



Intel Development Tools for Implementing UPnP* Devices

**Research &
Development
at Intel**

(Version 1.6, 03-10-2003)

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Copyright © Intel Corporation 2003

* UPnP is a certification mark of the UPnP Implementers Corp. Other names and brands may be claimed as the property of others.

Intel Development Tools for Implementing UPnP Devices

March 2003

Introduction

UPnP technology provides an effective foundation for networked devices that need simple, seamless interoperability for the home, office, and public domain. The UPnP device architecture enables connectivity of devices of all form factors. It provides networked devices with capabilities for automatic discovery, configuration, and control. It is a distributed networking architecture that leverages TCP/IP and other Internet technologies to enable flexible data communication between related devices under the command of UPnP control point entities. UPnP technology is independent of any particular operating system, programming language, or physical medium.

When product developers apply UPnP technology to

the network devices they create, it frees the user from involvement with complex issues like configuration, setup, maintenance, protocols, and software. All of these issues are dealt with behind-the-scenes and any associated tasks and processes are transparent. This technology enables non technical people to more easily install, use, upgrade, and maintain increasingly sophisticated products on their networks.

Until recently, designing and implementing a UPnP device required a considerable investment in writing service description files, building source code, testing, and debugging. Useful tools simply did not exist for automating any of the processes. The testing process was especially labor intensive, and debugging ineffective. Too often the end product was a compromise of features or quality, or both.

