USB Redirection

System and Software Requirements Document

	Prepared By	Reviewed By	Approved By
Name	Larry Hamelin		
Role	Project Manager		
Signature			
Date	Dec 2008		

Table of Contents

1 Introduction

1.1 About this document

This document specifies the functional and non-functional requirements for the USB Redirection system to be developed, and serves as input for design documentation and development.

The scope of this document is limited to addressing the user's requirements and specifying the quality requirements and overall system architecture. It is recommended that design documentation not be added to this document,

2 Proposed system

2.1 Overview

2.1.1 Nomenclature

In this document, the **client** designates the computer to which the USB device is physically attached; the **server** designates the computer using the USB device.

2.1.2 Description

The USB Redirection system makes a wide variety of USB devices available across a network, using either a direct network connections or Windows Terminal Services Remote Desktop Protocol. To use a remote USB device, users of the system will not have to perform any additional installation or configuration other than connecting to the remote USB device using the system.

The system will:

1. Support OHCI, UHCI and EHCI standards
2. Encrypt network traffic for secure devices sharing
3. Share USB device with others over network using a single application
4. Work with both physical and virtual Windows client and server machines
5. Work with both Windows, Mac and Linux clients
6. Auto-connect to new USB devices

The system will support the following kinds of USB devices:

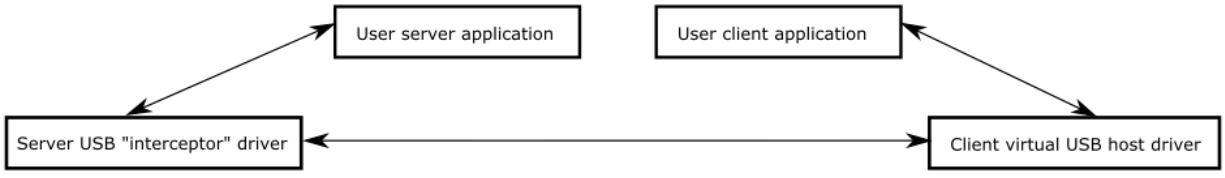
7. Communications: Modems, telephones
8. Human Interface Devices (HID): Keyboards, mice, audio devices
9. Peripherals: Printers, card readers
10. Storage Devices: Hard drives, flash drives and CD-ROM/DVD drives
11. Image Devices: Scanners, cameras, webcams
12. Serial Devices: Serial and Parallel USB converters
13. Other Devices: All other devices compliant with OHCI, UHCI and EHCI standards.

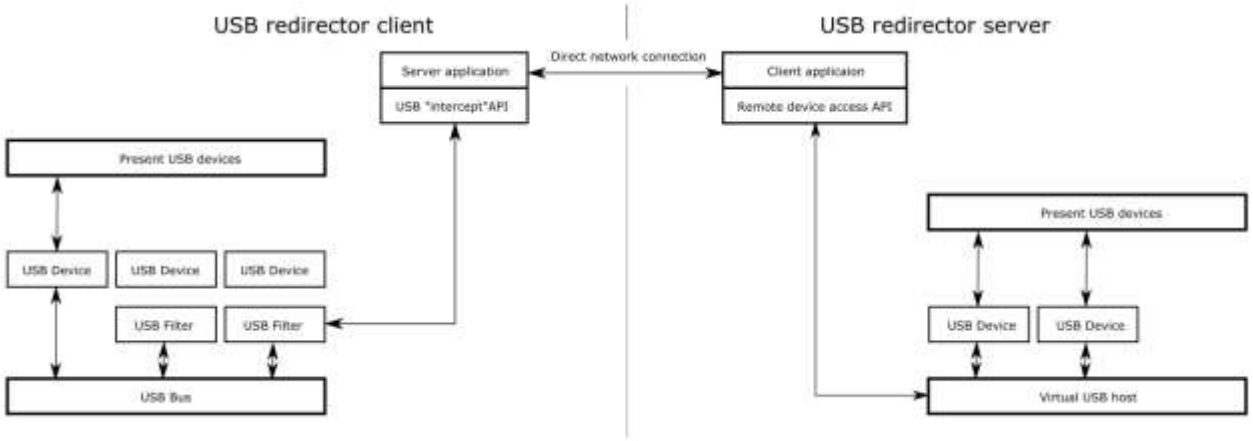
The system will include a graphical user interface allowing users to make USB devices available to other users and to attach to USB devices available on other computers. This user interface will also be used to communicate dynamically with the lower-level USB redirector.

