他界面的 UI 没有改动。



按照修改后的 action 代码，设备给传入参数”test”加上了”out ”前缀，使输出变为”out test”。

注意事项和问题

在模拟的设备中，调用 action 时，action 会先发送一个正在调用的事件：

```
Hashtable events = new Hashtable(2);
events.put(dev.sv_Printing.getName(), Boolean.TRUE);
dev.srv_Printer.generateEvent(events);
```

然后运行 action 的主要代码。

值得注意的一点是，执行完 action 的代码后，action 会睡眠 5 秒：

```
try {
    Thread.currentThread();
    Thread.sleep(5000);
} catch (Exception exc) {}
```

最后再发送调用完成的事件：

```
events.put(dev.sv_Printing.getName(), Boolean.FALSE);
dev.srv_Printer.generateEvent(events);
```

睡眠的代码来自于 prosyst 给出的示例代码，在我的测试中，若将时间改为 1 秒或直接不睡眠，那么对于 action 的调用只有第一次能够成功，之后的调用都会导致网络数据流传输的异常。我认为这个睡眠的时间是为了让网络中的其他设备知道这个 action 正在被调用，由于网络条件的不同，有可能某些设备获取该信息需要较长时间，若睡眠时间过短则可能导致出错。

参考资料

http://dz.prosyst.com/pdoc/mbserver_5.2/um/upnp/developer/osgi_upnp/osgi_upnp_export.html
Writing a UPnP™ Device with the OSGi API

http://dz.prosyst.com/pdoc/mbserver_5.1/Tutorial/upnp/bundles/upnp/upnp.html
UPnP Driver Bundle