Project P5 – A UPnP PowerPoint Controller

Report Due: March 23, 2005 (at 4:00 PM)

Part I – Objectives and Project Materials

Objectives

The objectives of the design project are to:

- develop intimate knowledge of the Universal Plug and Play (UPnP) discovery process;
- develop a UPnP device using C# and the Intel UPnP tools; and
- gain more experience with Ethereal, C#, and the .NET platform invoke.

After completing the assignment, you should be able to:

- document processes and protocols of UPnP; and
- use C# and the Intel UPnP tools to develop and test UPnP devices

Hardware

The following hardware is to be used in this lab assignment.

- Notebook computer
- iPAQ

Software

The following software is to be used in this lab assignment.

- Microsoft Windows XP on the notebook computer
- Intel UPnP tools and auth tools
- Microsoft Visual Studio .NET 2003
- UPnP Client (provided to you)
- Ethereal

Overview

The functional objective for this project is to have a working UPnP controller application. This application will allow remote control of a PowerPoint (PPT) presentation on a notebook computer (acting as a UPnP Device) using the iPAQ (acting as the UPnP control point). Communication will be via IEEE 802.11b operating in ad hoc mode. The control point will be able to:

- choose the presentation file;
- start and stop the presentation;
- advance to the next slide;
- go to the previous slide; and
- go to a specified slide.

You will be responsible for designing, implementing, and testing the UPnP device application that runs on the notebook computer. We provide you with the control point application that runs on the iPAQ. Intel's UPnP tools do not yet fully support the .NET Compact Framework. Consider yourself lucky for not having to build both the device and control point!

Part II – Design Project: UPnP PowerPoint Controller

1. Overview

For this project, you will design, implement, and test the UPnP device and services for the PPT controller. To make the project challenging and ensure that you have an understanding of how UPnP works, you will not be given all of the information that you were for the associated in-class laboratory. This means you will have to do a little "hacking" and exercise your analytical skills.

You will be given:

- the control point application for the iPAQ;
- the service name; and
- the PPT control DLL file and documentation.

The rest is up to you.

2. "Hacking" the Descriptions

To build the descriptions needed for Intel Device Builder you will need to rely upon the use of Ethereal and your knowledge of UPnP and Service Author. Traffic involved with UPnP is human-readable as it is in the form of XML content and HTTP messages. Use Ethereal to observe the packets between the control point and the service and collect the information that you need to determine the action names and which state variables are needed.

The following procedure is recommended. Begin by determining the device name for which the control point is looking. Once you have obtained the device name, use test devices to further discover what is required including the action names and the state variables. Use Ethereal and the test devices to deduce the information for which the control point is looking. Saving the settings in Device Builder will reduce redundant information entry.

Make sure to save the .xml description as you will need it for the report.

3. Creating the Descriptions and Building the Device

After you have found all of the needed actions and state variables, create a final service description. Use the following values.

- Service Name: P5_*nn*, where *nn* is your two-digit group number, e.g., 02, 10, etc.
- Service Type: urn:schemas-upnp-org:service:control:1
- Service ID: urn:upnp-org:serviceId:control

Once you have a service description that you know works, build the final device. For the device information, fill in the appropriate information in the Root Device Type and whatever you would like in the remaining sections. Set your version to 1.

4. Implementing the Actions

As you did in the associated in-class laboratory, comment out or remove the delegate assignments and the delegate functions in the SampleDevice.cs file; P5_<*group number*>.External_<*delegate function*> and the corresponding functions beneath. All of your coding should be done in the P5_<*group number*>.cs file.

5. The PPT Control DLL, PPTController.dll

PPTController.dll is written in unmanaged C++. Luckily, .NET includes the ability to make calls to legacy DLL files. To do so, research the PInvoke concept in MSDN documents.

The headers for the available functions in the DLL are as follows.

- `int PPT_Start(LPCSTR PPTFileName);`

  Opens PowerPoint (if it is not already open), opens the file named in PPTFileName, and starts the slide show. The returned value is the number of slides in the presentation. Upon error, an exception is thrown.

- `void PPT_Next();`

  Advances the slide show to the next slide.

- `void PPT_Previous();`

  Moves the slide show to the previous slide.

- `void PPT_Stop();`

  Halts the slide show and closes the open file. It does not close PowerPoint.

- `void PPT_Goto(long page);`

  Goes to the specified slide. If the desired slide does not exist (i.e., it is past the end of the presentation) the slide of the presentation should be shown (this is the last actual slide, not the black end-of-presentation slide generated by PowerPoint).

- `void PPT_InitInstance();`

  Initializes the DLL to allow correct operation. This must be called right before PPT_Start( ).

- `void PPT_ExitInstance();`

  This cleans up the DLL. It should be called right after PPT_Stop().

6. Requirements for Observed Client Operations

This section describes how the device should operate given input from the control point.

- Upon running the control point application, it will locate all matching devices and list them in the `Device` combo box.
- Clicking the `Files` button will result in a query to the device for the list of files that can be opened.
- Clicking the `Get Status` button updates the corresponding field. This field is also subscribed to an event that should change as the power status changes from “on” to “off.”
- The `Go To` button should advance the slide show to the specified slide. Sending a slide number less than or equal to 1 should return the slide show to the first slide. Sending a slide number that is beyond the slide length should advance the slide show to the last slide of the presentation. (Note that the last slide is not the black “End of slide show” display provided by PowerPoint.)
- The `Next` button should advance the slide by one each time it is clicked. Upon reaching the “End of slide show” slide, the presentation should no longer advance and should *not* return to the PowerPoint editing mode. Clicking the `Power` button is required to close the slide show.
- The `Previous` button should move the slide show backwards once each time it is clicked. Once at the first slide, it should have no effect.
- The `Power` button should open and start the specified slide show, if one is not already in progress. If a slide show is in progress clicking the `Power` should close the slide show. The `Power` button should also change the value seen in the `Get Status` information box.