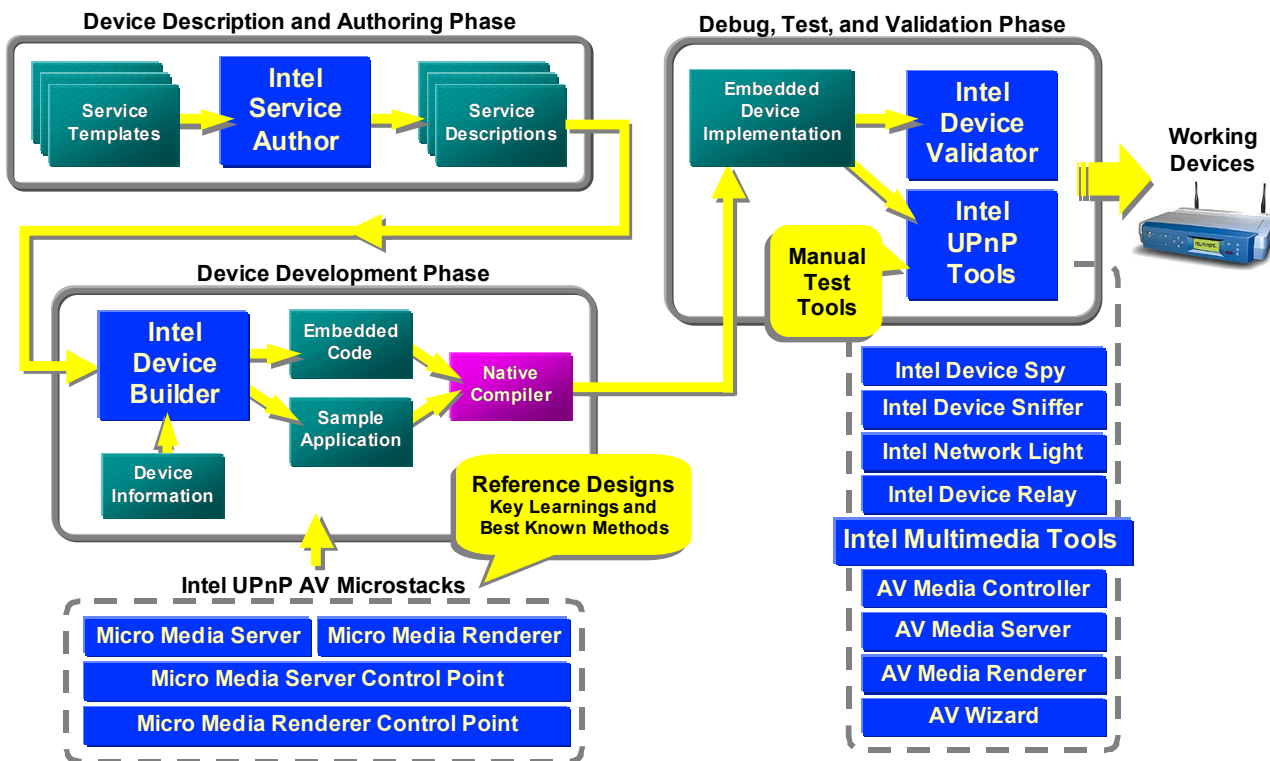


Figure 1. Intel Development Tools for UPnP Technology

About a year ago, the Solutions Architecture lab¹ (SA) at Intel began to work on ways to improve the process of designing and implementing UPnP capable devices. Having worked on various technologies for extending the PC in the home, including UPnP and UPnP AV technology, SA saw the need for development tools for all of the basic steps required to construct a UPnP device.

Applying Intel Development Tools for UPnP Technologies

This section describes, in general terms, how the development tools apply to the process of designing and implementing a UPnP capable device. Figure 1, located on page 1, illustrates this process, identifies the tools, and shows where they are used during device development.

Service Description Authoring Using Intel Service Author

Service description authoring is the process of creating service descriptions by customizing service templates (see Figure 2 located on page 3). A UPnP device has one or more service descriptions, which collectively describe the capabilities of the device.

Service Templates

Service Templates (STs), published by the UPnP Forum, represent a standard set of features for a particular service. Developers of UPnP devices select STs based on the requirements of their device. Often times, the UPnP Forum also defines a device template, which indicates the STs that logically combine to form a complete UPnP device.

¹A part of Desktop Platform Architecture, an organization within the Desktop Platform Group responsible for defining and developing platforms of the future.

A Service Template may include optional features that a device designer does not require for the product being developed. For example, an AV picture renderer does not support volume controls so the Get-and-Set Volume actions should be removed from the ST. In a few instances, the vendor may want additional non-ST specified features that logically go with the service.

Intel Service Author

Intel Service Author is designed to transform standard and proprietary STs into device specific Service Control Protocol Declarations (SCPDs), or Service Descriptions. Using Service Author's graphical user-interface (GUI), developers add or remove UPnP state variables and actions, and define custom UPnP actions, state variables, and arguments. The output of Service Author is a well formed SCPD, one that is usable by Device Builder in the next step of the development process.

Device Development Using Intel Device Builder

Device Builder automatically aggregates well-formed SCPDs and generates embedded code and a sample application. This frees developers from the tedious, time consuming task of creating UPnP stack-level logic, and allows them to focus instead on development of device-level logic. Device Builder also provides for the creation of UPnP devices containing embedded UPnP devices.

Traditional UPnP development methods have been problematic because they rely on the manual creation and editing of UPnP XML documents and the manual tracking of dependencies between XML documents.