2.2 Architecture

2.2.1 Direct Network Connections

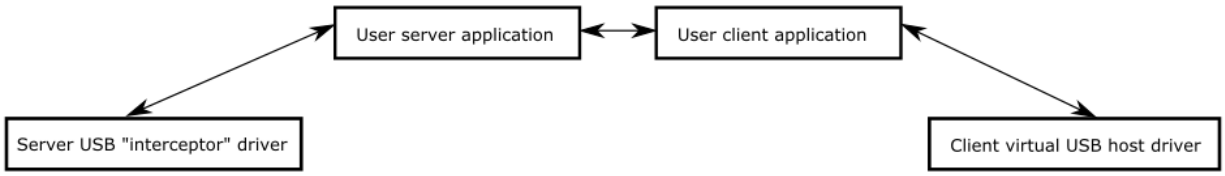
When a direct network connection is available between the client and server machines, the communications will occur directly between the USB drivers.

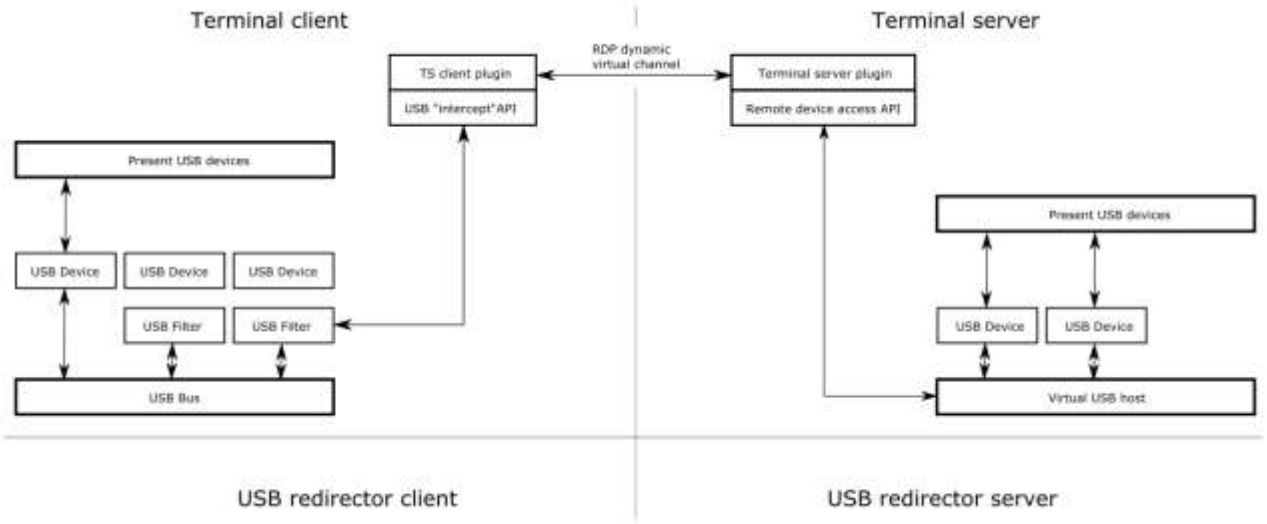




2.2.2 Terminal Services

When connecting via Terminal Services, a direct driver-to-driver network connection will be unavailable. Communications will have to be established between the application layers using an RDP virtual channel.





3 System Requirements

3.1 Development Environment

3.1.1 General

Development will examine the XXX and the YYY products for reverse-engineering and functionality ideas.

Development will be based on the existing USB/IP project. Care should be taken to comply with open-source licensing requirements.

3.1.2 Windows

Development will be conducted using Microsoft® Visual Studio® 2008 in C++ using the Driver Development SDK under Microsoft® Windows® XP Pro SP2.

