Net2Display™ Standard

860 Hillview Court, Suite 150
Milpitas, CA 95035

Phone: 408 957-9270
Fax: 408 957-9277
URL: www.vesa.org

VESA Net2Display Remoting Standard
Version 1
October 22, 2009

Purpose

The purpose of the Net2Display™ Remoting standard is to provide efficient remoting of computer display and I/O capabilities across local and wide area computer networks. Supported I/O capabilities include keyboards, pointers, audio and USB attached peripheral devices. Net2Display Remoting is designed to be capable of responsiveness, performance and full motion video user experience comparable to a local PC.

Summary

Net2Display Remoting is an extensible display and I/O remoting architecture for remoting displays, keyboards, pointers and USB I/O devices across local interconnects, local networks and wide area networks using IP wired or wireless networks.Net2Display Remoting is architected to allow simple client devices and host systems that can connect with multiple clients to provide multiple display capabilities.

Net2Display provides Virtual Channels for handling different types of data traffic with different priorities so that interactive traffic is prioritized. Discovery mechanisms for Net2Display Hosts and Client use basic IP network facilities so that discovery occurs without user intervention.

Table of Contents

Tables

Figures

Preface

Intellectual Property

While every precaution has been taken in the preparation of this standard, the Video Electronics Standards Association and its contributors assume no responsibility for errors or omissions, and make no warranties, expressed or implied, of functionality or suitability for any purpose.

Trademarks

All trademarks used within this document are the property of their respective owners.

DisplayPort, DPVL, E-DDC, E-EDID, Net2Display and VESA are trademarks of the Video Electronics Standards Association.

Microsoft, Direct3D, DirectX, Windows, Windows Media, Win32, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

X Window System is a trademark of The Open Group.

OpenGL is a trademark of Silicon Graphics, Inc.

DVI (Digital Visual Interface) is a trademark of the Digital Display Working Group.

HDCP (High-bandwidth Digital Content Protection) is a trademark of the Digital Content Protection LLC (www.digitalcp.com).

Patents

VESA draws attention to the fact that it is claimed that compliance with this specification may involve the use of a patent or other intellectual property right (collectively, "*IPR*") concerning the Net2Display Remoting protocol. VESA takes no position concerning the evidence, validity, and scope of this *IPR*.

The following holders of this *IPR* have assured VESA that they are willing to license the *IPR* on *RAND* terms. The statement of the holder of this *IPR* is registered with VESA.

Holder Name	Contact Information	Claims Known
Columbia University	Jason Nieh (nieh@cs.columbia.edu)	Pending US Patent: 20060184614
PCXTEC Technology Inc.	Chunyao Li (cli@pcxtec.com)	Chinese Patent ZL 200510059381.1 Pending Chinese Patent Application ZL 200710143473.1
Qualcomm Incorporated	Thomas R. Rouse (trouse@qualcomm.com)	US Patent 6,760,772 Pending US Patents: 12/098,025, US20080037506, US20080045149, US0060288133, US20040199652

Attention is drawn to the possibility that some of the elements of this VESA *Specification* may be the subject of *IPR* other than those identified above. VESA must not be held responsible for identifying any or all such *IPR*, and has made no inquiry into the possible existence of any such *IPR*.

THIS *SPECIFICATION* IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS *SPECIFICATION* MUST BE MADE ENTIRELY AT THE *IMPLEMENTER*'S OWN RISK, AND NEITHER VESA, NOR ANY OF ITS *MEMBERS* OR *SUBMITTERS*, MUST HAVE ANY LIABILITY WHATSOEVER TO ANY *IMPLEMENTER* OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS *SPECIFICATION*.

Support

Clarifications and application notes to support this standard may be written. To obtain the latest standard and any support documentation, contact VESA.

If you have a product that incorporates Net2Display, ask the company that manufactured your product for assistance. If you are a manufacturer, VESA can assist you with any clarification you may require. Submit all comments or reported errors in writing to VESA using one of the following methods.

Fax 408-957-9277 to attention of Technical Support at VESA

E-mail support@vesa.org

Mail Technical Support

Video Electronics Standards Association

860 Hillview Court, Suite 150

Milpitas, CA 95035

Acknowledgments

This document would not have been possible without the efforts of the VESA Display Systems Standards Committee's Net2Display Task Group (originally the Packetized Video Task Group). In particular, the following individuals and their companies contributed significant time and knowledge.

Table 1: Main Contributors to Version 1, Revision 1

David Glen	ATI	
John Hickey	Avocent	
Alan Ricker	Barco	
Uri Elzur	Broadcom	
Amit Oren	Broadcom	
Neal Margulis	Calista Technologies	
Ricardo Baratto	Columbia University	
Jason Nieh	Columbia University	
Joe Goodart	Dell	
Clinton Battersby	Desktone	
Joe Lamm	Display Labs	
Jim Webb	Display Labs	
Takashi Matsui	EIZO NANO	
Bryan Speece	Entech Taiwan	
Norm Brown	Hewlett-Packard	
Bob Myers	Hewlett-Packard	
Kenneth Ocheltree	IBM	Task Group Chair & Editor
Steve Millman	IBM	
Kai Schleupen	IBM	
Martin McDonnell	Intel	
Sridharan Ranganathan	Intel	
Eric Wogsberg	Jupiter Systems	
Tom Ligda	nComputing	
Gabriele Sartori	nComputing	
Bill Bredbenner	Neoware	
William Tsu	NVIDIA	
Chunyao Li	PCXTEC Technology	
Harry Peterson	PixelWorks	
Michael Anderson	Portrait Displays	
Phil Karn	Qualcomm	

Ranga Krishnan	Qualcomm	
Gorge Wiley	Qualcomm	
Peter Ghostine	Quest Software	
Jin Hui	Quest Software	
Stanley Kim	Samsung	
Ian Miller	Samsung	
Sanjeev Datla	Server Engines	
Robert Blanchard	Sony Corporation	
Graham Loveridge	STMicroelectronics Inc.	
Dave Hobbs	Teradici	Task Group Vice-chair
Stuart Robinson	Teradici	
Ken Unger	Teradici	
Bing Ouyang	TI	
Alain d'Hautecourt	ViewSonic Corp.	
Dustin Byford	VMWare	
Daniel Petersen	VMWare	
Daniel Barreto	Wyse Technology	
Curt Schwebke	Wyse Technology	

Revision History

October 22, 2009	Intitial release of the standard

1 Objectives and Nomenclature

1.1 Summary

This standard describes a set of commands used by a host computer system to send and manipulate display data and I/O on a display. A display utilizing this standard may be connected to a host computer via Ethernet, 802.11 wireless, USB, IEEE 1394 or any other suitable networking or I/O interface. This standard makes two simplifying assumptions:

- The interconnection used between the host computer and the display supports the Internet Protocol (IP) networking protocol
- I/O Devices connected to the client display are remotely connected to the host primarily using the Universal Serial Bus (USB) protocol

1.2 Compliance with Standard

This document contains example implementations of host, client and network systems. These are useful for explaining terms and possible configurations, which in turn help explain the features and capabilities of the standard and explain why specific protocols have been selected. However, these explanations do not define required compliance with the standard.

Sections 1 to 5 of this document are primarily system examples as described above and should not be consider necessary requirements of a compliant implementation unless specifically noted.

In general, compliance with the standard requires predictable and interoperability. This only requires compliant network communications, where the network communication includes media data, control and the configuration necessary for the media data to be correctly generated, communicated and interpreted.

A compliant implementation of Net2Display is limited to compliance of items enabled by this specification. Statements within this specification that indicate capabilities that have not been full enabled by this specification are not required by a compliant implementation of this specification.

1.3 Background

While there are display standards defining the transmission of display content over local display connections using digital transmission and packets, no standard exists that specifies the remoting of displays and I/O devices over a network. A number of approaches for remoting displays over networks currently exist, including Microsoft Remote Desktop Protocol (RDP), Citrix MetaFrame ICA protocol, VNC, Sun Ray, and X**.** While each of these approaches is capable of remoting displays over networks, each has distinct disadvantages and collectively, the multiplicity of protocols presents problems as well.

All of the existing approaches have limitations in supporting full motion video, supporting WAN latencies and in providing desktop PC-like responsiveness for the richly enhanced GUI provided by modern operating systems. The existing remoting approaches were all designed before motion video was prevalent on the PC and motion video quality is currently inadequate due to server pull models, lazy display update or synchronization issues. Server pull models, incompatibility of different versions and proprietary protocols also limit the ability of existing protocols to provide an open remoting environment. Consequently, there is an opportunity for a remoting standard that is designed at the onset to support motion video and provides the interoperability benefits of a standard, allowing the innovations in performance and low cost implementations that an open standard promotes.

1.4 Objectives

The objective of the VESA Net2Display standard to create a standard which allows host server computers to send display data and I/O to a client over a high speed data channel. The standard is agnostic to the physical

and datalink channels used. The Net2Display standard protocol is designed to operate at the most primitive level and should provide enablement for remoting I/O devices, for example, keyboard, mouse, audio, and storage. For display remoting, it will provide facilities which closely mirror the capabilities provided by the display subsystem. Net2Display was designed with the following features in mind:

- Minimizing the complexity of the client
- Supporting remoting distances from a meter to thousands of kilometers
- Keeping response times comparable with directly attached devices, typically below 100 milliseconds
- Using moderate bandwidth, typically less than 10 Mbps.
- Supporting high quality, full motion video
- Remoting USB devices without device driver modification
- Leveraging existing security protocols and infrastructures

1.5 Net2Display Features

In developing Net2Display Remoting to meet the above objectives, a number of desirable features were achieved. The features include the following attributes:

- **Simple Client** – A simple Client means that there is minimal state in the Client and the Client will require infrequent upgrades, if any at all.
- **3D Graphics and Flash on Server** – 3D Graphics and Flash are run on the higher performance Server processor to given the highest performance with minimal Client compatibility issues.
- **Flexible Server Location**
    - **Plug and Play Direct Connection –** When a Net2Display Client is attached directly to a Net2Display Host they recognize the direct attachment and the Client acts as display for the Host
    - **Automatic Local Service Location** – Local Net2Display Servers are found in the local environment by Net2Display using multicast requests.
- **Integrated Quality of Service –** Quality of Service delivery is built into the architecture of Net2Display with different types of data using different Virtual Channels delivered in different connections and with different priorities
- **TCP and UDP Transport** – The Transmission Control Protocol (TCP) is used for reliable delivery and the User Datagram Protocol (UDP) is used for real-time traffic such as motion video and audio.
- **Audio and Motion Video Synchronization** – Audio and motion video are delivered with timestamps to allow synchronization
- **Tolerant of High Latency**
    - **Local Cursor** – A local cursor may be provided on the Client to provide greater responsiveness
    - **Minimal Round Trips** – Net2Display Remoting is accomplished at a lower layer in the networking stack using a minimal number of round trips for each operation to minimize the effects of latency.
    - **Virtual Channels** – Virtual Channels can be used to offload additional function to the Client to reduce the effects of latency

- **Automatic Reconnection** – reconnection on restored or alternative network paths occurs automatically
- **Security** – Encryption is required, except for point-to-point direct connections
- **Efficient Resource Utilization**
    - **Variable Fidelity** – Display, motion video and audio quality can be adjusted to match the network and computing resources that are available
    - **Efficient Compression –** Motion video, audio and image compression algorithms are negotiated to use the best common algorithm and to minimize the network bandwidth

1.6 Scenarios

Net2Display supports a broad range of different client configuration for display remoting situations that exist today. While Net2Display Remoting is designed to allow displays to be connected to remote computers with ease, it is not limited to connections over long distances. It is able to be used in virtually all configurations where conventional displays or thin clients are currently used. Net2Display provides all of the simplicity of a conventional display, along with the flexibility of remote access when needed. Some major user scenarios that are considered for this standard are:

Administrative and Service Worker: Many administrative, service and clerical workers could easily have their PC systems replaced with a single Display remoting displays. The single display would be sufficient for typical tasks such as data entry, word processing and electronic forms. The frequency of screen updates is relatively slow and only a small amount of motion video is typically viewed. Support for an integrated IP Phone is desired in this scenario.

Professional Office Worker: By providing higher performance graphics and motion video, Net2Display Remoting can meet the requirements of a proportionally larger set of users than existing display remoting solutions. The professional office worker expects PC-like responsiveness and multiple display capability. Typical usage includes productivity applications and moderate usage of motion video. Support for an integrated IP Phone is also desired in this scenario.

Financial Trading Station: A financial trading station configuration has from two to 10 displays connected to one or multiple Net2Display Hosts Systems. Some network infrastructures prevent the installation of network switches for connecting multiple Clients in the financial trading station office, so other means for supporting the larger number of displays need to be used. Net2Display Remoting needs to meet certain latency requirements to be an acceptable replacement for existing PC trading stations.

Graphics Station: A graphics station configuration may be used for motion video editing or CAD wire frame rendering, providing higher performance graphics and capable of handling high video bandwidth with low latency. The graphics station needs to be highly responsive for the required rapid graphical manipulation.

Medical Display: A medical display used at by a doctor or at a nurse's station is a display with an integrated keyboard and no local patient information or external I/O ports to ease HIPPI compliance. Medical applications require the use of lossless compression across the network so that any images remain undistorted. Other medical displays may require USB ports for the local attachment of medical monitoring equipment.

Direct Display Attach: When a Net2Display Client is connected directly into the Ethernet port of a Net2Display enabled Host as the only display, the Host automatically recognizes the attached Client and configures it to receive the Host display output.

Display or Advertising Wall: This configuration is used for internal corporate communications or public advertising display walls. Such a wall is made up of one or an array of displays that would typically be all connected to the same Net2Display Host. These displays typically have no Input Facilities connected.

Conference Room Projector: A conference room projector is typically a stationary projector located in a conference room that is network connected as a Net2Display Client. It may be located from the Host using the Discovery Process or specified by name.

Internet Hosted Home or Small Business User: A home or small business user has Net2Display Clients on their local premises and typically accesses a remote computing service providing Net2Display Host capabilities. The Client is typically connected to the network via a NAT Router, which provides a locally assigned address to the Client.

Distributed Home Displays: A consumer may desire to have a number of Net2Display devices throughout the home that either access the same Host or multiple Hosts located either locally at home or remotely on the Internet.

Wireless Tablet: Net2Display Remoting allows a user to access a corporate or home wireless network using a wireless Client in a secure manner, allowing the user to discover services and access Hosts while preventing unauthorized access.

Mobile Device: A mobile user uses Net2Display to access their primary computer remotely via a smartphone or handheld device.

Additional Scenarios: Other additional usage scenarios include:

- Store Catalog/Map/Sizing Kiosks
- Store Purchase Kiosk with Credit Card Readers
- ATM Machines
- Cash Registers
- Patient Room Medical Instruments Monitors
- Library Internet Browsing PCs
- Remote Help Desk PCs
- Remote Local Insurance Office PCs
- Network TV, Projection TV or Projected Home Computer Displays
- High End Medical or Satellite Imaging Displays
- Gaming Home PCs

A summary of the characteristics of some of the major user scenarios are shown in Table 1-1.

Table 1-1: User Scenario Requirements Summary

Scenario	Number of Displays	Special Requirements
Administrative and Service Worker	1	Low cost, small footprint
Professional Office Worker	2	PC-like responsiveness
Financial Trading Station	4-10	Low Latency, Large number of displays, No Switch in Office, Multiple Hosts
Graphics Station	2	Low Latency
Medical Station	1-2	Lossless Compression
Direct Display Attach	1-2	Self Configuring
Display or Advertising Wall	1-16	No Input Facilities connected at display
Conference Room Projector	1-2	Simple Discovery
Internet Hosted Home or Small Business User	1-2	Needs to work across NAT
Distributed Home Displays	1-16	Needs to work across NAT
Wireless Table	1	Wireless, Security
Mobile Device	1	Security

1.7 Document Overview

This document is organized as follows:

1. **Objective and Nomenclature** gives the purposes and attributes of Net2Display remoting and the terms and conventions used in this document.
2. **Overview** gives a high level view of Net2Display remoting and the major components that make up a Net2Display remoting installation.
3. **Nodes** describes the different types of Net2Display nodes and their structures and gives examples of different Client configurations.
4. **Networking** describes the different network topologies, facilities and characteristics applicable to Net2Display remoting.
5. **Security** describes the security requirements and attributes of Net2Display remoting.
6. **Net2Display Element Requirements** defines the requirements for the different instances of Net2Display Nodes.
7. **Protocol Data Units** defines the structures, fields and processing used for passing Net2Display remoting traffic across a network.
8. **Associations** describe how a Net2Display Host and Client form a relationship for communicating Net2Display remoting traffic.
9. **Virtual Channels** defines the protocol for carrying different types and priorities of traffic independently across the network.
10. **Aggregated Coordinate Space** defines the use of coordinates in Net2Display for relating multiple displays, offscreen areas and pointing devices.
11. **Display Remoting Process** characterizes how display content is remoted by Net2Display remoting for different implementation approaches and operating systems.

12. **Motion Video Remoting** characterizes how motion video content is carried across a network using Net2Display remoting.
13. **Universal Serial Bus (USB) Remoting Architecture** defines approach for transporting USB traffic within Net2Display remoting.
14. **Keyboard Remoting** defines the approaches used for transporting the different keyboard configurations remoted by Net2Display remoting.
15. **Pointer Remoting** defines how the location and icon are transported for variously attached pointing devices.
16. **Audio Remoting Architecture** defines the approach for remoting Audio Input and Output traffic.
17. **Extended Virtual Channels** defines the general structure used for transporting vendor specific remoting data.
18. **Principles of Operation** describes the general process for initializing and reconnecting Net2Display remoting instances.
19. **Node Association Management (NAM)** describes a set of state machines that can be used to characterize the general behavior of Net2Display Nodes.
20. **Host Operation** describes a set of state machines that can be use to characterize the behavior of a Net2Display Host Node.
21. **Client Operation** describes a set of state machines that can be use to characterize the behavior of a Net2Display Client Node.
22. **Configuration Management** describes the representation of Net2Display Nodes using XML.