Device Information

The Device Information block in Figure 1, located

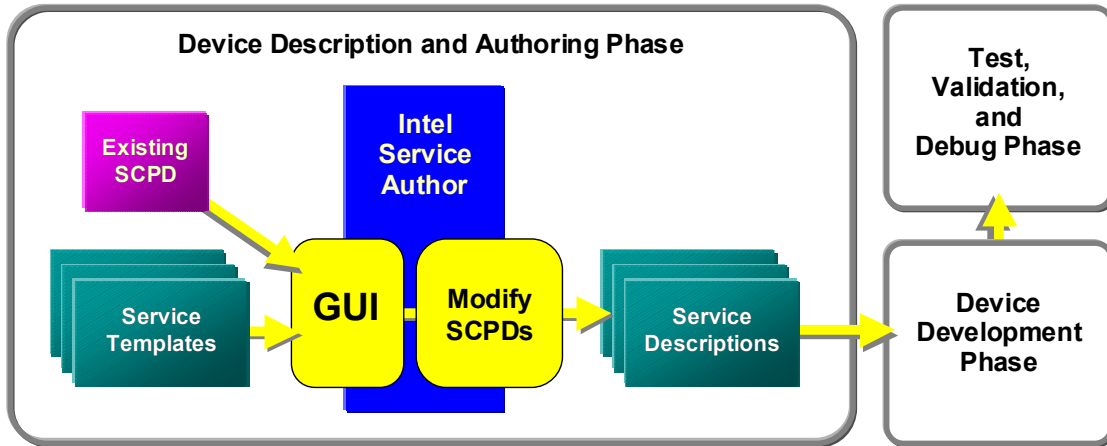


Figure 2. Intel Service Author

on page 1, represents the developer’s high level organization of a device. A high level organization describes a device’s services and embedded UPnP devices. The developer provides this information through a graphical user-interface (GUI) that displays the hierarchy of devices and services. The GUI also allows the developer to easily specify basic information about the device, such as the manufacturer and model number.

Intel Device Builder

After organizing the SCPDs into a hierarchy of UPnP devices and services, the developer chooses from a set of code-generation options including the target platform and the source code language. When the developer uses Device Builder to generate a device or control point stack, it generates two outputs: a custom UPnP stack and a sample application.

Code Generation for Multiple Platforms

The magic of Device Builder is its ability to generate code (C, C++, and C#) for many different platforms, both embedded and PC based. The resulting UPnP stacks, called Microstacks, are very

compact, making them suitable for embedded devices.

Microstack (C/C++) Features and Benefits

Supported platforms include POSIX (which includes Linux/Unix derivatives), Win32, and WinCE

- Extremely compact C and C++ compliant code
- Supports Winsock1 and Winsock2 on Windows
- Very few library dependencies
- Complete inbound type checking
- Automatic error generation
- Asynchronous and fragmented action response options
- Multiple network interface support and Auto-IP hooks
- Fully UPnP compliant—every time it’s generated
- Automatically generates a make-file or Microsoft Visual Studio project file, with a sample application that is ready to compile and run immediately

Intel Development Tools for Implementing UPnP Devices

March 2003

Device Builder also generates C# source code for the .NET platform. These stacks are larger, but offer additional features and benefits.

.NET Host Stack (C#) Features and Benefits

- No native code, runs on any .NET host
- Full, dynamic, multi-network support
- Bi-directional type checking
- Fully HTTP/1.1 compliant, chunked encoding and pipelining; auto fallback
- Fully multi-processor and hyper-threading enabled

No Hand-Coding Means No Errors

Intel Device Builder relieves the developer from having to hand-code the UPnP device description document, which eliminates UPnP or XML syntax issues. The process also guarantees that the dependencies between devices and services match the developer's UPnP specifications for the device, effectively eliminating interdependency errors between XML documents. Furthermore, all of these documents are embedded in the source code eliminating the need for a file system.

Sample Application

To ease the learning curve for developers, Device Builder generates a sample application built on the foundation of embedded code that makes up the UPnP stack. The sample application executes as a UPnP device without any device logic, but the generated code is complete enough to handle all of the UPnP level behavior. Comments in the sample application advise developers where they should add their own specialized code to build the actual device logic. They are free to modify the sample application, or they can simply use the sample as a learning tool for developing their own application code.

Native Compiler

The native compiler used will depend on the platform, but the generated output supports GNU GCC/G++ compilers for Posix-C/C++ source code and Visual Studio for Win32 and .NET source code. The generated source code is complete enough that the compiled executables will pass the syntactic tests for advertisements, subscriptions/eventing, and method invocations found in the UIC (UPnP Implementers Corporation) certification tests².