GUI components will use Microsoft® Visual Studio® 2008 in C++ using MFC. C++/MFC provides the best capability for communicating with lower driver-level components. [Alternatively C# and Windows Forms can be used

Settings and other persistent configuration information will be stored in the registry.

Testing will be conducted using the operating systems specified in the Operating System Requirements.

3.1.3 Linux

3.1.4 Macintosh

3.2 Operating System Requirements

- Clients and Servers
  - Microsoft® Windows® 32- & 64-bit XP Home Edition SP2 or higher
  - Microsoft® Windows® 32- & 64-bit XP Professional SP2 or higher
  - Microsoft® Windows® 32- & 64-bit Vista Home Basic
  - Microsoft® Windows® 32- & 64-bit Vista Home Premium
  - Microsoft® Windows® 32- & 64-bit Vista Business
  - Microsoft® Windows® 32- & 64-bit Vista Enterprise
  - Microsoft® Windows® 32- & 64-bit Vista Ultimate
  - Microsoft® Windows® 32- & 64-bit Vista Home Basic with SP1
  - Microsoft® Windows® 32- & 64-bit Vista Home Premium with SP1
  - Microsoft® Windows® 32- & 64-bit Vista Business with SP1
  - Microsoft® Windows® 32- & 64-bit Vista Enterprise with SP1
  - Microsoft® Windows® 32- & 64-bit Vista Ultimate with SP1

  - Microsoft® Windows® 32- & 64-bit 2003 Server or higher (with Terminal Services for server)
  - Microsoft® Windows® 32- & 64-bit 2008 Server or higher (with Terminal Services for server)

- Clients only
  - Linux version xxx or higher
  - Macintosh OS-X

3.3 Hardware Requirements

3.3.1 Computer Hardware

The system should run on any computer capable of running the specified operating system. The general requirements are:

Required Processor	Recommended Processor	Required RAM	Recommended RAM	Hard Disk Space
Pentium 400 MHz*	Pentium 1 GHz or faster	96 MB*	256 MB or higher	200 MB

**Or the minimum required by the operating system, whichever is higher.*

3.3.2 USB Hardware

The system will be able to operate with the following USB hardware:

1. Communications: Modems, telephones
2. Human Interface Devices (HID): Keyboards, mice, audio devices
3. Peripherals: Printers, card readers
4. Storage Devices: Hard drives, flash drives and CD-ROM/DVD drives
5. Image Devices: Scanners, cameras, webcams
6. Serial Devices: Serial and Parallel USB converters
7. Other Devices: All other devices compliant with OHCI, UHCI and EHCI standards.

3.4 Additional System Requirements

The client must install and configure Microsoft Terminal Services to employ the Terminal Services RDP.

Name	Version	Installation
Microsoft Terminal Services	xxx	Customer

4 System Modules

1. USB driver redirector/interceptor (client)
2. USB virtual host controller interface (service)
3. Network data protocol handler
4. GUI and/or command-line applications for configuration/status

4.1 USB driver redirector/interceptor

The USB driver redirector/interceptor is a device driver on the client computer that mediates between the USB bus (for discovery) and devices and the network protocol handler to service device usage. It must also interface with the GUI for settings

4.1.1 USB device communication

The redirector must communicate with each physical USB device
Standards:

1. OHCI (USB 1.0)
2. UHCI (USB 1.1)
3. EHCI (USB 2.0)

Operating system specific:

4. Windows: Single USB bus upper filter driver for all USB devices
5. Linux: Stub driver for device redirection per USB/IP
6. Macintosh: USB Interface Module (UIM) kernel extension (KEXT)

4.1.2 USB bus

1. Enumerate existing USB devices
2. Detect new USB devices when inserted
3. Detect when devices have been stopped and/or removed

4.1.3 Interface with network protocol handler

Device operations:

Status:

1. Notify protocol handler when devices mounted, stopped, or removed
2. Notify device driver when server remotely mounts/dismounts device?