Additional informative material on Net2Display remoting is included in Appendicies:

A. **Appendix: Net2Display Remoting Requirements** lists the requirements that were considered in the formation of the Net2Display remoting standard.

B. **Appendix: Design Decisions** describes the choices that were made in the course of the definition of Net2Display remoting and the rationale behind those choices.

C. **Appendix: Performance Monitoring Virtual Channel** characterizes issues relating to setting up a performance monitoring channel.

D. **Appendix: Motion Video Control Virtual Channel** provides the basic facilities needed for setting up a channel for controlling motion videos from the Client.

E. **Appendix: Serial Port Virtual Channel** provides information for remoting a serial port over Net2Display remoting.

F. **Appendix: Parallel Port Virtual Channel** provides information for remoting a serial port over Net2Display remoting.

G. **Appendix: Network Setup for Net2Display Remoting** gives the advance configuration needed for using Net2Display in an infrastructure environment.

H. **Appendix: XML Descriptions for Example Configurations** is a placeholder for examples of XML descriptions of Net2Display Nodes.

I. **Appendix: Recommended Security Certificate Authorities and Certificates for Net2Display** recommends the Certificate Authorities to use for Net2Display remoting.

1.8 Definitions and Abbreviations

Active Client: A Client operating in Active Mode may be referred to as an Active Client or Active Mode Client.

Active Mode Client: A Client operating in Active Mode may be referred to as an Active Mode Client or Active Client.

Active Mode: The Active Mode is a Client Mode where the user provides authentication, typically hosting the connection of a keyboard and pointer. A Client in Active Mode initiates a connection to a Host, called an Association. A Client in Active Mode may be a Hub or may include an Integrated Display and may allow the connection of one or more separate displays. A Client operating in Active Mode may be referred to as an Active Client or an Active Mode Client.

Aggregated Client: An Aggregated Client is a grouping of multiple Clients that together function as a single Client with multiple displays.

Aggregated Coordinate Space: The Aggregated Coordinate Space is used to specify the relative location of display surfaces, motion videos and cursor location to coordinate this information between the Client and the Host.

Associate: The act of the Client or Host entering into an Association relationship.

Association: The relationship formed between a Net2Display Host and a Client so that the Client Display and I/O are connected to the Host computer.

Association Establishment: The final step in the Initialization Process that forms an Association between a Client and a Host.

CAM: Client Association Management

Canonical Name: The canonical name is an identifier that uniquely identifies a device that sources a Virtual Channel.

Client: A system, device or software that translates the networked Net2Display protocol into display output signals and provides for the attachment of user I/O devices. A Client may be any system, device or software that receives remote commands from the source Host and converts those commands into a visible image (e.g. computer monitor, television, etc). A Client may operate as an Active Client or as a Passive Client.

Client Association Management (CAM): Client Association Management controls the operation of a Client End, the Client instance displaying remote content from one Host.

Client Configuration: The Client instantiation in a system either as a hardware device, which may include associated user I/O, or as a software application supporting the remoting of the display and user I/O devices.

Client Displays: The Client Displays show the display content that is remoted from the Hosts. There are one or more Client Displays for a Client System.

Client End: A Client End is the single instance of the Net2Display Remoting protocol that implements the sourcing end of the display and communicates with the Host on the other end of the Connection. There may be multiples instances of Client Ends in a Client Node.

Client Management: A management entity that exists at the Client end of a Net2Display Association. There is one instance of Client Management in the Client for each Association to a Host.

Client Mode: A Net2Display Client may be designed to operate in a limited set of Client modes and needs to clearly indicate its Net2Display capabilities.

Client Node: The Client Node is the full Net2Display Client instance that resides at the Client System and includes the Client Ends of all of the Net2Display connections to remote Clients. The Client Node may contain multiple Client Ends.

Client Services: The capability to connect to Net2Display Hosts which may be advertised on the network or registered with a Connection Management Server. Typically, Clients in Passive mode will answer requests for Client Services.

Client System: The entire Client system, including the Net2Display Remoting, located at the Client end of a Net2Display Association. The Client System may be a PC or a display with an embedded Net2Display Client.

CMS: Connection Management Server

Configuration Information: Configuration Information is the information contained within Net2Display Hosts and Clients that expresses the Node's capabilities and is exchanged during Association.

Connection Management: The process of forming and maintaining an Association between a Client and Host.

Connection Management Server: A network attached server that provides the location of a Host computer for a given Client. The operation of the Connection Management Server in selecting a Host is beyond the scope of this standard, e.g., the Connection Management Server may use load balancing to optimize the selection of Host computing resources. The Connection Management Server may also be known in the industry as a connection broker.

Control: Net2Display network traffic that contains display formatting commands or flow control information.

Control Channel: The Net2Display communication path for passing control information via Control PDUs.

Daisy Chained Displays: A means for connecting displays where three or more Displays are connected to a single Client in a linear arrangement so that all of the display signals pass over the same display connection.

Data: Net2Display network traffic that contains display or USB content within the body of the command.

Device Authentication: verifying that a Net2Display endpoint (Client, Host or Connection Management Server) is the device it claims to be.

Device Driver Interface: the set of function calls to a graphics driver

DDI: Device Driver Interface

DiffServ: Differentiated Services networking architecture for providing quality of service

Discovery Process: The procedure used by a Clients and Hosts to locate each other. The Discovery Process is a stage in the Initialization Process for forming an Association.

Display Client: A Client Node that does not have any attached I/O and serves as a stand-alone display or serves to provide as an additional display to a Terminal Client. A Display Client does not typically provide USB functionality.

Display Connector: a generic term for a video output port that provides for connecting a display device. The Display Connector may support VGA, DVI, DisplayPort™ or another display interface.

Display Remoting: The portion of the Net2Display Remoting that provides the primitives and protocols for supporting the display at a remote distance.

Display Remoting Primitive: One of a set of commands that is used to represent changes on a display which are sent across the network to update the Client Display.

Display Wall: An arrangement of one or more Net2Display Clients to form a display area, typically permanently mounted on a horizontal surface.

Display Wall Element: A Net2Display Integrated Display Client without any attached I/O that is typically arranged in an array to form a larger display surface called a Display Wall. A Display Wall Element is a special type of Integrated Client which operates in Passive Mode.

Display Window: A display area that is remoted for a remoted display or remoted motion video.

DPVL: A VESA display interface standard that uses the DVI connector and allows the daisy chained connection of displays that support the interface.

DVI: Digital Visual Interface

End: An End is one termination of an Association. A Net2Display End may be a Host End or a Client End.

Extended Virtual Channel: A Virtual Channel used to encapsulate application or vendor specific data that is defined outside of the Net2Display standard.

Framebuffer: A region in graphics or systems memory that contains the complete frame of data for a video display, typically consisting of color values for every pixel. A Framebuffer is either: Off-screen, meaning that writes to the Framebuffer will not appear on the visible screen, or On-screen, meaning that writes to the Framebuffer will appear on the visible display.

Full Motion Video: Motion Video displayed at the full rate at which it was produced, which is typically 25 to 30 frames per second, depending upon the source and original media.

GDI: Graphics Device Interface

Graphics Device Interface: also known as the graphics engine, provides kernel-mode support for 2D graphics in the Windows® operating system environment through the Windows NT® operating system. The Windows® XP operating system graphics engine is known as GDI+.

HAM: Host Association Management

HD Video: any video system of higher resolution than standard-definition (SD) video, most commonly at display resolutions of 1280×720 (720p) or 1920×1080 (1080i or 1080p).

HDCP: High-bandwidth Digital Content Protection is a form of digital copy protection developed by the Intel Corporation to prevent copying of digital audio and video content as it travels across a video connection. The specification is proprietary, and implementing HDCP requires a license.

Header Field Group: A collection of fields in the Net2Display header.

Host: A generic term used to refer to the sourcing end of Net2Display traffic that includes the Host Computer and the Net2Display Host Node.

Host Address: Host Address is IP addressing information that identifies the Host Node, which is either an IP address or an IP host name.

Host Association Management (HAM): Host Association Management controls the operation of a Host End, the Host instance serving remote data to one Client.

Host Display Interface: A software interface, hardware connector or storage location where the display contents can be captured for remoting to a Client.

Host Cursor: A Host Cursor is a cursor that is generated and moved at the Host.

Host End: A Host End is the single instance of the Net2Display Remoting protocol that implements the sourcing end of the display and communicates with the Client on the other end of the Connection. There may be multiples instances of Host Ends in a Host Node.

Host Management: Host Management is a management entity that exists at the Host end of a Net2Display Association. There is one instance of Host Management in the Host for each Association to a Client.

Host Node: The Host Node is the full Net2Display Host instance that resides at a Host Server and includes the Host Ends of all of the Net2Display connections to remote Clients. There is one Host Node for each Host Server which may contain multiple Host Ends.

Host Server: A Host Server is a computer or operating system instance that is capable of sourcing Net2Display Remoting traffic for display on Net2Display Displays. In general, this standard will use the term "Host Server" to refer to the image source (may be a PC, workstation, set-top-box, etc).

Host Services: Host Services are the capability to connect to Net2Display Clients which may be advertised on the network or registered with a Connection Management Server.

Host System: The entire computing system, including the Net2Display Remoting, located at the Host end of a Net2Display Association.

Hub: A Client device that connects to a network that produces DVI or VGA output for the connection one or more Legacy Displays and optionally includes one or more USB connectors for the connection of USB devices. This would typically be a small adapter box. A Hub may optionally support multiple displays.

IETF: Internet Engineering Task Force

Initialization Process: The procedure used by Clients and Hosts to become Associated with each other so that the Client is displaying information from the Host. The Initialization Process is the startup sequence within Connection Management.

Input/Output (I/O): USB devices used for user input and user connected storage and devices

Input Facilities: Input Facilities include the keyboard, touch screen, mouse, pointer, smart card reader or USB key fob typically connected to a Client for input of Host or Connection Management Server address or authentication information.

Integrated Display: An Integrated Display is a Net2Display Client that has a built-in display and connects directly to a network. An Integrated Display may optionally have USB ports for connecting I/O devices or Display Connectors for connecting additional displays.

Integrated Projector: An Integrated Projector is a special case of an Integrated Display, where the display is a projection device.

Internet Protocol: a network layer protocol used for all Net2Display traffic

IP: Internet Protocol

LDAP: Lightweight Directory Access Protocol

Legacy Display: A display device that has VGA or DVI as display input.

LLMNR: Link-local Multicast Name Resolution

Local Cursor: A cursor icon that is generated and moved at the Client system to lessen the effects of network latency.

Managed Client: A Managed Client is a Client typically configured by an IT administrator for a managed set of remoting and I/O capabilities.

mDNS: Multicast DNS

Motion Video: Video that displays a sequence of images (frames) rapidly enough that the eyes see the image as a continuously moving picture. A Motion Video Virtual Channel uses a single encoding format at one time.

N2D: Net2Display

NAM: Node Association Management

NAT: Network Address Translation

NCM: Node Control Management

Net Display: A remoted display area that is remoted using a single Net Display Virtual Channel. The Net Display area may include both visible areas of the display called Viewports and invisible areas.

NIM: Network Interface Management

Node: A general term used for a Net2Display Client or Host, characterized by having a single IP Address.

Node Control Management (NCM): Node Control Management coordinates the operation of a single Net2Display Node and maintains the Node wide status and state.

Negotiation Process: The stage within the Initialization Process where Clients and Hosts exchange capabilities, such as compression, encryption and display commands, with each other and select which capabilities are used for the Association.

Net2Display Priority: The priority level of a particular data type when processed within Net2Display Remoting.

Net2Display Remoting: The system architecture and protocol used in connecting Hosts that support graphics remoting and Ports and Displays capable of supporting the displaying of the remoted graphics.

Network: Any homogeneous or heterogeneous computer interconnection capable of carrying Internet Protocol (IP) network traffic. The Network may include Ethernet segments, WAN links, wireless networks and DSL links, all which support IP traffic.

Network Interface: Network Interface connects the Host or Client to one network.

Network Interface Management (NIM) works with each Network Interface to ensure that a unique IP address is assigned

Network Transport: The Network Transport is TCP, UDP or another transport. The Network Transport sits below Net2Display remoting in the protocol stack and encapsulates the Net2Display PDUs and transports them across the network.

NIC: Network Interface Card that connects the Host or Client to the network, synonymous with Network Interface

Node: A general term used for a Net2Display Client or Host characterized by having a single IP Address.

Node Association Management (NAM): Node Association Management manages Net2Display Nodes and manages the formation, maintenance and Associations betweens Clients and Hosts.

Passive Client: A Client operating in Passive Mode may be referred to as a Passive Client or a Passive Mode Client.

Passive Mode Client: A Client operating in Passive Mode may be referred to as a Passive Mode Client or a Passive Client.

Passive Mode: In a configuration where a user associates multiple Clients to the same Host, the Passive Mode is a Client Mode where the Client is added in addition to an Active Mode Client. A Client in Passive Mode may be an identical hardware device as an Active Mode Client, but is primarily identified by the role it assumes with respect to the Host. A Passive Mode Client may or may not have attached Input Facilities. A Passive Mode Client may include an integrated display and may allow the connection of one or more separate displays. A Client operating in Passive Mode may be referred to as a Passive Client or a Passive Mode Client.

PDU: Protocol Data Unit, a unit of communication between Net2Display entities

PHB: Per hop behavior

Port: An endpoint for TCP and UDP communications.

Port Number: A numerical identifier for the endpoint for TCP and UDP communications. Net2Display remoting has a registered Port Number of 9086 which is used by default for the Host Node Port.

Primary Transport: The Primary Transport is the Network Transport where the Association is initially setup and the Control Channel is contained.

Protocol Data Unit: A unit of information that is delivered in a single unit between peer Net2Display entities of a network that may contain control information, address information, or data.

RFC: Request for Comments (RFC) is a document describing an Internet technology issued by the Internet Engineering Task Force (IETF).

RR: Directory Name Services Resource Records

SCTP: Stream Control Transmission Protocol is a transaction oriented network transport protocol.

Secondary Transport: A Network Transport in addition to the Primary Transport, setup to provide different Network Transport characteristics or priority.

Secure Transport: A transport level security protocol like IPsec that securely encapsulates all Net2Display data traffic for transport across the network.

Security Parameters Index: The Security Parameters Index (SPI) is an IPsec parameter that helps the recipient select which of possibly many ongoing conversations this packet applies. Each AH-protected connection implies a hash algorithm (MD5, SHA-1, etc.), some kind of secret data, and a host of other parameters. The SPI can be thought of as an index into a table of these settings, allowing for easy association of packet with parameter.

Single Sign-on: Single Sign-on (SSO) is a method of access control that enables a user to authenticate once and gain access to the resources of multiple software systems.

SLP: Service Location Protocol

Socket: a local end-point of a bidirectional communication flow between a local and a remote pair of processes. Each socket is typically referred to by a unique number by the operating system. A socket is a unique combination of the protocol (TCP, UDP or raw IP), local IP address, local Port Number, remote IP address and remote Port Number.

SRV RR: SRV Type Resource Record has DNS type code of 33

SSDP: Simple Service Discovery Protocol

Target Locator Address: A domain name or IP address given to a Client that provides a place to go to find a Net2Display Host. The Target Locator Address may be the address of a Connection Management Server or a Net2Display Host.

Terminal Client: A Client Node with attached I/O devices that typically allow user interaction. A Terminal Client provides USB and Audio functionality.

TCP: Transmission Control Protocol is a reliable transport protocol described in multiple specifications as summarized in IETF RFC 4614.

Timestamp: A parameter transmitted along with media data to indicate the relative time when the data was generated and to give guidance to when the data should be presented on the Client.

UDP: User Datagram Protocol is an unreliable datagram protocol specified in IETF RFC 768.

Unmanaged Client: An Unmanaged Client is a Client device setup by the end user with unrestricted Host and I/O remoting.

uPNP: Universal Plug and Play

VC: Virtual Channel

VC_Timestamp: A parameter transmitted along with media data to indicate the relative time when the data was generated and to give guidance to when the data should be presented on the Client. The specific parameter is called the VC_Timestamp since it is associated with a Virtual Channel.

Viewport: The visible area of a display surface.

Virtual Channel: A communication path between an Associated Host and Client that allows the passing of control or data. There are multiple Virtual Channels between an Associated Host and Client to allow different priorities of Data and Control to travel without blocking higher priority traffic. A single Virtual Channel uses the same Transport protocol in both directions and operates at a single priority level.

Virtual Channel Open Process: The steps used in the formation of a Virtual Channel within an existing Association.

Virtual Channel Parameters: The set of parameters that define the characteristics of a Virtual Channel.

Wallclock Time: The absolute time and date, expressed relative to 0h UTC 1 January 1900, used as a time reference. The Wallclock time may be produced by the Network Time Protocol or may be a locally generated time reference.

Windows Presentation Foundation is the graphics stack in the Windows Vista® operating system.

WPF: Windows Presentation Foundation

WSDL: Web Services Definition Language

XML: Extensible Markup Language

XML Schema: A specification of a set of rules to which an XML document conforms in order to be considered valid.

1.9 Conformance Glossary – Definition of Terms

The following is a list of definitions for certain keywords used through this document:

must: A keyword that indicates a mandatory requirement for compliance with this standard.

should: A keyword that indicates a choice with a strongly preferred preference – equivalent to "is strongly recommended

may: A keyword that indicates a choice with no expressed or implied preference

1.10 Conventions

In the presentation of diagrams, dashed lines are used to indicate optional entities, data paths, transitions and states.

Capitalized terms used in the document have the specific meanings as defined in Section 1.5. Non-capitalized usage of any of these terms should be understood as a reference to their usual English meaning.

The notation 0xABCD will be used for hexadecimal representation and 0b0101 for binary representation. Spaces may be added to ease interpretation of fields.