Developer's Focus on Device-Specific Logic

Using Device Builder allows the UPnP device developer to focus on creating the device specific logic. As demonstrated by the sample application, the developer no longer needs to perform many tasks required by other available UPnP stacks such as:

- Parsing XML messages at the UPnP level
- Formatting events at the UPnP level
- Writing event XML messages

Device Builder abstracts the standard UPnP behaviors so the developer is required to know little or nothing about UPnP technology and XML. As a result, the developer's energy can be focused on the development of logic for a specific UPnP device.

Intel UPnP AV Microstacks

Intel UPnP AV Microstacks are reference designs for UPnP AV devices and control points. They include a media server, media renderer, media server control point, and media renderer control point. As reference designs, they contain many of Intel's key learnings and represent the best known

² Certified devices are those devices that have implemented at least one complete device standard approved by the UPnP Forum and have successfully completed the UIC certification process.

Intel Development Tools for Implementing UPnP Devices

March 2003

methods for creating UPnP AV devices and control points. (See Figure 3, below.)

Each AV Microstack was initially generated by Intel Device Builder with a minimal set of UPnP AV features. Intel's key learnings and best known methods were added manually (for example, efficient play list processing and transport controls for an AV media renderer). The resulting AV Microstacks are lightweight (Micro Media Server is only 74.5 KB on PocketPC/ARM) and able to run

on cost-sensitive, resource limited platforms like (many) consumer electronics devices.

Depending on the feature set desired for a specific product, a developer can use an Intel UPnP AV Microstack in one of two ways: 1) as the required UPnP stack with little or no revision, or 2) as a cut-and-paste resource. In the latter case, the developer would first use Service Author and Device Builder to generate an embedded code stack as close to the product specification as possible and then customize the UPnP AV feature set as needed. A customized feature set for an AV media server might include advanced sorting and content management.

Micro Media Server

Micro Media Server (MMS) provides the minimum functionality a content directory server must have in order to pass UPnP compliance testing. In operation, MMS searches for a storage card attached to the device it is serving and attempts to share it. Suitable storage cards are PC Flash, MMC (Multimedia Memory Card), or SD (Secure Digital).

Micro Media Renderer

Micro Media Renderer (MMR) advertises support for MP3 and WMA media and M3U play lists. The Win32/Pocket-PC versions of the device use Microsoft's Media Player for actual playback, but the renderer's state machine logic is in POSIX-C, making it very easy to run on Linux/Unix platforms as well.

Micro Media Server Control Point

In addition to source code for devices, Device Builder generates control point source code for Linux, POSIX, Windows, and PocketPC. As with Micro Media Server, the Micro Media Server