- The `Refresh` button does a new query for desired devices and updates the `Device` box as necessary. This has no effect on the device.
- The `Demo` and `Diagnostic` buttons switch to a demonstration and diagnostic mode, respectively. Diagnostic mode may be used to watch what the client is doing for debugging purposes.

7. Notes on the Device Application

The device application should never throw an exception.

All PowerPoint files should be located in the C:\P5 directory. Files should be returned with just the file name and extension. The files should be returned in a single string with '|'s separating each name. Only .ppt files should be returned to the control point.

All state variables should have eventing turned on.

8. Hints and Other Useful Information

Breakpoints and the output of debugging information is always useful.

MSDN is the suggested resource for C# and .NET related questions.

Use the Project 5 discussion board for queries specific to this assignment, the provided control point application, and the provided PPT controller DLL.

9. Optional Features

You may implement the optional features to earn up to a maximum of 5 extra points that will compensate for points lost on other aspects of the graded report. Your maximum grade for the report cannot exceed 100. Note that an optional feature must function completely correctly, be tested during the project demonstration, and be documented in your report to get any points for it.

- **5 points** – Enable logging of all action invocations, parameters and the time/date to a file. Also include any errors generated by your application, and the stopping/starting of the log. Here is an example of how the log file might start.

```
1/24/2006 ( 16:28 ) – Log started
1/24/2006 ( 16:34 ) – GetStatus ( )
```

- **5 points** – Build a graphical user interface (GUI) front-end to your device application that displays debugging information. The GUI should include a single, scrollable window that displays the same information as the above logging option.

Part III – Deliverables and Report

Deliverables

You need to provide the following deliverables by the project due date.

- C# source files for the project (four files). All code you write should be fully documented.
- The service description file (the .xml file).
- A project report as described below. The report should be submitted as an MS Word or Adobe Acrobat PDF file.

Adhere to the guidelines provided earlier when preparing the report and when submitting files. All deliverables listed above must be submitted to Blackboard's Dropbox by the due date as a single .zip file with the name P5_*Parter1LastName_Partner2LastName*.zip.

Project Report

The project report should contain the following items in the order specified below. Use section headings to organize your report.

- A cover page specifying the course number and name, the project name (P5 – UPnP Device); the name, student identification number, and the email address of each team member; location (Blacksburg or NVC); and the date of submission.
- A listing and brief description of all files included in the deliverables and instructions on rebuilding the project (no more than one-half page).
- A brief discussion (no more than one page) of the steps taken in deriving the descriptions, building the application, and testing the application.
- Show a timeline of the UPnP discovery process. Include invocation of at least one action and one event. Include snippets from Ethereal when applicable. (Use no more than two pages for this item.)
- A list and brief discussion (no more than one page) of any optional features implemented and their operation.
- Answers to the following questions. Your answers should be concise, but complete. (The answer to all questions should take no more than two pages.)
    1. What are at least two differences between UPnP services and web services?
    2. From what you have seen and read concerning UPnP, what extensions, services, and functionality would you add to it?
    3. What problems did you encounter during the project? What steps did you take in resolving these problems?
    4. Are there any known problems or shortcomings with your project?
    5. What was your overall feeling of the project? Any thoughts/comments/suggestions you wish to add?
- The source code for any files that you *modified*. (Do not include source files that you did not change.)

Grading

Grading will be based on the following factors. The project will be graded based on a maximum score of 100 points.

- Correct operation as demonstrated to an instructor or GTA (70 points). Partial credit will be awarded for partial functionality. Be prepared to demonstrate all functionality that works, including optional features.
- Report content, including discussion of your approach and the above questions (25 points).
- Overall quality of the submission, including writing of the report, comments in code, etc. (5 points).
- Implementation of optional features can compensate for up to 5 points, depending on the features successfully implemented. The overall project grade cannot exceed 100 points.

Teams will be asked to demonstrate their project's operation to an instructor or GTA on or after March 21, 2006.

Part IV – Demonstration

All groups will need to meet with an instructor or GTA to demonstrate their working device application and have it tested. Demonstration sessions will be held in lieu of the Week 9 in-class lab session. Details will be announced later.

Part V – Honor Code

Each group must work on their own on this project. You may not share your code with anyone except your partner or an instructor or GTA for this class. You may not borrow code from anyone except your partner. You may not discuss your design or code with anyone except your partner, a GTA for this course, or an instructor for this course. You may not help other students, except your partner, in debugging code or have others, except your partner, help you. Simply stated, you may not discuss any aspect of your original work with anyone except your partner, a GTA for this course, and an instructor for this course. If you use libraries or any code developed by others, except the standard tools used in this class and the code provided with this assignment, its use must be properly acknowledged.

You may discuss the basic use of Microsoft Visual Studio and the Intel UPnP tools with other students. You may also discuss the requirements for this assignment with others. Contact an instructor if you have any questions about the honor code requirements.