In state diagrams, conditions for state transitions are shown next to the transition arrows. Any actions on state transitions are shown below the transition condition. Actions on entry into states are shown within the state block below the name of the state.

1.11 Reference Documents

Table 1-2: Reference Documents

Document	Version / Revision	Date
DNS SRV (RFC 2782) Service Types (http://www.dns-sd.org/ServiceTypes.html)		
Digital Visual Interface (DVI)	Rev. 1.0	April 2, 1999
IETF RFC 768: User Datagram Protocol	Version 1	August 28, 1980
IETF RCF 1034 Domain Names - Concepts and Facilities	Version 1	November 1987
IETF RFC 1035: Domain Names – Implementation and Specification	Version 1	November 1987
IETF RFC 1122: Requirements for Internet Hosts -- Communication Layers	Version 1	October 1989
IETF RFC 1123: Requirements for Internet Hosts -- Application and Support	Version 1	October 1989
IETF RFC 2104: HMAC: Keyed-Hashing for Message Authentication	Version 1	February 1997
IETF RFC 2131: Dynamic Host Configuration Protocol	Version 1	March 1997
IETF RFC 2462: IPv6 Stateless Address Autoconfiguration	Version 1	December 1998
IETF RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers	Version 1	December 1998
IETF RFC 2475: An Architecture for Differentiated Services	Version 1	December 1998
IETF RFC 2608: Service Location Protocol, Version 2	Version 2	June 1999
IETF RFC 2782: A DNS RR for specifying the location of services (DNS SRV)	Version 1	February 2000
IETF RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	Version 1	April 2002
IETF RFC 3513: Internet Protocol Version 6 (IPv6) Addressing Architecture	Version 1	April 2003
IETF RFC 3550: RTP: A Transport Protocol for Real-Time Applications	Version 2	July 2003
IETF RFC 3551: RTP Profile for Audio and Video Conference with Minimal Control	Version 1	July 2003
IETF RFC 3715: IPsec-Network Address Translation (NAT) Compatibility Requirements	Version 1	March 2004
IETF RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses	Version 1	May 2005
IETF RFC 3947: Negotiation of NAT-Traversal in IKE	Version 1	January 2005
IETF RFC 3948: UDP Encapsulation of IPsec ESP Packets	Version 1	January 2005
IETF RFC 4301: Security Architecture for the Internet Protocol		December 2005
IETF RFC 4303: IP Encapsulating Security Payload (ESP)		December 2005

Document	Version / Revision	Date
IETF RFC 4306: Internet Key Exchange (IKEv2) Protocol	Version 2	December 2005
IETF RFC 4346: The Transport Layer Security (TLS) Protocol	Version 1.1	April 2006
IETF RFC 4614: A Roadmap for Transmission Control Protocol (TCP) Specification Documents	Version 1	September 2006
ITU-T T.120 Data protocols for multimedia conferencing		January 2007
Lei Zheng, "Implementing Remote Display on Commodity Operating Systems," M.S. Thesis, Columbia University, http://ncl.cs.columbia.edu/publications/zhangthesis.pdf		January 2006
Microsoft Developer Network: USB Structures and Enumerations (http://msdn.microsoft.com/en-us/library/ms793358.aspx)		
Microsoft Remote Desktop Protocol: Basic Connectivity and Graphics Remoting Specification	V20080207	February 7, 2008
OpenGL® 2.1 Specification	Version 2.1	July 30, 2006
The PS/2 Keyboard Interface by Adam Chapweske		
Universal Serial Bus Specification 2.0	Revision 2.0	
Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)	Version 1.11	June 27, 2001
Universal Serial Bus (USB) HID Usage Tables	Version 1.12	October 28, 2004
USB/IP Project (http://usbip.sourceforge.net/)		
VESA Digital Packetized Video Link (DPVL) Standard	Version 1	February 2006
VESA DisplayPort Standard	Version 1.1a	January 11, 2008
VESA Enhanced Display Data Channel (E-DDC) Standard	Version 1.2	Dec. 26, 2007
VESA Enhanced Extended Display Identification (E-EDID) Standard		
Web Services Description Language (WSDL) Version 2.0 Part 0: Primer	Revision 2.0	March 27, 2006
Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language	Revision 2.0	March 27, 2006
Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts	Revision 2.0	March 27, 2006
XML Schema Part 0: Primer	Revision 1.0	April 7, 2000
XML Schema Part 1: Structures Second Edition	2nd Edition	October 2004

2 Overview

Note: This Overview section describes how Net2Display works at a high level and does not define specific requirements for conformant implementations.

Net2Display Remoting is designed to meet existing and future display remoting needs. To consider how Net2Display Remoting would be used and configured, scenarios for the application of Net2Display are enumerated, the basic Net2Display configuration summarized and the different Client Configurations characterized.

The basic structure for Net2Display Remoting is shown in Figure 2-1: A basic Net2Display topology consists of a Host, a Client and an optional Connection Management Server. A Network Interface Card (NIC) connects the Host or Client to the network. Both the Net2Display Host and Client are referred to as Nodes, a general term for network devices capable of sourcing or receiving Net2Display data traffic. The parts of the Net2Display basic structure are described in the following sections.

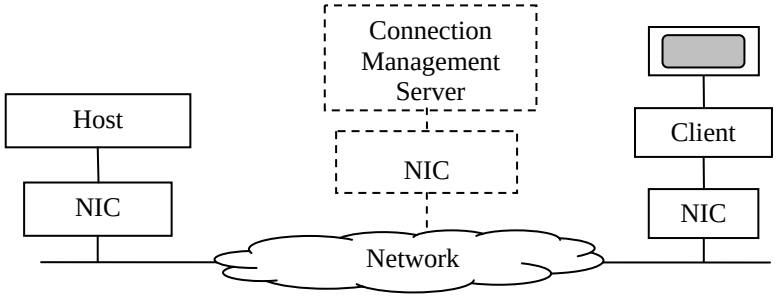


Figure 2-1: Basic Topology

2.1 Host

The Host is a computer, operating system instance or application that is capable of sourcing Net2Display Remoting traffic over a Network. The Host may be an entire computer system that is interfaced remotely using the Net2Display Remoting protocol or may be certain users, operating system partitions or applications that are remoted.

2.2 Network

The Network is any homogeneous or heterogeneous computer interconnect capable of carrying Internet Protocol (IP) network traffic. The Network may include Ethernet segments, WAN links, Wireless networks, DSL links, or any other networking segment that supports IP traffic. IP networks are capable of carrying both reliable Transmission Control Protocol (TCP) and unreliable User Datagram Protocol (UDP) traffic.

2.3 Client

The Client is a system, device or software that translates the networked Net2Display Protocol into display output signals and provides for the attachment of user I/O devices. The Client may be a stand-alone Display device or may be a software application that runs on a conventional PC or notebook PC.

2.4 Connection Management Server

A Connection Management Server (CMS) is a network attached server that provides the location of a Host computer for a given Client and may be capable of network or processing load balancing across multiple Hosts. The Connection Management Server is shown in dashed lines because it is an optional part of a Net2Display topology.

A Connection Management Server receives requests from Net2Display Clients and responds providing an available Host for Association. The Connection Management Server gathers information on available Hosts and selects the most appropriate Host to associate with the Client, all in a manner outside the scope of this standard. The communication between the Client and the Connection Management Server is specified within this standard, while communications between the Connection Management Server and Hosts is outside of the scope of this standard.

2.5 Net2Display Remoting

The Net2Display remotes displays by 1) remoting the display content from the Host to the Client Display and 2) optionally remoting I/O devices attached to the Client to the Host. Additionally, a Host Management block is present at the Host end of the Association and a Client Management block is present at the Client end to manage the Association between Host and Client. An optional Connection Management Server provides the service of locating Host Computer resources for Clients. These elements are shown in Figure 2-2.

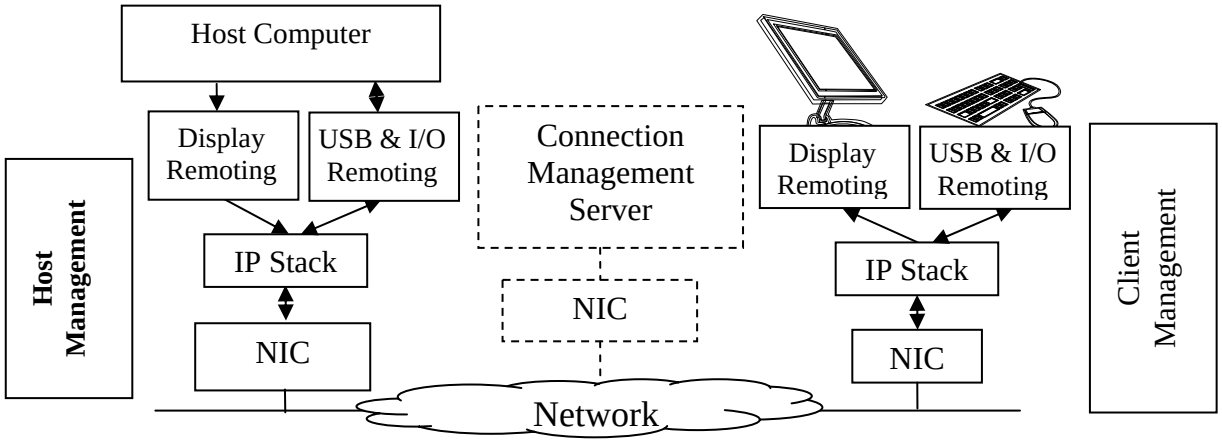


Figure 2-2: Net2Display General Structure

Net2Display is specified to allow the support of a varied set of topologies to meet the needs of the scenarios described in Section 1.5. Examples of these topologies are shown in Figure 2-3: Simple topologies that consist of a single Host and Client connected together are topologies Figure 2-3: a), b), c), and d). Multiple Client devices may be connected to the same Host as shown in Figure 2-3: e), f) and g). Additionally, display information from multiple Hosts may be displayed by one Client as shown in Figure 2-3: h) either simultaneously on different areas of the display or in a switchable manner where the display information from only one Host is visible at a time.

Multiple display user stations may be formed using several different configurations including: a multi-headed Client (Figure 2-3: c)); a Display Port based Client (Figure 2-3: d)) which uses daisy-chaining to support multiple displays; or using multiple separate Clients (shown in Figure 2-3: e), f) and g)) to provide multiple displays. Using multiple separate Clients to constitute a single user station can be configured either by: 1) using separate Associations of each separate Clients to the Host, typically through login processes, or by 2) providing a mechanism for associating multiple Client devices together to constitute an Aggregated multiple Display Client. Specific configurations for Net2Display Remoting supporting multiple displays are discussed in Section 3.3.8.3.

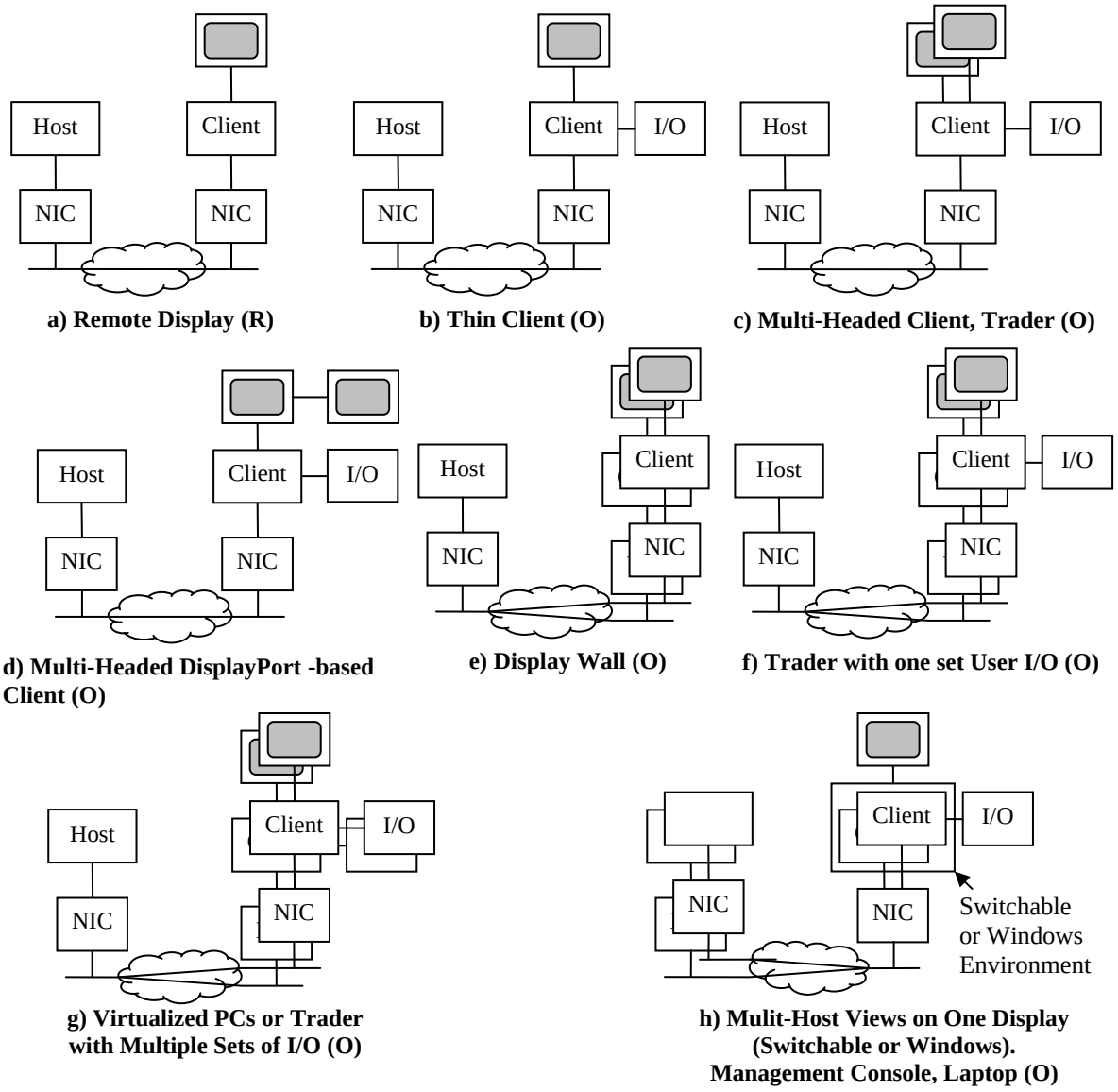


Figure 2-3: Net2Display Example Topologies

2.6 Virtual Channels

A Virtual Channel is a communication path between an Associated Host and Client that allows the passing of control or data. Net2Display Remoting uses multiple Virtual Channels between an Associated Host and Client to allow different priorities of data and control to travel without blocking higher priority traffic. A single Virtual Channel uses the same Transport protocol in both directions and operates at a single priority level.

3 Nodes

In general, a Net2Display Node may have one or more Host End instances, one or more Client End instances, and one or more Network Interfaces. Examples of multiple Network Interfaces include:

- Laptop with Ethernet and wireless interfaces which switches between the two depending upon network availability
- Dual Network Interface server, which has two Network Interfaces for reliability and bandwidth. Net2Display should be able to use whichever of the Network Interfaces it chooses.
- Dual NIC Integrated Display, with both Ethernet and wireless Network Interfaces. Net2Display should choose which ever one is connected allowing both wired and wireless operation
- A PC with multiple Ethernet Network Interfaces with one dedicated Network Interface for attaching displays using Net2Display Remoting.

To support the flexibility of a system both sourcing and displaying remote content, both Host and Client End instances may reside in the same Net2Display Node.

3.1 General Node Structure

The different functions of Net2Display can be broken into a number of distinct structures that perform separable functions in the overall Net2Display function. These structures are described in different sections in this document. Both Net2Display Clients and Hosts have the same general structures that make up a Net2Display Node as shown in Figure 3-1.

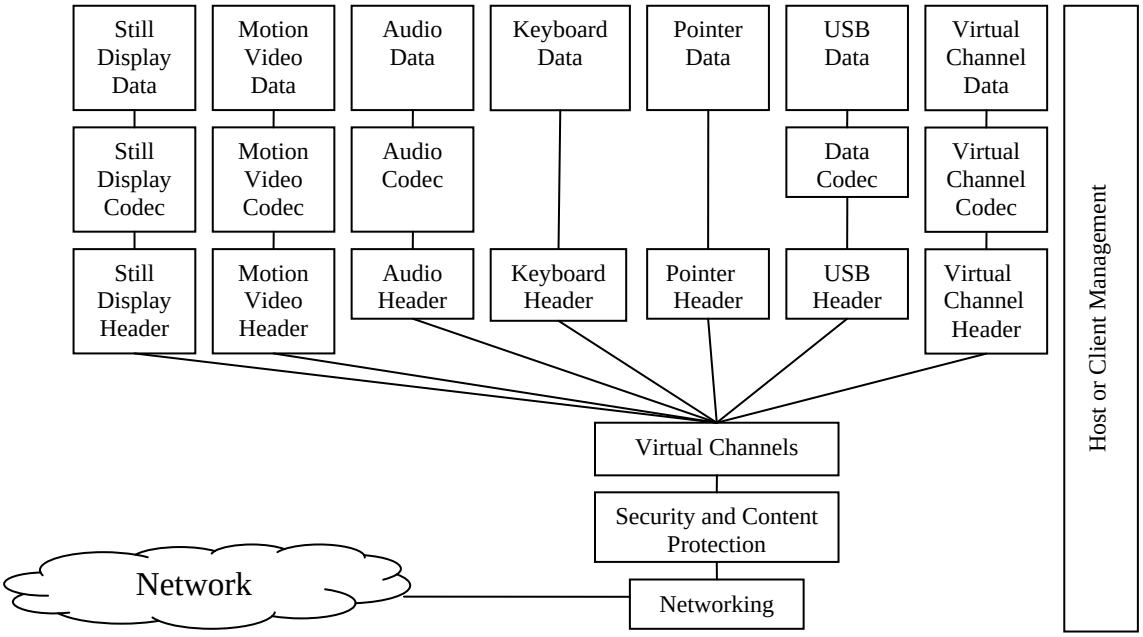


Figure 3-1: Net2Display General Node Structure

Net2Display is architected to be flexible in the codecs and compression that it allows. As it transfers a number of different types of data including static images, motion video, audio, and USB data, a number of different types of compression are defined for use with Net2Display data. For an acceptable performance for Net2Display interoperability, all Net2Display instantiations are required to support certain required compression types, but may support other additional more complex or vendor defined compression approaches.