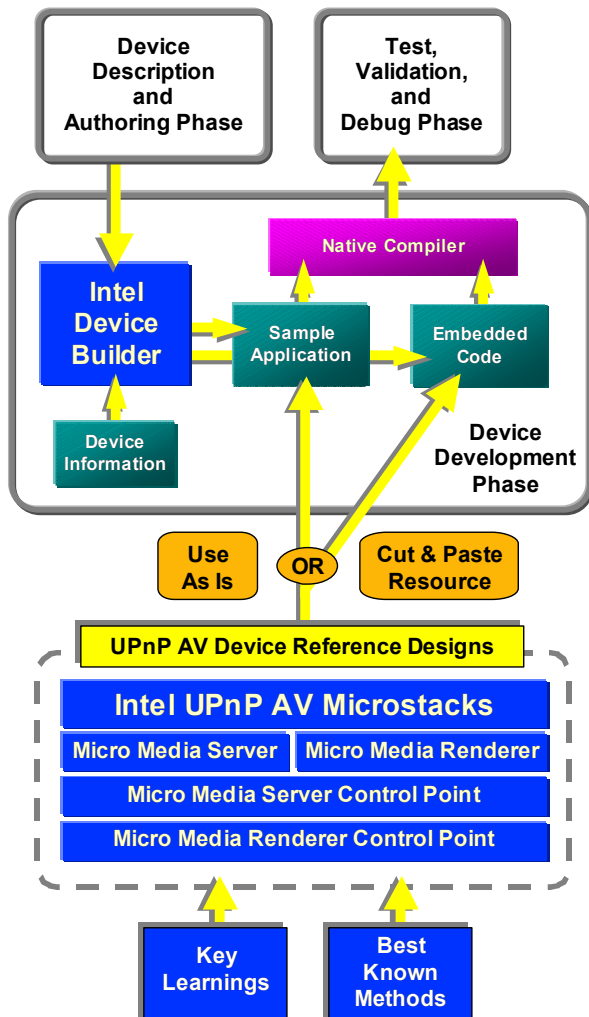


Figure 3. Intel UPnP AV Microstacks

Intel Development Tools for Implementing UPnP Devices

March 2003

Control Point (MMSCP) generated stack is very small (around 50 to 100 KB) and is custom built to control a specific device type. The generated sample control point application makes it easy to understand how a MMSCP works and how to integrate it with a device or application.

Micro Media Renderer Control Point

Intel Micro Media Renderer Control Point (MMRCP) is a generic renderer control point able to control many types of renderers, including those for audio, video, and imaging. Since renderers need to be very flexible devices, this module looks at both the device and service description files and offers an appropriate set of options depending on what the renderer device supports.

Debug ,Test, and Validation Using Intel Device Validator

Until the advent of Device Validator, debug, test, and validation of UPnP devices was mostly a manual process. The Device Validator tool provides a method to automate device testing.

Intel Device Validator

With the push of a few buttons, Device Validator automatically tests devices for UPnP and UPnP AV compliance. When testing is complete, the UI indicates test results by one of three light states (see Figure 4).

- Green Light: Pass—complies with pertinent UPnP standards
- Red Light: Fail—specific points of non-compliance are identified
- Yellow Light: Warning—areas of technically compliant behavior, but not recommended behavior

In addition to checking for compliance with the UPnP specification, Device Validator will warn the

developer when best known practices and implementation guidelines³ are not followed. Another useful feature of Device Validator is its plug-in architecture that allows the addition of new test modules. This makes Device Validator a valuable tool for adding new test functionality to a regression harness.

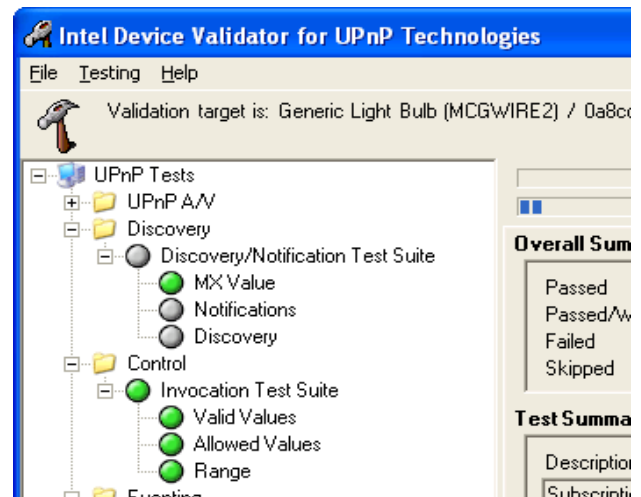


Figure 4. The GUI for Intel Device Validator.

Intel UPnP Tools

In addition to development and validation tools, Intel has provided developers with tools for manual testing, including a few tools that can work with any type of UPnP device. This suite of tools is available at:

<http://www.intel.com/labs/connectivity/upnp/>

Intel Device Spy

Intel Device Spy is a universal control point (UCP) that supports manual diagnostics for individual actions and events. Device Spy offers a developer these capabilities:

³ The official Universal Plug and Play Vendor's Implementation Guide can be found at <http://upnp.org/resources/documents.asp>.