4.1.4 Interface with GUI/settings

[How? What functionality? The following are guesses on general principles]

Settings:

1. Which devices to publish. *[This functionality might best be implemented in the network protocol handler component.]*
2. Security: Get users are allowed to use each published device. *[I recommend using ACLs on registry entries. This functionality might best be implemented in the network protocol handler component.]*

Status:

3. Notify GUI which devices are mounted *[in use?]* on a remote server.
4. Disconnect server(s) when directed by GUI

5. Notify GUI when device physically inserted, stopped (from OS) or removed.

Control:

Respond to device stop attempt from GUI. If the device is in use, the redirector reports failure (if this functionality is not provided by the operating system). If device mounted on remote server, notify GUI and give user a chance to confirm. If confirmed, stop device and allow safe physical removal.

4.2 USB virtual host controller

The virtual host controller is a device driver [service? application?] that resides on the server machine and presents the remote USB devices as if they were available locally.

4.2.1 Interface with Network Protocol Handler

Device operations: *[How?? Who drives USB communications?]*

Status:

1. Notify client when mounting/dismounting remote device
2. Respond when client inserts, stops or removes a physical device

Control: Respond to forced dismount from client

4.2.2 Interface with GUI/Settings

[How? What functionality? The following are guesses on general principles]

1. Settings: Determine which remote devices to mount automatically.
2. Status: Dynamically notify GUI when remote devices are inserted, stopped or removed.
3. Control: Mount/dismount device as directed by GUI.

4.3 Network Protocol Handler

The network protocol handler resides as a service *[device driver? application?]* on both the client and server machines to handle the communications between the two machines. All USB communications will be redirected to this protocol handler to enable device operation.

The protocol handler must *[optionally?]* encrypt all communications between the client and server using *[what method?]*

4.3.1 Device communications

Communications with the device redirector and virtual host controller are specified under the appropriate modules.

4.3.2 Abstract network layer

The abstract network layer is an abstract layer that can be specialized to individual network low-level protocols.

The network protocol handler will consist of a component to talk to the device redirector/virtual host controller and a virtual network layer. The network layer can be specialized to communicate directly over TCP/IP or use Windows Terminal Services RDP as necessary.

4.4 GUI/CLI applications

[The following are guesses based on general principles. Specific functionality/screen shots are not yet included; they might best be deferred until the design phase.]

The GUI (Windows/Mac) or CLI (Linux) applications display and set configuration information, show status, and issue immediate commands to the redirector and virtual host controller.

In those cases where a computer is both a client and a server (Windows only) the GUI for both the client and server functionality should appear in the same application.

GUI applications on Windows should use C++/MFC. This platform should allow the most efficient communications with the underlying drivers.

4.4.1 Redirector (client)

The GUI/CLI for the redirector should:

1. Display the USB devices available for remote users to access
2. Display which devices have or have not yet been made available
3. Allow the user to specify the availability of each device
4. Allow the user to specify which users or groups may remotely mount each available device
5. Show whether each device is presently remotely mounted
6. Allow the user to forcibly dismount the device from the remote server

On Windows and Mac systems, this application should be a GUI. On Linux systems, the application should be use a command line interface.

4.4.2 Virtual host controller (server)

The GUI/CLI for the virtual host controller should:

1. Allow the user to browse the machines on the network, and see each USB device available for mounting (must be published and the user must have permission)
2. Allow the user to transiently mount remote devices
3. Allow the user to automatically mount remote devices (device is automatically mounted when available

4. Allow the user to transiently dismount a mounted device
5. Allow the user to permanently dismount a device already selected for automatic mounting
6. Notify the user when the device has been mounted by another server

Since the virtual host controller is specified only for Windows machines, this application should be a GUI.

5 Other issues

5.1 Concurrency

How is concurrency handled? Can two servers mount the same client device? If two remote servers attempt to use the same USB device at the same time, how is that handled? Are all USB transactions synchronous and stateless? If synchronous but not stateless, where is the best place to keep state information?

5.2 Encryption

How should encryption be handled? Should SSL-like certificate management will work on all platforms, but requires additional administration. An embedded shared symmetric key will prevent casual eavesdropping, but is vulnerable to decompiling or other hacking. Is encryption really necessary? Communications should be either within a corporate intranet (physically secure) or through wireless or a VPN (already encrypted).

Revision History

Version	Date	Description	By
	29 Dec 2008		