3.2 Hosts

A Net2Display Host is a computer or operating system instance that is capable of sourcing Net2Display Remoting traffic for display on Net2Display Displays. In general, this standard will use the term “Host” to refer to the image source (may be a PC, workstation, set-top-box, etc). Net2Display Remoting is processor and operating system independent, able to remote displays and I/O from any Host system that supports IP networking and USB I/O. A simple representation of a Host as used in topology diagrams is shown in Figure 3-2.

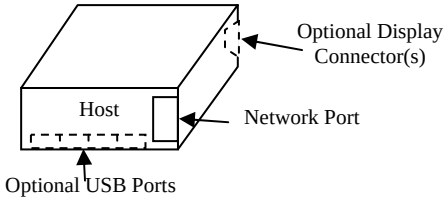


Figure 3-2: Host System

Within the Net2Display standard, different designations are used for: 1) the computer system or operating system instance that produces the display output to be remoted, 2) the Net2Display instance that includes all of the hosted Associations from the system and manages them and 3) the Host end of the Net2Display connection that goes to a single Client. These are designated as follows:

- **Host System:** The entire computing system, including the Net2Display Remoting, located at the Host end of a Net2Display Association.
- **Host Server:** A Host Server is a computer or operating system instance that is capable of sourcing Net2Display Remoting traffic for display on Net2Display Displays. In general, this standard will use the term “Host Server” to refer to the image source (may be a PC, workstation, set-top-box, etc).
- **Host Node:** The Host Node is the full Net2Display Host instance that resides at a Host Server and includes the Host Ends of all of the Net2Display connections to remote Clients. There is one Host Node for each Host Server which may contain multiple Host Ends.
- **Host End:** A Host End is the single instance of the Net2Display Remoting protocol that implements the sourcing end of the display and communicates with the Client on the other end of the Connection. There may be multiples instances of Host Ends in a Host Node.

The relationships of these different Host entities are illustrated in Figure 3-3.

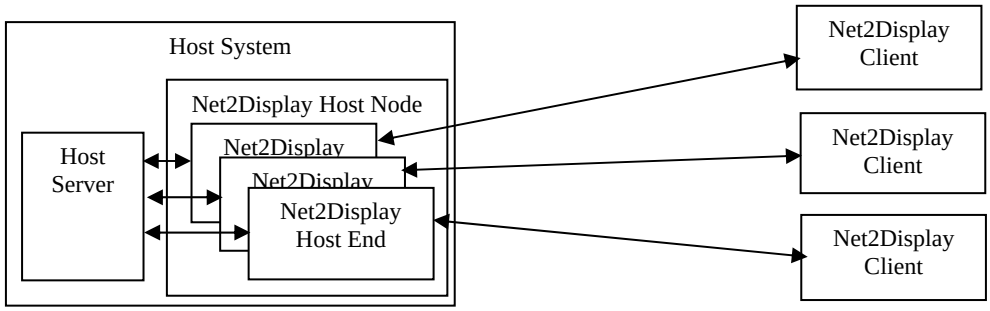


Figure 3-3: Host Structures

3.2.1 Host Structure

The Net2Display Host sends display content and USB to a remote Net2Display Client. The core components of a Net2Display Host are the facilities for remoting display content, remoting USB and connection Control. These facilities are illustrated in an example of a Host shown in Figure 3-4:.

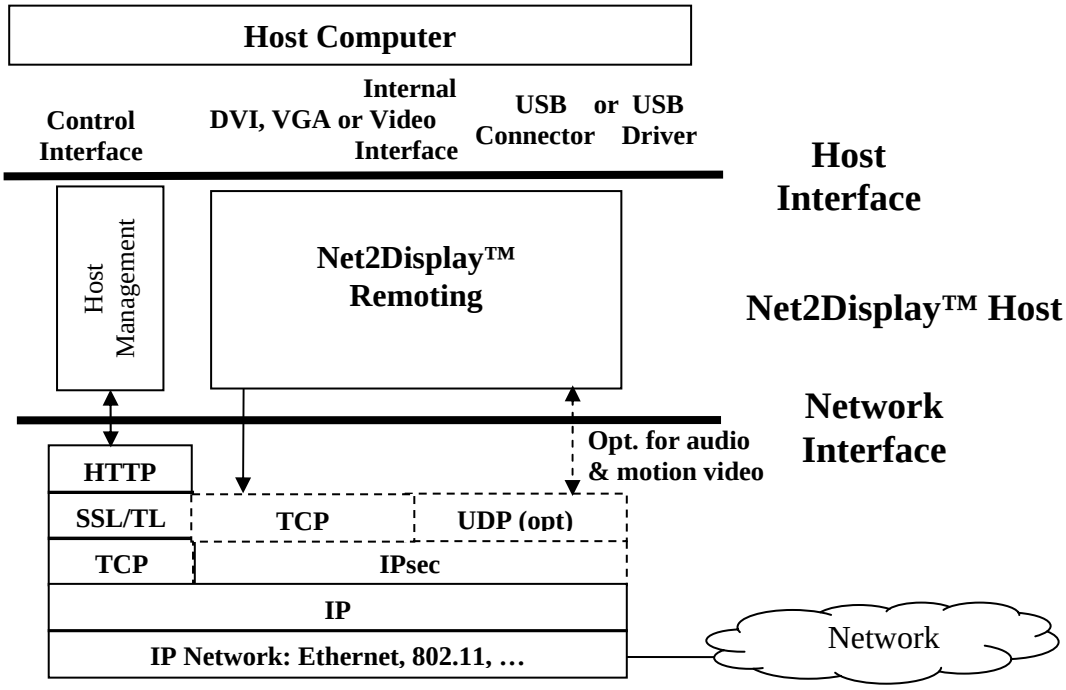


Figure 3-4: Net2Display Host Block Diagram Example

3.2.2 Host Interfaces

There is an interface that exists between the Net2Display Host Instance and the Host Computer. Examples of information that is passed across this Host interface include:

- Number of Displays remoted from the Client
- Display Information
- Send Display Data
- Keyboard and Pointer data
- USB Send and Receive

3.2.3 Host Model

A high level model of a Net2Display Host is shown in Figure 3-5. The Net2Display Host serves to interface to the Host System and remote the data across an IP network to Net2Display Clients.

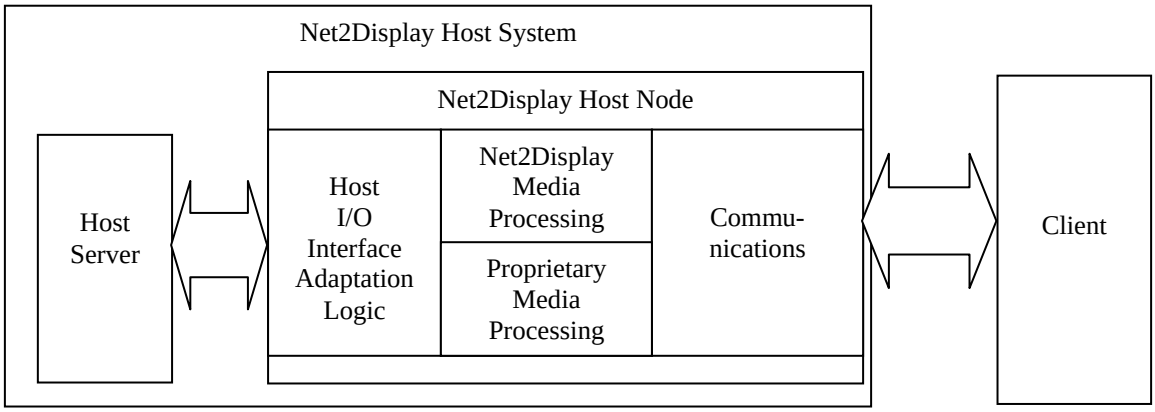


Figure 3-5: High Level Host Model

3.2.4 Client Aggregation

Multiple Clients may be optionally grouped together to behave as a single Client in a configuration called Client Aggregation which is illustrated in Figure 3-6. The requirements for Client Aggregation are given in Section 6.5. The mapping of displays in an Aggregated Client is characterized in Section 10.1.5 and the handling of pointers in an Aggregated Client is described in Section 15.5. The connection sequence of forming Associations with the multiple Clients that make up an Aggregated Client is described in Section 18.1.3. The operation of Client Aggregation is not further specified within the Net2Display Standard.

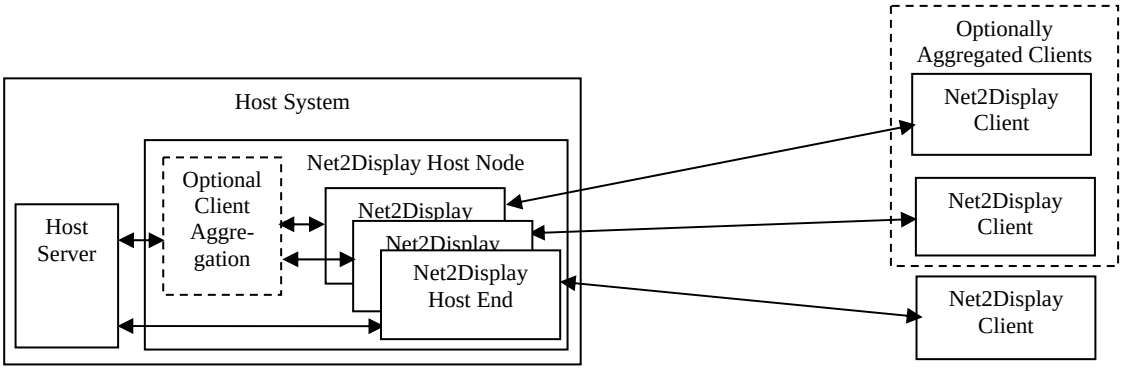


Figure 3-6: Host Structures with Optional Client Aggregation

3.3 Clients

A Client is a system, device or software that translates the networked Net2Display protocol into display output signals and provides for the attachment of user I/O devices. A Client may be any system, device or software that receives remote commands from the source Host and converts those commands into a visible image (e.g. computer monitor, television, etc). Net2Display Remoting is processor and OS independent so that Clients may contain any means capable of processing Net2Display traffic and may use any operating system or no operating system in the Client, depending on the specific Client system design decisions. A Client may operate as an Active Client or as a Passive Client. A Client system may optionally simultaneously maintain Associations to multiple Hosts. The organization of such a Client is illustrated in Figure 3-7 with the constituent structures:

- **Client System:** The entire Client system, including the Net2Display Remoting, located at the Client end of a Net2Display Association. The Client System may be a full PC or a dedicated Client device.

- **Client Node:** The Client Node is the full Net2Display Client instance that resides at the Client System and includes the Client Ends of all of the Net2Display connections to remote Clients. The Client Node may contain multiple Client Ends.
- **Client End:** A Client End is the single instance of the Net2Display Remoting protocol that implements the sourcing end of the display and communicates with the Host on the other end of the Connection. There may be multiples instances of Client Ends in a Client Node.
- **Client Displays:** The Client Displays show the display content that is remoted from the Hosts. There are one or more Client Displays for a Client System.

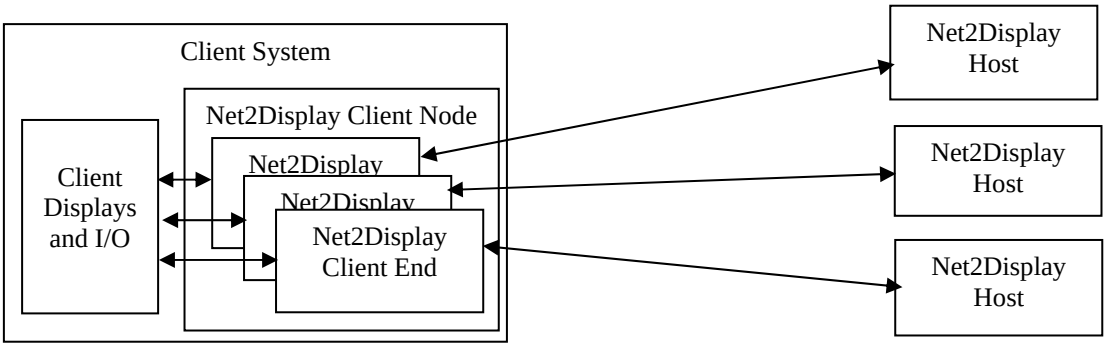


Figure 3-7: Client Structures

Net2Display Remoting enables a number of different Client configurations. The minimal Net2Display Client system configuration consists of a Network Interface Connection (NIC) and a Client instance. Examples of Net2Display Client configurations are shown in Figure 3-8. The gray enclosing boxes illustrate the portions of the system that are usually in a single hardware enclosure.

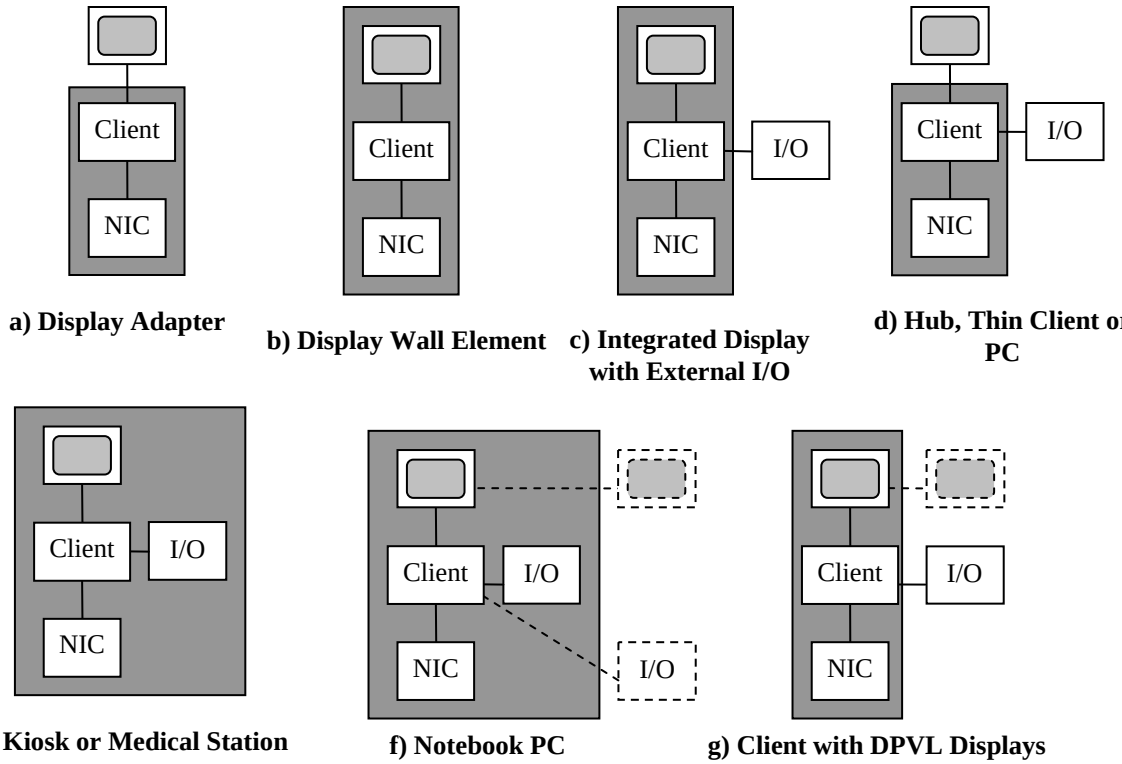


Figure 3-8: Client Configuration Examples

3.3.1 Client Structure

The Net2Display Client displays the remoted interface from the Net2Display Host. An example block diagram of a Net2Display Client is shown in Figure 3-9.

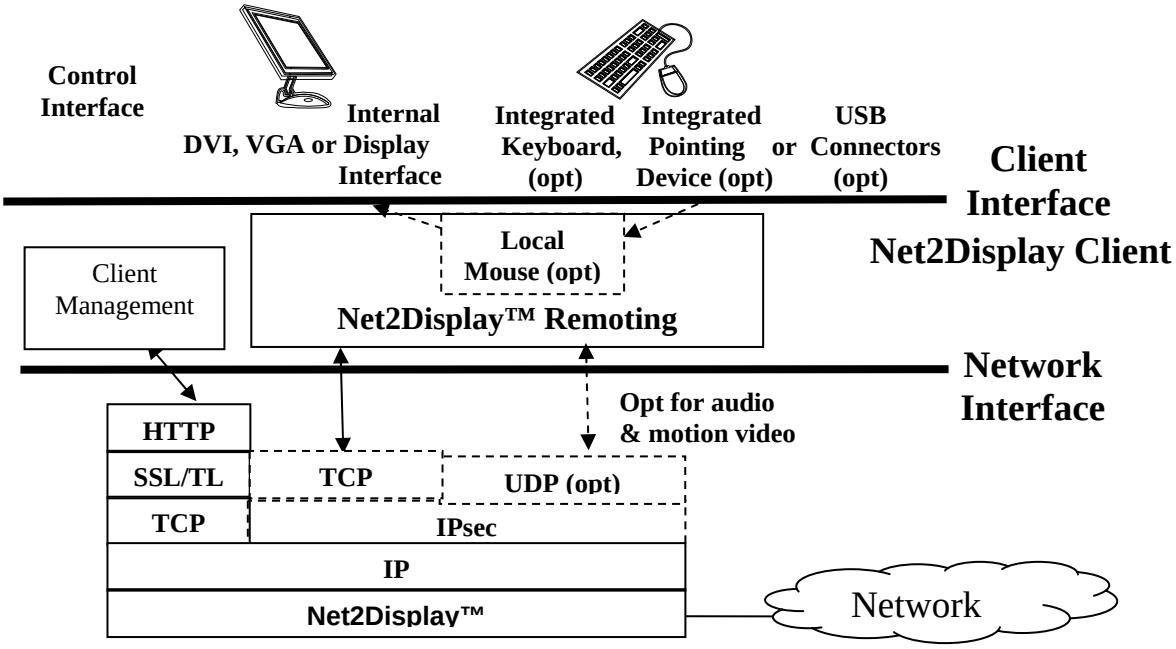


Figure 3-9: Net2Display Client Block Diagram Example

3.3.2 Client Interfaces

An interface exists between the Net2Display Client Instance and other parts of the Client. These interfaces allow the transfer of the necessary control information and display and I/O data for a remote Net2Display Association. Examples of information that is passed across the Client Interface are:

- Connect/Disconnect requests
- Reset/Initialize Client
- Set Client IP Address, Client Name, DNS Server, N2D_Server, Client Security, Number of Displays, Display Resolutions (Display #, Resolution)
- Send Keyboard and Pointer Data
- USB Data

3.3.3 Client Models

A high level model of a Net2Display Client is shown in Figure 3-10. The Net2Display Client serves to collect Client I/O and send it across an IP network to a remote Net2Display Host.