Intel Development Tools for Implementing UPnP Devices

March 2003

- Detection of all UPnP devices on a network
- Display of detailed UPnP device information
- Action invocation
- Event monitoring
- Error reporting

Device Spy is an excellent learning tool for developers new to UPnP technology. It provides an opportunity to interact with UPnP devices and begin to understand how such devices work.

For more experienced developers, Device Spy provides a way to control any type of UPnP device. It allows them to manually test features during implementation-debug cycles, or to examine how existing UPnP devices work. Through its invocation stress testing, error reporting, and outbound/inbound packet tracing, Device Spy enables developers to discover and fix potential problems that may occur during UPnP software development.

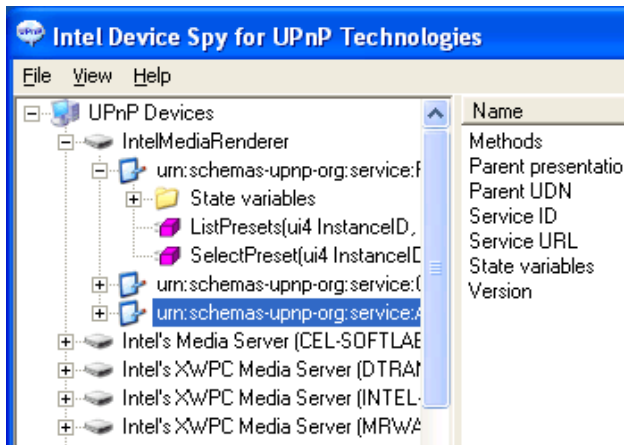


Figure 5. The GUI for Intel Device Spy.

Intel Device Sniffer

Intel Device Sniffer monitors all UPnP broadcast traffic on the network. It behaves like a UPnP ‘packet sniffer’ and provides a log of UPnP device

advertisements. This informs the developers of UPnP devices, their location on the network, and provides a means to examine advertisements for incorrect formatting. Device Sniffer capabilities include:

- Performing HTTP requests
- Scanning SSDP (Simple Service Discovery Protocol) broadcasts
- Issuing M-search requests
- Retrieving documents responsible for representing devices on the network
- Filtering IP Address packets
- Issuing SSDP search requests
- Solving SSDP interoperation problems

Intel Device Relay

Intel Device Relay is used to mirror UPnP devices onto another UPnP network—even across the Internet between two distant points. When started on two different computers that have an IP route between them, Device Relay mirrors UPnP devices onto the remote UPnP network. On the remote network, control point nodes are able to discover a mirrored UPnP device and this device effectively behaves as a proxy for action invocations and events. The mirrored device even advertises on behalf of the actual device.

Intel Multimedia Tools

Intel Multimedia Tools are UPnP AV 1.0 compatible and are provided to the industry to help developers build products based on this new specification. These tools have been thoroughly tested with many vendors’ implementations at numerous UPnP AV interoperation events.

Intel Multimedia Tools are valuable to implementers as well as to novices interested in understanding the behavior and process flow of UPnP AV devices and control points. Each one of

Intel Development Tools for Implementing UPnP Devices

March 2003

the tools represents a key component of the AV media distribution and control model, either as a media server, a media renderer, or a control point. With these tools, anyone can setup complete AV scenarios or test new AV products.

When a developer is building a UPnP device that is only a part of a larger UPnP scenario, the variety of AV tools will be particularly helpful. For example, a vendor building a media server device can do some basic interoperability testing with the Media Renderer and AV Controller, while a vendor building a media renderer can test with Media Server and AV Wizard.

AV Media Controller

Intel AV Media Controller functions as the developer's switchboard for home media distribution with an emphasis on AV diagnostics. It implements a control point for finding content on media servers and also implements a control point for controlling media renderers. A tree view of all the AV content on the network is provided, along with all the necessary on-screen controls like volume, play, and pause. The developer is able to select content and direct it to a specific renderer. This tool provides a developer with the means to perform playback scenarios involving a Media Server and a Media Renderer.

AV Media Server

Intel AV Media Server is a complete, easy-to-use UPnP AV content directory that advertises content distributed through the HTTP-GET protocol. It uses the file system as a database allowing the user to drag-and-drop folders for quick sharing of media. AV Media Server's GUI allows real-time tracking of endpoints that are downloading content from the server.

AV Media Renderer

Intel AV Media Renderer is a full-featured UPnP AV media renderer that supports play lists, complete moderated eventing, track-seeking, position-seeking, and much more. This tool implements all of the actions that the UPnP AV specification defines for AV media renderers. AV Media Renderer is configurable to expose different features, allowing developers to experiment with permutations of renderers.

AV Wizard

Intel AV Wizard is a UPnP AV control point for playing locally stored content to available UPnP media renderers. It includes many ease-of-use features such as auto-play, drag-and-drop, and transparent play list creation.



Figure 6. The GUI for Intel AV Wizard.

Conclusion

Intel development tools for UPnP devices are designed to reduce the development time for UPnP devices and assure fully compliant UPnP and UPnP AV functionality. They are designed to be easily used by novices and experts alike, covering all phases of the development and implementation process.

See the latest Intel software for UPnP Technology at:

<http://www.intel.com/labs/connectivity/upnp/>

Appendix

Concepts and Terminology

Control Point (CP): An intelligent network node or application that has the ability to discover and control UPnP devices on the network. UPnP technology makes a clear distinction between a logical device and a Control Point on the network.

Control Point Stacks: Code modules that abstract the specifics of the UPnP protocols for a control point, offering the user a relatively easy API that can be used to discover devices, invoke actions, subscribe to and receive events.

Device: A logical container of services that may contain other devices. All UPnP devices advertise basic information about themselves on the network such as manufacturer and serial number. Some devices also offer a device web page.

Device Stack: A code module that abstracts the specifics of the UPnP protocols for a device. The provided API allows the developer to quickly advertise services, respond to control point invocations, and fire events. Some Device Stacks allow the user to advertise a web page along with the device.

Intel Microstack: Intel embedded UPnP stack, the output of Intel Device Builder.

Services: Logical entities providing a specific service to the UPnP device network. They are composed of actions that can be invoked on the

service and events that a control point can subscribe to.

Service Control Protocol Declaration (SCPD):

An XML file that describes a UPnP service. It contains information such as actions, state variables, arguments for each action, and available events. Control Points can download SCPD documents from discovered devices to verify which services, actions, and events are available.

UPnP Device Reference Design: A sample design for a specific type of UPnP device (e.g.: AV media renderer) that can be used by product developers as a basis for their design. Typically a reference design has all the features and capabilities of the UPnP device it references.

UPnP Stack: A generic term used to describe a Device Stack or Control Point Stack, or a combination of both.

Universal Control Point (UCP): A generic or universal type of Control Point that can control all UPnP devices on the network. Universal Control Points are generally used when developing and testing UPnP devices. Intel Device Spy is a good example of a UCP.

Operational Phases of UPnP Devices

The operation of UPnP devices is divided into the following phases:

Discovery: In this phase, UPnP control points search for UPnP devices and services, and UPnP devices advertise their services.

Description: Once a UPnP control point finds a UPnP device or service of interest, it requests its description document. UPnP devices and services respond by sending XML description documents that define the actions and attributes they support.

Intel Development Tools for Implementing UPnP Devices

March 2003

Control: In this phase, UPnP control points invoke the actions described in the XML description documents associated with the services they control. These actions are executed by the services and typically cause changes in the service states and attributes.

Eventing: UPnP control points subscribe to event servers hosted by the services, which allows them to receive events from a specific service they are interested in. Similar to control actions, events are defined in the corresponding STs (Service Templates).

Presentation: Finally, UPnP devices may choose to host an HTML document that provides a simple GUI for the device.