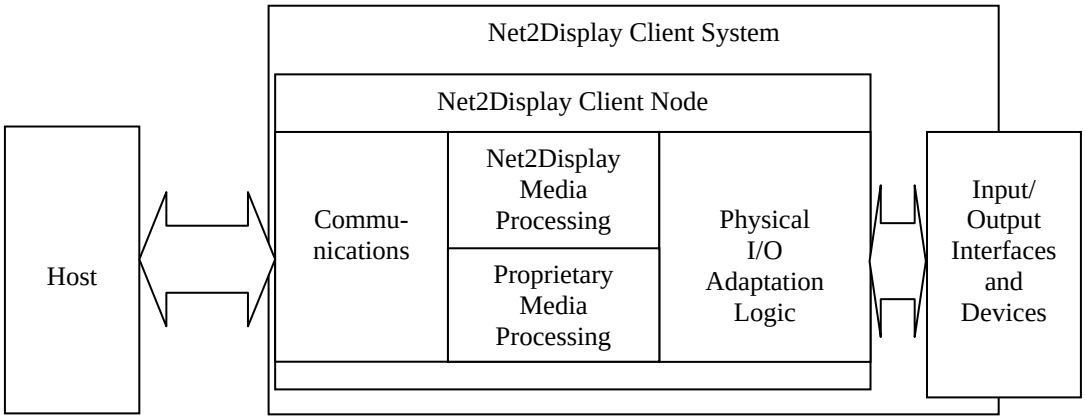


Figure 3-10: High Level Client Model

A more detailed model of a Client is shown in Figure 3-11 showing many of the Virtual Channels and their flow through the Net2Display Client. The Net2Display Client serves to receive and process the Net2Display network traffic in the different Virtual Channels and present it in the proper format to the different attached and embedded I/O devices.

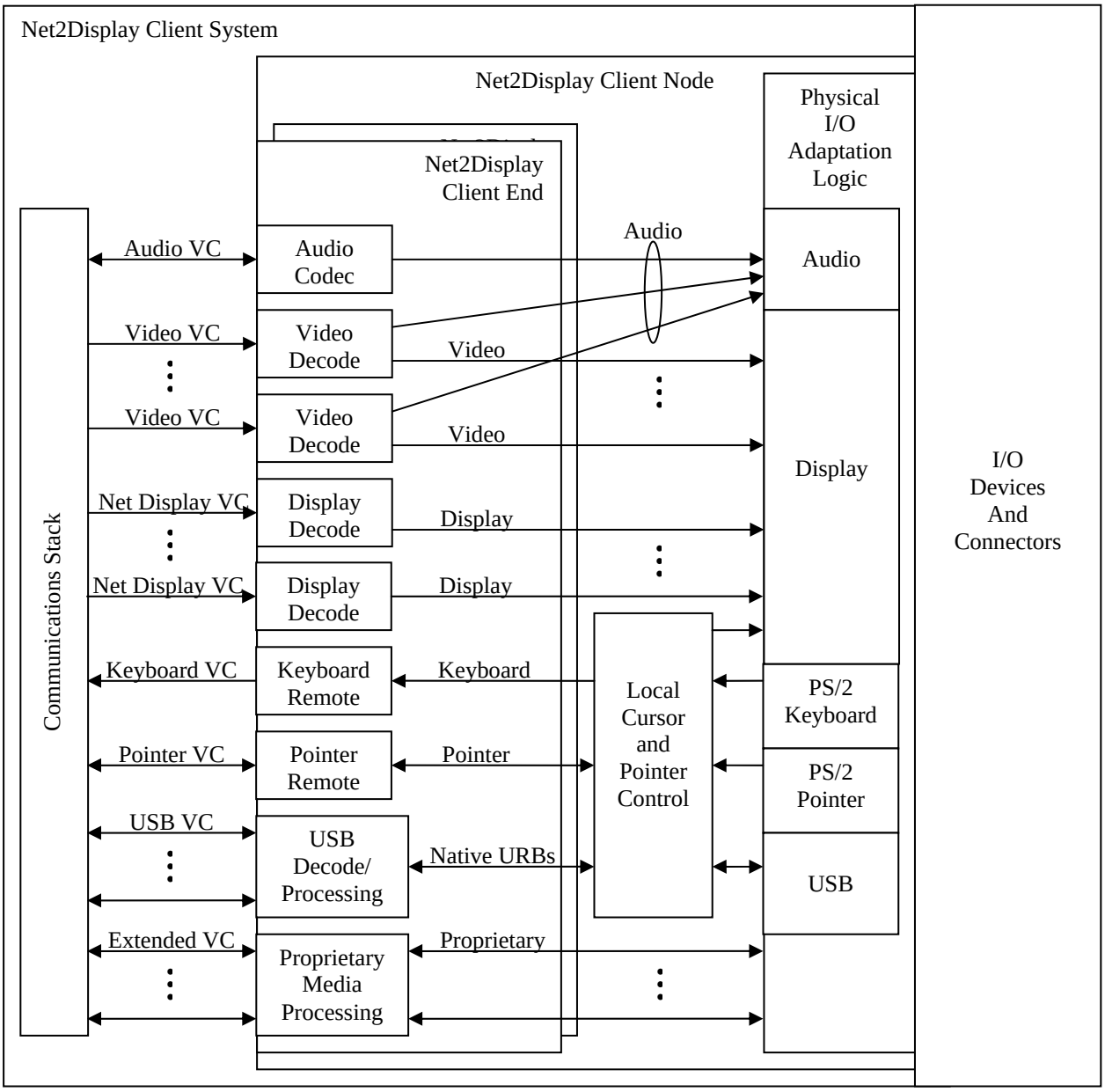


Figure 3-11: Client Model with Virtual Channels

3.3.4 Aggregated Client

An Aggregated Client is a Client that has been formed by two or more Client Displays that have become linked together so that they operate as a single Client with multiple displays. Aggregated Client configurations can be used for supporting a large number of connected displays as discussed in Section 3.3.8.3.2.

3.3.5 Client Modes

A Net2Display Client may be connected within an Aggregated Client to take on different roles in interacting with the Host, designated as Active Clients and Passive Clients.

3.3.5.1 Active Mode

The Active Mode is a Client Mode where the user provides authentication, typically hosting the connection of a keyboard and pointer. A Client in Active Mode initiates a Connection to a Host. A Client in Active Mode may be a Hub or may include an Integrated Display and may allow the connection of one or more separate displays. A Client operating in Active Mode may be referred to as an Active Client or an Active Mode Client. Some examples of ways that a Client operating in Active Mode may use to locate a Host for Association include:

- **Client and Host:** Client knows the Host name or address and connects to Host
- **Local Host Discovery:** Client finds the Host Service using local Discovery and connects to the Host. In this approach, the Host is advertising its Host Service.
- **Known Connection Broker provides Host:** Client knows Connection Broker name or address and requests then connects to Host
- **Local Discovery of Connection Broker:** Client finds Connection Broker using local Discovery and uses Connection Broker to find Host. For this approach, the Connection Broker is aware of the Host Service and informs the Client of the Host Service.
- **Recursive Connection Brokering:** Located Connection Broker provides Connection Broker
- **Discovery Server provides Host:** Active Client finds Host Service through Discovery Server in infrastructure then connects. For this approach, the Host is registered with the Discovery Server.
- **Discovery Server Provides Connection Broker:** An Active Client finds the Connection Broker through a Discover Server in the infrastructure and uses the Connection Broker to find the Host. For this approach, the Connection Broker is registered as a service with the Discovery Server and the Connection Broker is aware of the Host Service.

3.3.5.2 Passive Mode

In an Aggregated Client Configuration where a user Associates multiple Clients to the same Host, the Passive Mode is a Client Mode where the Client is added in addition to an Active Mode Client. A Client in Passive Mode may be an identical hardware device as an Active Mode Client, but is primarily identified by the role it assumes with respect to the Host. A Passive Mode Client may or may not have attached Input Facilities. A Passive Mode Client may include an integrated display and may allow the connection of one or more separate displays. A Client operating in Passive Mode may be referred to as a Passive Client or a Passive Mode Client. Likewise, some examples of ways that a Client operating in Passive Mode may use to become Associated with a Host include:

- **Host Requests Client to Connect:** Host knows Client name or address and requests Client to connect to Host
- **Local Client Discovery:** Host finds Client Service using local Discovery and requests Client to connect to Host. A Passive Client answers Host requests for Client Services.
- **Discovery Server provides Client:** Host finds Client Service through Discovery Server in infrastructure then requests the Client to connect to the Host. The Client answers Host requests for Client Services.

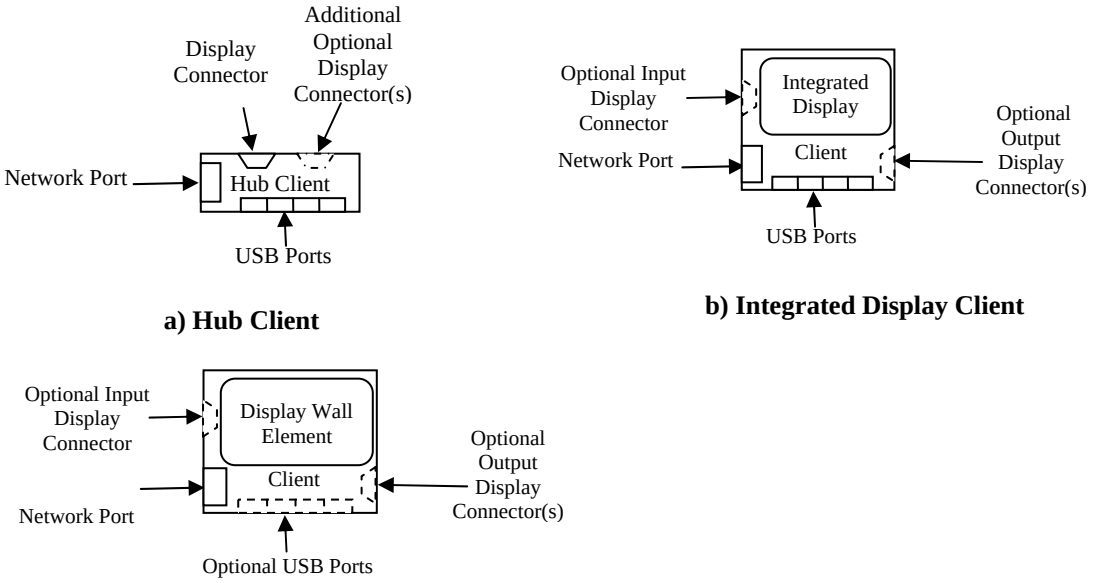
3.3.6 Managed and Unmanaged Clients

Net2Display provides for supporting both Managed Clients and Unmanaged Clients. A Managed Client is typically configured by an IT administrator for a managed set of remoting and I/O capabilities. An Unmanaged Client is a Client device setup by the end user with unrestricted Host and I/O remoting. Managed and Unmanaged Clients are not defined further in this standard.

Note: All Clients start out as Unmanaged Clients, but may be configured and locked down to be Managed Clients.

3.3.7 Client Type Examples

A Hub is a Net2Display Client device that connects to a network that produces DVI or VGA output for the connection one or more Legacy Displays and optionally includes one or more USB connectors for the connection of USB devices. This would typically be a small adapter box. A Hub may optionally support multiple displays as shown in the representation of the Hub Client shown in Figure 3-12a.



a) Hub Client

b) Integrated Display Client

c) Display Wall Element Client

Figure 3-12: Client Type Examples

An Integrated Display is a Net2Display Client that has a built-in display and connects directly to a network. An Integrated Display may optionally have USB ports for connecting I/O devices or Display Connectors for connecting additional displays. A representation of an Integrated Display is shown in Figure 3-12b.

A Display Wall Element is a Net2Display Integrated Display Client without any attached I/O that is typically arranged in an array to form a larger display surface called a Display Wall. A representation of a Display Wall Element Client is shown in Figure 3-12c.

3.3.8 Client Configuration Examples

Net2Display is used to support remote displays and provides the flexibility that existing display devices and thin clients have in display connections to local and remote computers. Net2Display is designed to allow it to be used to replace:

1) Display remoting currently done by RDP, ICA and VNC to thin clients and PCs
2) Display remoting as used for Display Walls
3) Displays directly attached to PCs

The following Client configurations and Discovery scenarios are described from a users' perspective of what is desired to happen.

3.3.8.1 Direct Display Attachment Configuration Examples

Net2Display may be used in multiple different ways to connect a Net2Display enabled display locally to a Host computer that supports Net2Display Remoting.

3.3.8.1.1 Single Integrated Display Direct Attach with no Input Facilities Example

The first configuration example is where a user with a new computer connects a Net2Display Integrated Display directly to the Ethernet port on the computer without a keyboard or any I/O connected to the display as shown in Figure 3-13. An Integrated Display is a Net2Display Client that has a built-in display and connects directly to a network. An Integrated Display may optionally have USB ports for connecting I/O devices or Display Connectors for connecting additional displays.

Input Facilities are connected directly to the Host computer. Input Facilities include the keyboard, touch screen, mouse, pointer, smart card reader or USB key fob typically connected to a Client for input of Host or Connection Management Server address or authentication information. The Input Facilities may be attached to the Client using USB or a serial port interface. Only the USB ports are shown in the following figures for the simplicity of illustration. The Input Facilities are represented as shown in Figure 3-13.

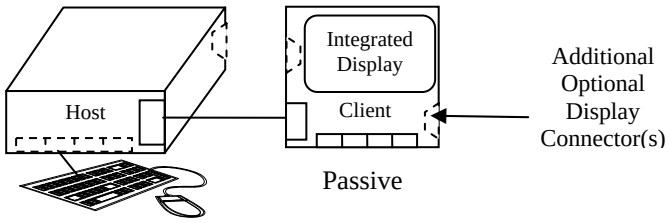


Figure 3-13: Single Integrated Display Direct Attachment with no Input Facilities Example

3.3.8.1.2 Single Integrated Display Direct Attach with Input Facilities Example

In the second example, the user has a new computer and connects a Net2Display Integrated Display directly to the Ethernet port on the computer with keyboard or other Input Facilities connected directly to the Integrated Display as shown in Figure 3-14.

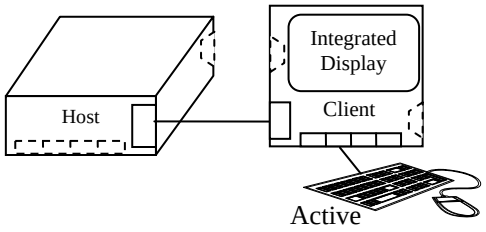


Figure 3-14: Single Integrated Display Direct Attach with Input Facilities Example

3.3.8.1.3 Multiple Client Direct Attach with Input Facilities Example

A network switch is used to attach multiple Net2Display Clients or Hosts to a single network access point and its representation is shown in Figure 3-15.

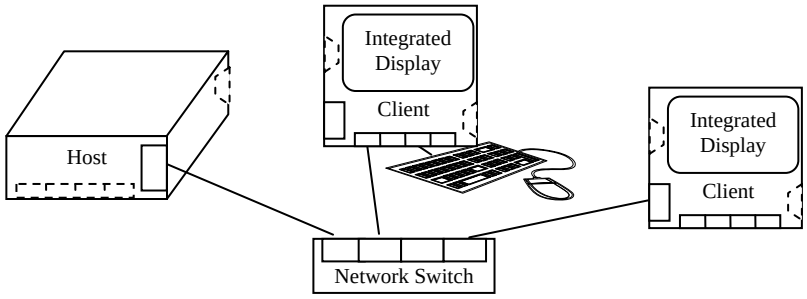


Figure 3-15: Multiple Client Direct Attachment with Input Facilities Example

3.3.8.1.4 Multiple Client Direct Attach without Input Facilities Example

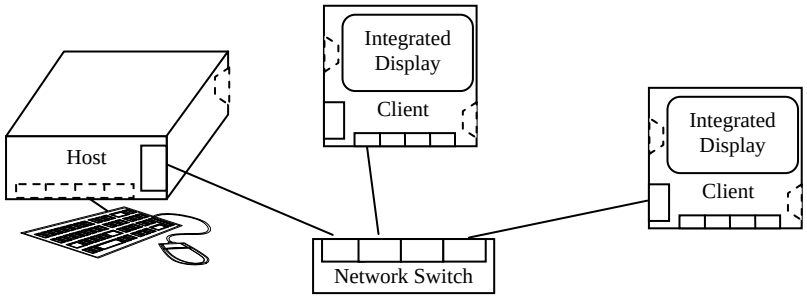


Figure 3-16: Multiple Client Direct Attachment without Input Facilities Example

3.3.8.2 Remote Display Attachment Configuration Examples

3.3.8.2.1 Remote Single Display Attachment Example

The simplest remoting configuration is a single Client remoted over a network. The Network may be any homogeneous or heterogeneous computer interconnection capable of carrying IP network traffic. The Network may include Ethernet segments, WAN links, Wireless networks and DSL links, all which support IP traffic. The Network may be over a local home network, a corporate network or over the Internet and may contain switches or routers. A simple cloud will be used to represent this general network as shown in Figure 3-17.

The remote configuration of Figure 3-17 is shown with a Hub Client and a Legacy Display. A Hub Client is a Net2Display Client device that connects to a network that produces DVI or VGA output for the connection one or more Legacy Displays and optionally includes one or more USB connectors for the connection of USB devices. This would typically be a small adapter box. A Hub may optionally support multiple displays. A Legacy Display is connected to the Hub Client in this example. A Legacy Display is a display device that has VGA, DVI, DPVL or an existing display interface standard as display input. The representation for a Legacy Display is shown in Figure 3-17.

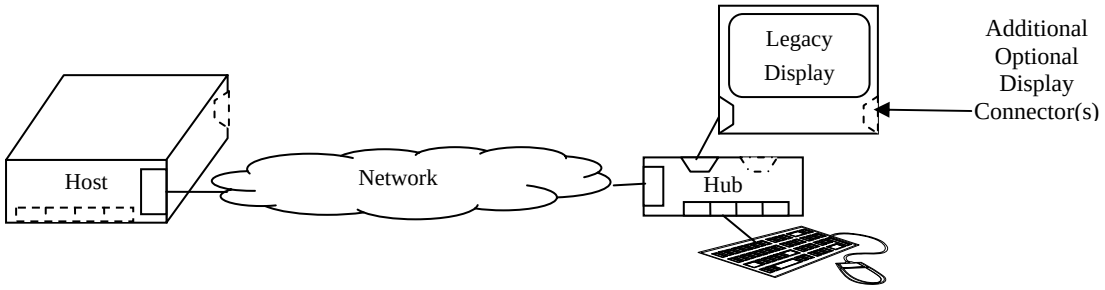


Figure 3-17: Remote Legacy Single Client Attachment Example

3.3.8.3 Remote Multiple Display Attachment Configuration Example

The remote attachment of multiple displays addresses the scenarios of Financial Trading Station, Professional Office Users and Graphics Station as described in Section 1.6 . There are several different ways that multiple displays can be presented to the same user, each with different attributes. These different configurations are shown in the following sections. They include:

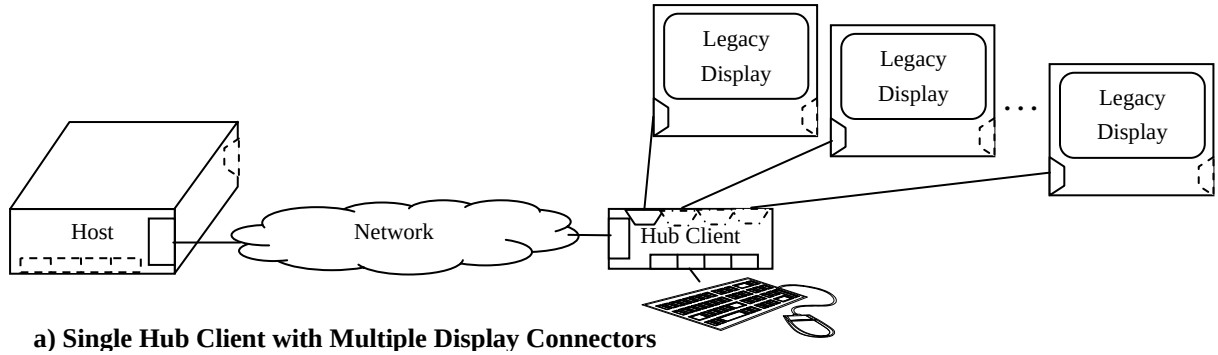
- Using a Single Hub Client with multiple output display connectors with attached displays
- Using a Single Hub Client with Daisy Chained Displays
- Using an Integrated Display with Directly Attached Displays
- Using an Integrated Display with Daisy Chained Displays
- Using multiple Hub Clients to connect multiple displays
- Using multiple Integrated Display Clients connected to the Network

All of these different approaches are supported by this Standard since they each have different advantageous attributes as shown in Table 3-1.

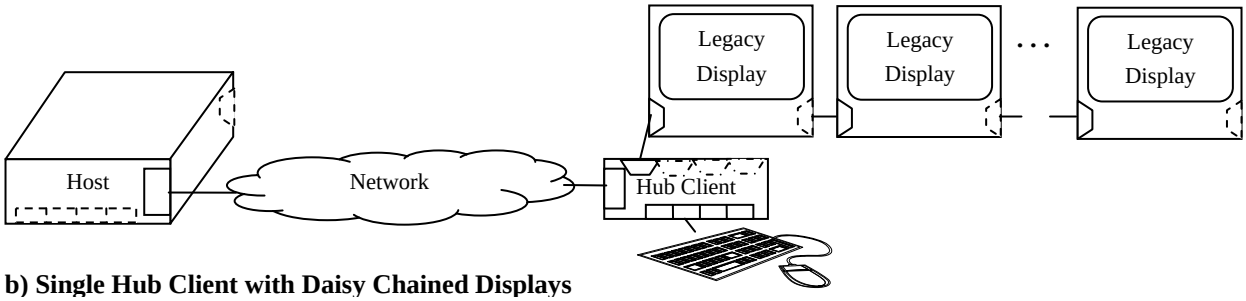
Table 3-1: Comparison of Multiple Display Approaches

Configuration	Advantage 1: Single Client to Manage	Advantage 2: Single Network Access	Advantage 3: Works with Existing Legacy Displays	Advantage 4: Full Network Bandwidth for each Client	Number of Displays Supported
Single Hub Client with multiple output display connectors with attached displays	Yes	Yes	Yes	No	Small
Single Hub Client with daisy chained displays	Yes	Yes	No	No	Medium
Integrated Display with Directly Attached Displays	Yes	Yes	No	No	Small
Integrated Display with Daisy Chained Displays	Yes	Yes	No	No	Medium
Multiple Hub Clients to connect multiple displays	No	No	Yes	Yes	Large
Multiple Integrated Display Clients connected to the Network	No	No	No	Yes	Large

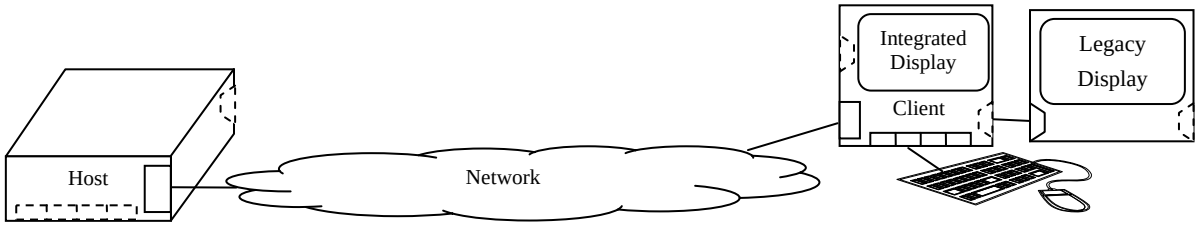
3.3.8.3.1 Single Client with Multiple Attached Displays Examples



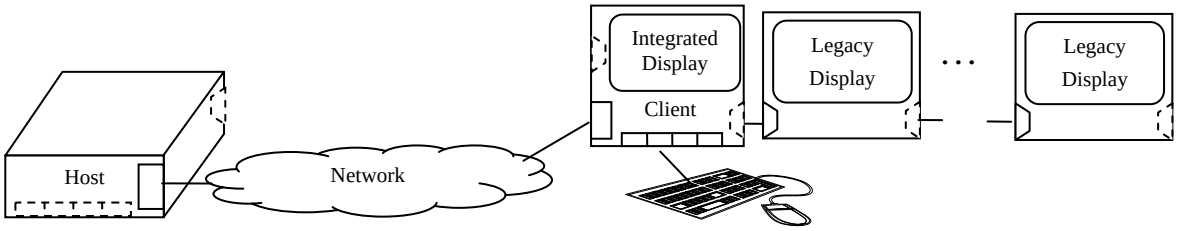
a) Single Hub Client with Multiple Display Connectors



b) Single Hub Client with Daisy Chained Displays



c) Single Integrated Client with Directly Attached Displays



d) Single Integrated Client with Daisy Chained Displays

Figure 3-18: Single Client with Multiple Attached Displays Examples

3.3.8.3.2 Multiple Displays using Multiple Clients Examples

Multiple Clients can be grouped together to function as one client with a large number of displays through the process of Aggregation as shown in Figure 3-19.

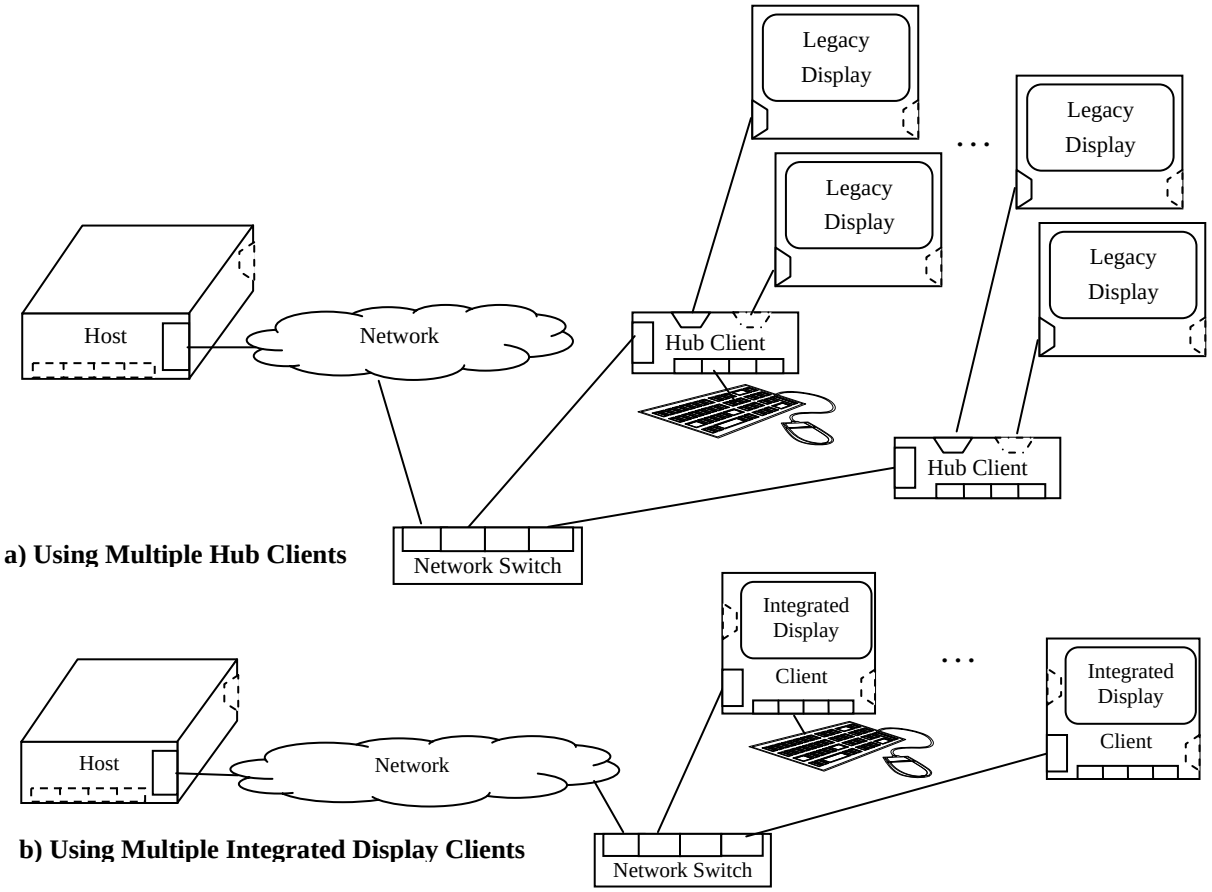


Figure 3-19: Multiple Displays using Multiple Clients Examples

3.3.8.4 Display Wall Examples

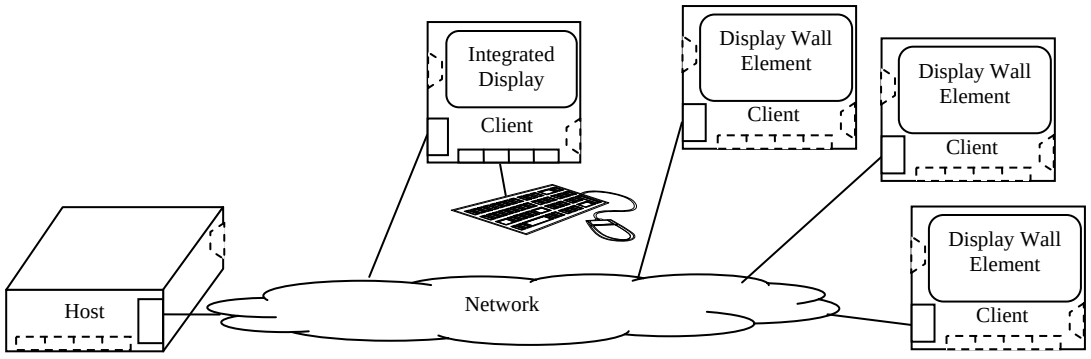
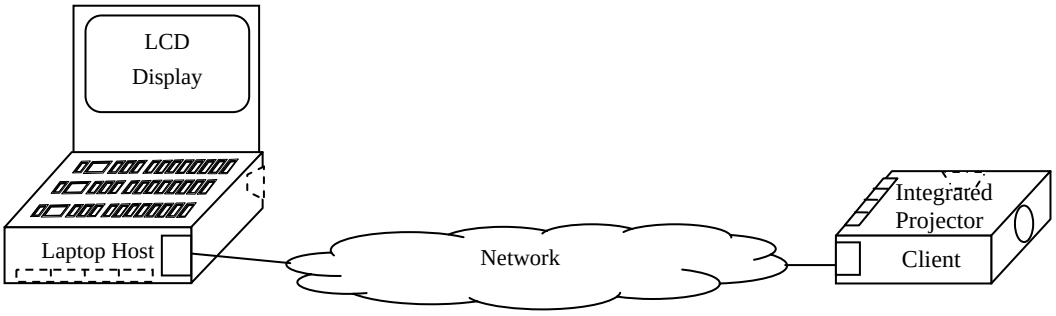


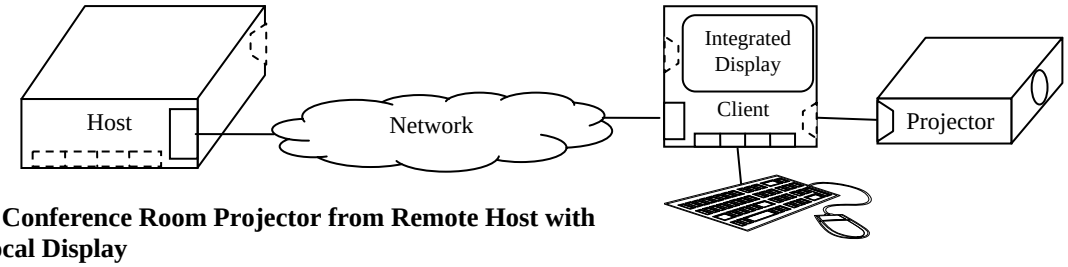
Figure 3-20: Distributed Display Wall of Multiple Clients Example

3.3.8.5 Conference Room Projector Examples

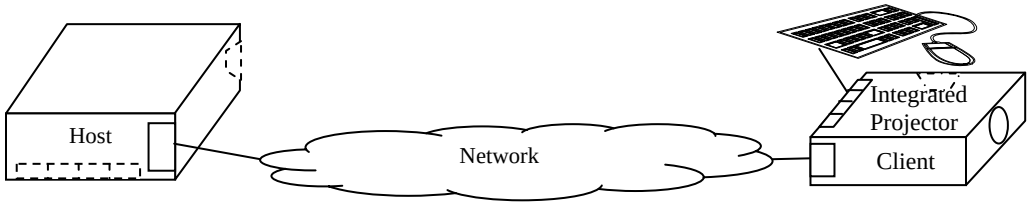
Different conference room configurations are shown in Figure 3-21. Integrated Projectors are shown in Figure 3-21 a) and c). An Integrated Projector is basically a projector that is directly connected to the network. An Integrated Projector is a special case of an Integrated Display, where the display is a projection device. An Integrated Projector may also include USB ports for the attachment of I/O devices.



a) Conference Room Projection from Laptop



b) Conference Room Projector from Remote Host with Local Display



c) Conference Room Projector from Remote Host

Figure 3-21: Conference Room Projector Examples

4 Networking

Net2Display Remoting operates over conventional IP networks without any special conditions or considerations. Typical conditions that can occur on IP networks include various network topologies, network latency, packet loss conditions and packet reordering. Net2Display works with the expected network topologies and network characteristics using the standardized IP protocols.

4.1 Network Topologies

Examples of different network topologies supported by Net2Display Remoting are enumerated in this section and are summarized in Table 4-1. Wired connections are shown with straight lines while wireless connections are shown with jagged lines. Cloud symbols indicate a network extent that may contain switches and routers. Optional Passive Clients are drawn in dashed lines in topologies where they may optionally be present. Both the Active Clients and the Passive Client may be Integrated Displays or Hubs. Active Clients have Primary Displays attached or integrated and may optionally have attached Secondary Displays. Passive Clients may provide for the attachment of one or more Secondary Displays.

4.1.1 Point-to-point Topologies

In a wired point-to-point configuration as shown in Figure 4-1, the Client is attached directly to the Host without any intervening devices. In this configuration, there are no DHCP or DNS services available and no Connection Management server is present. The Host and the Client detect that they are in a point-to-point configuration, locate each other and set up the Association with a minimum of manual intervention. In such a configuration, security requirements are low as the user has typically configured this to be like a directly attached display and there are no other devices attached or internet access. The ease of use of this case should be similar to a directly attached conventional display.

Figure 4-1: Wired Point-to-point

In the wireless point-to-point configuration shown in Figure 4-2, the Host and the Client would typically be connected using a wireless network such as 802.11 in infrastructureless mode. In such a configuration, there are no DHCP or DNS services available and no Connection Management Server is present. In the wireless point-to-point configuration, security concerns are somewhat greater since other wireless signals from other devices could be received and the Host could be spoofed by another device. Also, since the transmissions can be received by any neighboring receiver, encryption of traffic is imperative.

Figure 4-2: Wireless Point-to-point

4.1.2 Isolated Network Switches

Net2Display Remoting needs to work in configurations containing network switches, but isolated from infrastructure and the internet as illustrated in Figure 4-3. Typically, in this configuration, there are no DHCP or DNS services available and no Connection Management server is present. This configuration should be simple to form Associations as it is typical of configurations that would be used for the direct attachment of multiple displays. Security is more of a concern than the point-to-point case because other devices could be connected increasing the potential for spoofing or snooping.

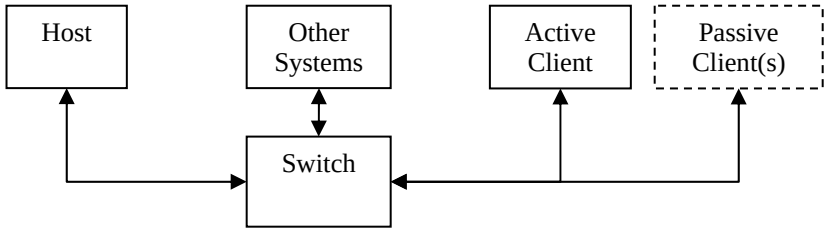


Figure 4-3: Isolated Network Switches

4.1.3 Within Infrastructure

Net2Display Remoting supports Hosts and Clients located within an infrastructure environment as illustrated in Figure 4-4. In such an environment, the Host or Clients could be either wire or wirelessly connected. DNS servers, DHCP servers, and Connection Management servers are likely to be present, but may be absent and are considered optional. Addresses may be either assigned statically or dynamically. Such an environment is typical of a corporate environment.

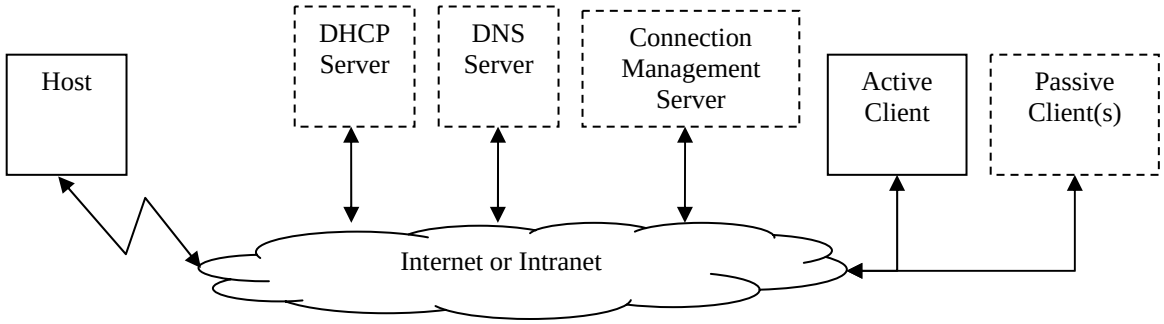


Figure 4-4: Connected to a Network Infrastructure

4.1.4 NAT Connected Clients

Net2Display Remoting supports remoting to Clients connected behind NAT devices as illustrated in Figure 4-5. DNS servers, DHCP servers, and Connection Management servers are likely to be present, but may be absent and are considered optional. This configuration is typical of home located displays connected to a local Host, an internet connected Host or corporate Firewalled connected Host.

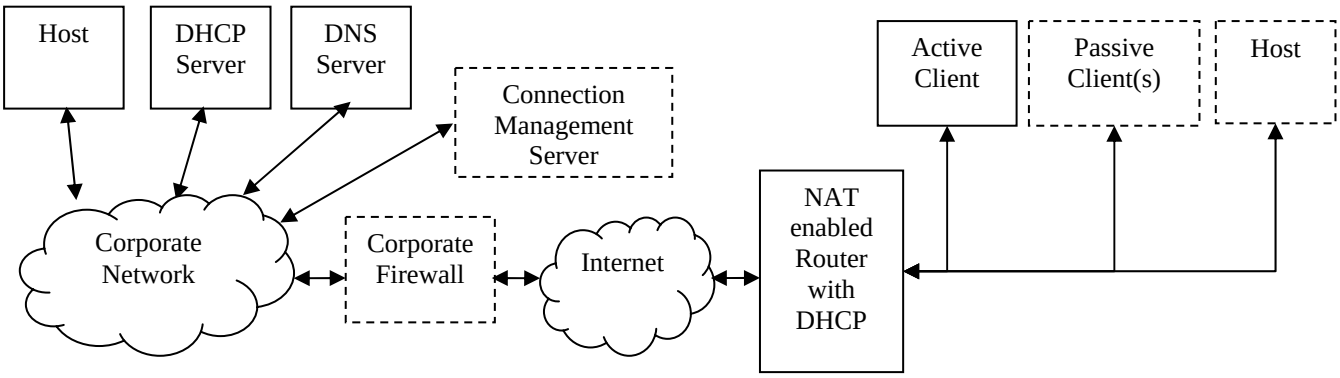


Figure 4-5: Client connected through a NAT to a local Host or to an Optionally Firewalled Host

4.1.5 NAT Connected Hosts – Not Supported

Net2Display does not directly support Hosts that are connected behind NAT devices as shown in Figure 4-6. Hosts behind a NAT-enabled router do not have true end-to-end connectivity and cannot participate in some

Internet protocols. NAT interferes with the initiation of TCP connections from the outside network and stateless protocols such as UDP. Unless the NAT router makes a specific effort to support such protocols, incoming packets cannot reach their destination.

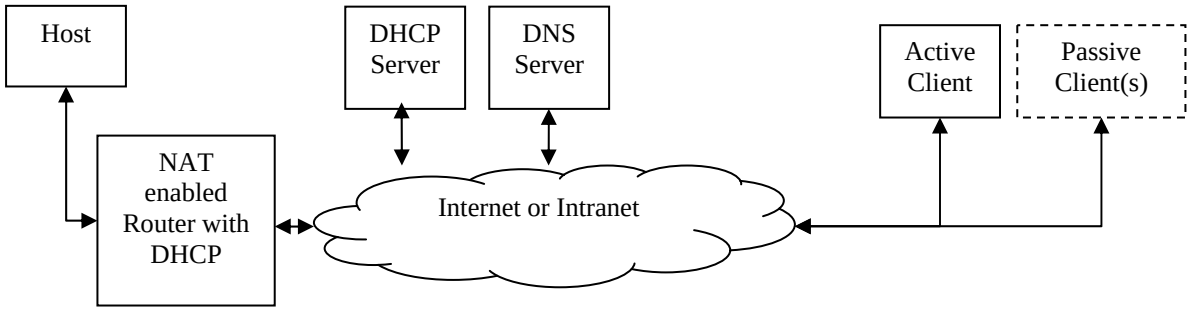


Figure 4-6: NAT Connected Hosts (Not Supported)

4.1.6 Network Firewalls

Currently, existing network firewalls may block or limit UDP traffic. When Net2Display Remoting is using IPsec to secure its data traffic and NAT traversal is used, all packets may appear to be UDP traffic. Thus, Net2Display Remoting is architected expecting that UDP and IPsec traffic can get through the network firewalls, or that administrators will allow Net2Display traffic through firewalls. This prevents the complication of negotiating whether TCP and TLS are used for all traffic or whether UDP and TCP are carried over IPsec.

4.1.7 Summary of Supported Network Topology Examples

A summary of the examples of the topologies supported by Net2Display Remoting is shown in Table 4-1.

Table 4-1: Examples of Supported Topologies

Name	DHCP Server	DNS Server	Connection Management Server	Across NAT
Wired Point-to-point	No	No	No	No
Wireless Point-to-point	No	No	No	No
Isolated Network Switch	No	No	No	No
Wired to a Network Infrastructure without DHCP Server or CMS	No	Yes	No	No
Wired to a Network Infrastructure without CMS	Yes	Yes	No	No
Wired to a Full Infrastructure	Yes	Yes	Yes	No
Client connected through NAT to a Full Infrastructure	Yes	Yes	Yes	Yes
Client connected through NAT to a Firewalled Host	Yes	Yes	Yes	Yes
Client and Host Wired through a NAT to a Full Infrastructure	Yes	Yes	No	No

4.2 Network Characteristics

4.2.1 Network Latency

IP network latencies can be hundreds of milliseconds for large distances or during times of network congestion. A Net2Display Node should be capable of continuing operation, typically with degraded performance, during periods of high network latency.

4.2.2 Variable Latency Conditions

Packets may be delayed by variable times due to network congestion. A Net2Display Node should be capable of operating in an IP network where the network latency is highly variable.

4.2.3 Packet Loss Conditions

Packets may be lost in an IP network due to media errors or congestion in routers or switches. Packet losses due to congestion can become large before congestion control resolves the condition. A Net2Display Node should be capable of continuing operation, typically with degraded performance, during periods of high packet loss.

4.2.4 Packet Reordering

IP networks make no assurances of in order packet delivery. Packets may be reordered due to load balancing across parallel routes or by switches or routers. A Net2Display Node should be capable of operating in the presence of out of order packet delivery.

4.3 Networking Facilities

Net2Display Remoting uses existing standardized network transport and security protocols to transport and secure the remoted Client display and I/O data. Net2Display should be capable of operating in existing networks of different extents. To accommodate the diversity of networks described in Section 4, Net2Display Remoting uses the following networking facilities:

- IP Networking – the Internet Protocol (IP) is used for carrying all data
- TCP and UDP Transport – TCP is required for all Net2Display Nodes and UDP is optionally used for motion video and audio
- DNS Servers – provides name resolution and service location using DNS Service Discovery
- mDNS – provides for name resolution and service location in networks without DNS servers
- IPsec – IP Security used for securing nodes and traffic across unsecured networks
- NAT Traversal (NAT-T) – Additional UDP encapsulation header that is added to allow IPsec traffic to traverse NAT gateways as typically found in most residences.
- Network Transport – Reliable and unreliable transport for traversing IP Networks with a range of network characteristics.
- IP Differentiated Services (DiffServ) – different Quality of Service levels used to ensure expedited delivery of real time data
- Maximum Transmission Unit discovery finds the maximum size that can be transmitted along the current network path without fragmentation.

Net2Display Remoting is architected to negotiate networking protocols, security protocols and compression between the Host and Client and to select the highest performance commonly supported. During the

Discovery Process, each end communicates all of the supported protocols and compressions and then selects the highest performance set to use for transmitting display content, motion video and data.

4.3.1 Internet Protocol (IP)

Net2Display Remoting requires the Internet Protocol for communication between a Host and a Client. All communication for Net2Display Remoting takes place through the IP protocol.

4.3.1.1 IP Version 6

A Net2Display implementation must be capable of operating with Intenet Protocol Version 6 (IPV6).

4.3.1.2 IP Differentiated Services

Net2Display supports the transport of a number of different types of data with different Quality of Service requirements. The different levels of Quality of Service are indicated in the Net2Display traffic through the IP Differentiated Service (DiffServ) Field. These fields are used according to their specification in IETF RFC 2474 and RFC 2475. The Per-Hop Behavior (PHB) is indicated by encoding a 6-bit value—called the Differentiated Services Code Point (DSCP)—into the 8-bit Differentiated Services (DS) field of the IP packet header.

The default behavior of Net2Display is that only best effort forwarding is used as indicated by a 0x00 value of the DSCP bits. During the startup of the Association between the Client and the Host, each end of the Association indicates the level of support for both sending and receiving Differentiated Services. Differentiated Services are only used if they are:

- Supported by the end sending the Differentiated Services
- Supported by the end receiving the Differentiated Services

4.3.1.2.1 Default PHB

A default PHB is the only required PHB. Essentially, any traffic that does not meet the requirements of any of the other defined classes is placed in the default PHB. Typically, the default PHB has best-effort forwarding characteristics. The recommended codepoint for the default PHB is '000000'.

4.3.1.2.2 Expedited Forwarding (EF)

The IETF defines Expedited Forwarding in RFC 3246. The EF PHB has the characteristics of low delay, low loss and low jitter. These characteristics are suitable for voice, motion video and other real-time services. EF traffic is often given strict priority queuing above all other traffic classes. Because an overload of EF traffic will cause queuing delays and affect the jitter and delay tolerances within the class, EF traffic is often strictly controlled through admission control, policing and other mechanisms. Typical networks will limit EF traffic to no more than 30%—and often much less—of the capacity of a link.

4.3.1.2.3 Assured Forwarding (AF)

The IETF defines the Assured Forwarding behavior group in RFC 2597. Assured forwarding allows the operator to provide assurance of delivery as long as the traffic does not exceed some subscribed rate. Traffic that exceeds the subscription rate faces a higher probability of being dropped if congestion occurs.

Usually, traffic policing is required to encode drop precedence. Typically, all traffic assigned to a class is initially given a low drop precedence. As the traffic rate exceeds subscription thresholds, the policer will increase the drop precedence of packets that exceed the threshold.

4.3.2 Network Transport

Typically, TCP is used to provide a reliable transport for the remoting of Net2Display traffic. However, for Motion Video and Audio traffic, the unreliable UDP protocol may be used, allowing lost packets to be skipped rather than introducing the additional latency that a reliable protocol such as TCP would incur when recovering from lost network packets. Net2Display communication typically takes place using TCP connections and UDP datagrams for transport. Other Network Transport means may be used optionally for transporting Net2Display data, provided both the Host and the Client support the transport.

4.3.2.1 Specified Transport Protocols

TCP is required for both the Association formation using SOAP and the reliable communication of control information. TCP is required as a transport protocol for Net2Display Remoting. UDP is optional to allow for a very simple implementation of Net2Display with all data flow across only one TCP connection. A summary of the different Transport Protocols that may be used by Net2Display are shown in Table 4-2.

Table 4-2: Transport Protocols used by Net2Display Remoting

Transport	Net2Display Usage	Rationale
TCP	Required	TCP is already required for the Association formation using SOAP.
UDP	Optional and Specified	UDP should be optional to allow for a very simple implementation of Net2Display
Reliable Transport over UDP	Optional	Optional as it defines an alterative service to TCP, which may have higher performance for hardware assisted implementations
SCTP	Optional, Vendor Defined	No real advantages over TCP, cannot mix traffic and have different DiffServ values due to the combining of chunks from different streams into packets
Other	Optional, Vendor Defined	Vendor Specific Transport could be used

4.3.2.2 Transmission Control Protocol (TCP)

TCP is used to transport several different types of traffic with reliable delivery requirements:

- Web Services using SOAP communication between the Client and the CMS as the Association is set up and managed. The SOAP is carried using the HTTP protocol which is transported using TCP.
- Web Services using SOAP communication between the Client and the Host communicating configurations and options. The SOAP is carried using the HTTP protocol which is transported using TCP.
- Reliable transport of data traffic between the Client and the Host for USB, keyboard data and persistent display data is transported using TCP.

One TCP connection must use the same DiffServ Values for all communications within the connection. Since all packets in a TCP connection are presented at the receiving end strictly in the order that they were presented to the transmitter, any differences in the DiffServ values within a single connection would cause misordering at the receiving end and would lessen the TCP performance. The DiffServ priority could be changed if there were no outstanding packets, but arranging for that would likely be more difficult than starting a new TCP connection with the different DiffServ Value. Likewise, each Virtual Channel that uses TCP uses the same DifServ values for all traffic within that Virtual Channel.

The Web Services SOAP traffic and other reliable traffic is transported using the Transmission Control Protocol (TCP) which is described in multiple specifications as summarized in IETF RFC 4614. The default

port is for the SOAP communication is TCP 80, but other Ports may be used. The specific processing and XML Schema of the Web Services content is defined in Section 22.

USB traffic and static display traffic is transported using TCP. The Net2Display TCP traffic is transferred using TCP Port Number 9086 for the Host Port by default. If Port 9086 is not used by for Host Node Port, then the Client and Host must provide other means for converging on the Host Port that will be used for Net2Display communication. The specific format of the transport of the reliable traffic using TCP is defined in Section 6.

4.3.2.2.1 TCP Keep-Alive

The TCP Keep-Alive function sends periodic traffic on TCP connections to ensure that the TCP connection is maintained and kept active. The TCP Keep-Alive function is specified in IETF RFC 1122 as an optional feature of TCP. The TCP Keep-Alive is used on every TCP connection that is used for Net2Display transport to ensure that connections do not time out and require reestablishment on the sending of new data after times of quiescence.

4.3.2.3 User Datagram Protocol (UDP)

The preferred method used for the transfer of motion video and audio is with datagrams using the User Datagram Protocol (UDP), which is specified in IETF RFC 768. Both Motion Video and audio data would suffer in latency or jitter if TCP were used and retransmissions occurred. UDP must keep the same values of IP Differentiated Services for all packets within a Virtual Channel. UDP is optional for Net2Display, since Motion Video and audio may also be supported using TCP. The Net2Display UDP traffic is transferred using UDP Port Number 9086 for the Host Port. If Port 9086 is not used for the Host Port, then the Client and Host must provide other means for converging on the Host Port that will be used for Net2Display communication. The specific format for the transport of Motion Video and audio using UDP is defined in Section 6.

To prevent problems of out of order delivery, UDP must keep the DiffServ values the same within a Virtual Channel. For multiple Virtual Channels that use UDP, they may each use different DiffServ values without affecting one another.

4.3.2.4 Stream Control Transmission Protocol (SCTP)

SCTP supports multiple streams within the same connection, allowing chunks from the different streams to be combined in the same packet. This combining of chunks from different streams is done by SCTP and is not under the control of the application, making the setting priorities on packets with mixed content problematic. Since SCTP can combine chunks from different reliable streams and unreliable datagrams within one SCTP packet, each SCTP connection must use just one DiffServ value to prevent the reordering that different DiffServ values would produce. This reordering of packets within the SCTP connection could cause reordering within a lower priority data stream leading to performance degradation.

4.3.2.5 Reliable Transport over UDP

If the reliable transport provides reliability within individual Virtual Channels, then the DiffServ values can be set differently for each Virtual Channel.

Note: Since TCP requires significant resources for high performance, it would be desirable to have a lower overhead reliable datagram protocol. While Net2Display Remoting would benefit from an alternative reliable datagram protocol, the Net2Display task group decided not to include a reliable transport protocol in the body of the first version of the standard. Such a reliable datagram protocol could be added later and would be indicated by utilizing one of the reserved bits in the Net2Display Header.

4.3.2.6 Summary

A summary of the relationships of priorities and the different types of transport are shown in Table 4-3.

Table 4-3: Summary of Priorities for Different Communication Channels

Communication Path	DiffServ
Virtual Channel	One DiffServ value per Virtual Channel
TCP	One DiffServ value per TCP Connection
UDP	One DiffServ value per UDP Virtual Channel
Reliable Transport over UDP	One DiffServ value per Virtual Channel
SCTP	One DiffServ value per SCTP Connection

Table 4-4: DiffServ Priorities used by Different Transport Protocols

Transport Protocol	Support in Net2Display	Examples of Highest Priority Traffic (Expedited Forwarding)	Examples of High Priority Traffic	Examples of Medium Priority Traffic	Examples of Low Priority Traffic
TCP	Required	Pointer, Keyboard, USB Interrupt	Net Display, USB Primary/Control		USB Bulk
UDP	Optional, Specified		Audio Out, Audio In, USB Isochronous[1]	Motion Video[1]	
Reliable Transport over UDP	Optional, Specified	Optional: Pointer, Keyboard, USB Interrupt	Optional: Net Display, USB Primary/ Control		Optional: USB Bulk
SCTP	Optional, Vendor Defined	Optional: Pointer, Keyboard, USB Interrupt	Optional: Net Display, USB Primary/Control, Audio out, Audio In, USB Isochronous	Motion Video	Optional: USB Bulk
Other	Optional, Vendor Defined				

4.3.3 Security

Security is required for all Net2Display traffic. Net2Display traffic is typically provided using the IP Security (IPsec) protocol as specified in the following section. Other security approaches may be optionally used. If another security protocol such as TLS or HDCP is used, it must be supported by both ends of the Net2Display Association and used in a manner that is outside the information in this standard.

Note: TCP or UDP may be used to transport HD Video already protected end-to-end by High-bandwidth Digital Content Protection (HDCP). HDCP is a form of digital copy protection developed by the Intel Corporation to prevent copying of digital audio and motion video content as it travels across a video connection. The specification is proprietary, and implementing HDCP requires a license.

4.3.3.1 IPsec Security

Net2Display Remoting traffic uses the IP Security (IPsec) protocol as specified in IETF RFC 4301, also known as IPsec ESPbis. Within IPsec, the Encapsulating Security Payload (ESP) for providing integrity, data origin authentication, anti-replay features and confidentiality. The ESP is employed as specified in IETF RFC 4303. The ESP anti-replay feature is to be enabled, which is the default setting. The headers included in a

[1] If UDP or other unreliable transport is not available for use, then Reliable Transport must be used with the same priority level.

TCP packet secured with IPsec are shown in Figure 4-7. The headers included in a UDP packet secured with IPsec using NAT Traversal are shown in Figure 4-8.

IP Header	Version	IHL	Type of Service	Total Length	
	Identification				
	Time to Live		Protocol	Header Checksum	
	Source Address				
	Destination Address				
	Options				Padding
IPsec ESP Header	Security Parameters Index (SPI)				
	Sequence Number				
TCP Header	Source Port			Destination Port	
	Sequence Number				
	Acknowledgment Number				
	Data Offset		Flags	Window	
	Checksum			Urgent Pointer	
	Options				Padding
Data					

Figure 4-7: TCP Packet Structure with IPsec

IP Header	Version	IHL	Type of Service	Total Length	
	Identification				
	Time to Live		Protocol	Header Checksum	
	Source Address				
	Destination Address				
	Options				Padding
IPsec ESP Header	Security Parameters Index (SPI)				
	Sequence Number				
UDP Header	Source Port			Destination Port	
	Length			Checksum	
Data					

Figure 4-8: UDP Packet Structure with IPsec

Note: Since all Net2Display traffic is secured using IPsec, then the specific transport protocol that is used is not visible to the network. The Net2Display task group made a decision that security is required in Net2Display Remoting. The only approach to securing UDP traffic is with IPsec, since it is below the transport level and can secure unreliable traffic. When IPsec is used, the UDP and TCP packet headers are contained in the encrypted portion of the IPsec payload and are not visible for processing by the network or network interfaces. Thus, the network cannot differentiate TCP or UDP traffic; they are all just IPsec traffic.

4.3.4 NAT Traversal

Net2Display Remoting utilizes NAT Traversal (NAT-T), which defines UDP encapsulation of the ESP packets to assist in traversing firewalls. The negotiation of this capability is performed during the Internet key exchange (IKE) phase as defined in IETF RFC 3947 using UDP encapsulation as defined in IETF RFC 3948. Other requirements to enable the IPsec compatibility with NAT are specified in IETF RFC 3715. The headers

included in a TCP packet secured with IPsec using NAT Traversal are shown in Figure 4-9.The headers included in a UDP packet secured with IPsec using NAT Traversal are shown in Figure 4-10.

Note: When NAT Traversal is used, a UDP header is added in front of the IPsec header to provide the Port Number for processing. With the UDP header present, all packets appear to the network as UDP packets, since a UDP Header is used to encapsulate the IPsec Header for traversing NAT.

<table>
<tr><td rowspan="6">IP Header</td><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td colspan="2"></td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
<tr><td rowspan="2">UDP Header</td><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="3">Length</td><td colspan="2">Checksum</td></tr>
<tr><td>IPsec ESP Header</td><td colspan="5">Security Parameters Index (SPI)</td></tr>
<tr><td rowspan="7">TCP Header</td><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="5">Acknowledgment Number</td></tr>
<tr><td colspan="2">Data Offset</td><td>Flags</td><td colspan="2">Window</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
<tr><td>Data</td><td colspan="5"></td></tr>
</table>

Figure 4-9: TCP Packet Structure with IPsec and NAT-T

<table>
<tr><td rowspan="5">IP Header</td><td>Version</td><td>IHL</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td colspan="2"></td></tr>
<tr><td colspan="2">Time to Live</td><td>Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source Address</td></tr>
<tr><td colspan="5">Destination Address</td></tr>
<tr><td colspan="4">Options</td><td>Padding</td></tr>
<tr><td rowspan="2">UDP Header</td><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="3">Length</td><td colspan="2">Checksum</td></tr>
<tr><td>IPsec ESP Header</td><td colspan="5">Security Parameters Index (SPI)</td></tr>
<tr><td rowspan="3">UDP Header</td><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="3">Length</td><td colspan="2">Checksum</td></tr>
<tr><td>Data</td><td colspan="5"></td></tr>
</table>

Figure 4-10: UDP Packet Structure with IPsec and NAT-T

4.3.5 DNS Services

Net2Display makes use of DNS Servers for both name resolution and locating Net2Display services. These services may include the location of Net2Display CMS Servers, Hosts and Clients.

Note: DNS Servers are considered to be not accessible when multiple DNS queries with sufficient timeouts have failed. It is recommended to set the DNS query timeouts to values of: 1 2 2 4 8 0 for a total cumulative timeout of 17 seconds.

4.3.5.1 Multicast DNS (mDNS)

When a DNS Server is not accessible, mDNS will be used for name resolution. mDNS allows a network device to choose a domain name in the ".local" namespace and announce it using a special multicast IP address. This introduces special semantics for the .local namespace (which is considered a problem by some members of the IETF. http://www1.ietf.org/mail-archive/web/ietf/current/msg37126.html.) mDNS is compatible with DNS-SD. mDNS is that name resolution provided by ZeroConf. Multicast DNS suffers from a number of scalability issues (imagine asking for a printer in a building with 200 printers), but mDNS should only be used in limited size environments where DNS services are not available.

4.3.5.2 DNS Service Discovery (DNS-SD)

DNS Service Discovery (DNS-SD) is a way of using standard DNS programming interfaces, servers, and packet formats to browse the network for services. DNS Service Discovery is compatible with, but not dependent on, Multicast DNS. It uses DNS SRV (RFC 2782), TXT, and PTR records to advertise Service Instance Names. The hosts offering the different services publish details of available services like instance, service type, domain name and optional configuration parameters. The Net2Display Remoting service is referenced using the service type “net2display.” (See http://www.dns-sd.org/ServiceTypes.html)

4.3.5.2.1 DNS SRV Records

Currently, one must either know the exact address of a server to contact it, or broadcast a question. The SRV RR allows administrators to use several servers for a single domain, to move services from host to host with little effort, and to designate some hosts as primary servers for a service and others as backups.

Clients ask for a specific service/protocol for a specific domain (the word domain is used here in the strict RFC 1034 standard sense), and get back the names of any available servers. In general, it is expected that SRV records will be used by clients for applications where the relevant protocol specification indicates that clients should use the SRV record. Such specification must define the symbolic name to be used in the Service field of the SRV record as described below. It also must include security considerations. Service SRV records must not be used in the absence of such specification.

For Example, if a SRV-cognizant LDAP client wants to discover a LDAP server that supports TCP protocol and provides LDAP service for the domain example.com., it does a lookup of

_ldap._tcp.example.com

as described in [ARM]. The example zone file near the end of this memo contains answering RRs for an SRV query. Note that LDAP is chosen as an example for illustrative purposes only, and the LDAP examples used should not be considered a definitive statement on the recommended way for LDAP to use SRV records. As described in the earlier applicability section, consult the appropriate LDAP documents for the recommended procedures.

4.3.6 Maximum Transmission Unit Discovery

Maximum Transmission Unit (MTU) may optionally be discovered using the Packetization Layer Path MTU Discovery (PLPMTUD) algorithm specified in RFC 4821. The PLPMTUD method is applicable to TCP and other transport- or application-level protocols that are responsible for choosing packet boundaries (e.g., segment sizes) and have an acknowledgment structure that delivers to the sender accurate and timely indications of which packets were lost.

Most of the difficulty in implementing PLPMTUD arises because it needs to be implemented in several different places within a single node. In general, each Packetization Protocol needs to have its own implementation of PLPMTUD. The simplest mechanism to share Path MTU information between concurrent or subsequent connections is a path information cache in the IP layer. The various Packetization Protocols need to have the means to access and update the shared cache in the IP layer. RFC 4821 describes

PLPMTUD in terms of its primary subsystems without fully describing how they are assembled into a complete implementation.

Note: The approach of RFC 1191 Path MTU Discovery cannot always be used because the ICMP packets it uses are often blocked in the network to prevent denial of services attacks. The PLPMTUD approach works around this limitation.

4.4 Networking Summary

The TCP/IP networking protocol stacks exist at both the Net2Display Host and Client. An example of the networking protocols that are present in a Net2Display Host are shown in Figure 4-11. At the Host End, Net2Display gathers the display and Motion Video data from various display interfaces within the Host End. Net2Display provides the audio, pointer and keyboard input to the associated host drivers and presents the remoted USB data to the Host USB controller.

<table>
<tr><td rowspan="4">Net2Display Association Setup and Management</td><td colspan="13">Application Interface</td></tr>
<tr><td colspan="2">Display Interfaces</td><td>Audio Driver</td><td>Mouse Driver</td><td>Keyboard Driver</td><td colspan="7">USB Controller</td><td>Vendor Specific Device Interface</td></tr>
<tr><td>Net Display Channels</td><td>Motion Video Channels</td><td>Audio Channel</td><td>Pointer Channel</td><td>Keybrd Channel</td><td rowspan="2">USB Primary/ Control</td><td>USB Keybrd</td><td>USB Pointer</td><td>USB Speakers</td><td>USB Micro-phone</td><td>USB Webcam</td><td>USB Printer</td><td>USB Flash Key</td><td>USB Disk Drive</td><td rowspan="2">Extended Virtual Channels</td></tr>
<tr><td colspan="2">USB Interrupt</td><td colspan="3">USB Isochronous</td><td colspan="3">USB Bulk</td></tr>
<tr><td colspan="14">TCP/UDP</td></tr>
<tr><td colspan="14">IPsec</td></tr>
<tr><td colspan="14">IP</td></tr>
</table>

Figure 4-11: Example of the Networking Protocol Stack at Host End

An example of the networking protocols that are present in a Net2Display Client are shown in Figure 4-12. At the Client end, the display and Motion Video data are passed onto the display controller and the audio is passed to the audio codec. Pointer and Keyboard input may be interpreted locally, but is also passed on across Net2Display Remoting to the Host. USB data is collected and separated into types and passed across Net2Display Remoting.

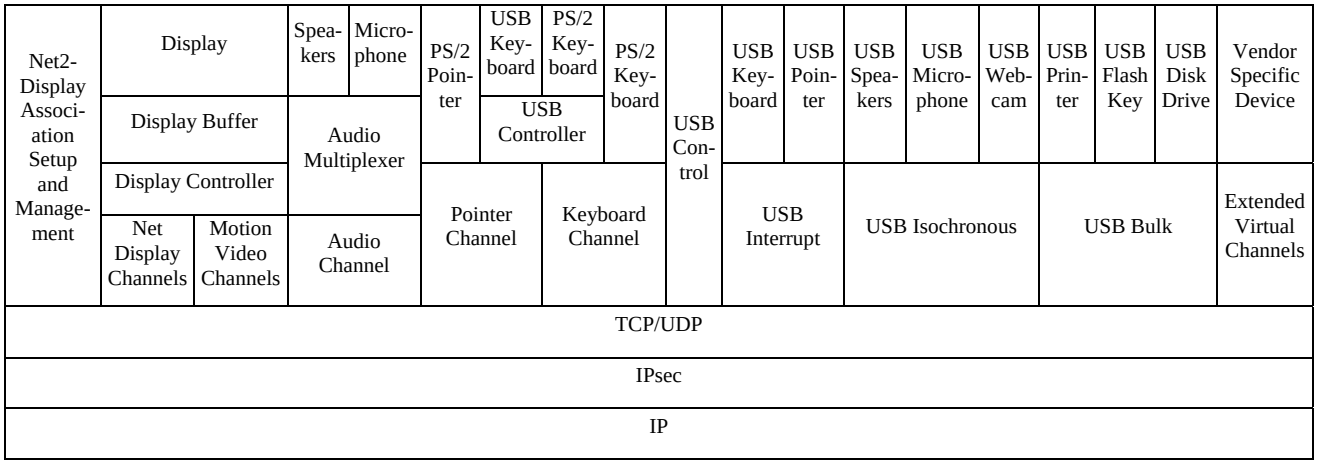


Figure 4-12: Example of the Networking Protocol Stack at Client End

5 Security

Net2Display Remoting requires authentication and confidentiality in all Net2Display Associations between Hosts and Clients. This avoids the complexities of making security optional and lessens security considerations of the user. The security and encryption capabilities of an individual implementation are identified in the Net2Display Configuration and Identification data. The Net2Display Remoting security is intended to provide security suitable for business office use, financial services and medical services. Net2Display Remoting implementations may optionally provide user defined security services for use by the military or other government agencies.

For Net2Display remoting, TLS will be used for Certificate exchange and IKEv2 will be used for IPsec Key Exchange.

5.1 Certificate Management

Public Key Infrastructure (PKI) X.509 is used as specified in IETF RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.

5.1.1 Recommended Certificate Authorities and Certificates

Net2Display makes used of certificates from recognized authorities for security. For general guidance, a list of recommended certificate authorities and certificates to be contained in a system that will operate as a Net2Display Client are included in Appendix I.

5.2 Key Distribution and Management

Key distribution and management must be performed according to the Internet Key Exchange Version 2 (IKEv2), which is specified in IETF RFC 4306. The automated key distribution and management specified in IKEv2 is to be used.

5.3 Trusted Platform Module (TPM)

A TPM chip, a microcontroller that can store secured information, is an option in Net2Display Remoting. The TPM capability is communicated in the Configuration information. When a TPM is present it can be used for storing keys and certificates.

5.4 Authentication

Net2Display Remoting provides support for Device Authentication, it must be possible to authenticate that an endpoint (client, host, admin server, etc) is the device it claims to be. Net2Display Remoting also provides support for user authentication, verifying a user's identity. While user authentication is not directly part of Net2Display Remoting, secure connections of user authentication devices such as biometric or smartcard devices must be supported.

While initial user authentication is outside the scope of this document, Net2Display remoting retains the user authentication as additional secure transport sessions are setup. Once a secure TLS session is initiated, Active Directory Tickets may be carried by the TLS session to authenticate a user’s access to a Host. Once the user is authenticated in a N2D TLS session, an Association_Cookie is generated and exchanged. The Association_Cookie is exchanged as described in IKEv2 to authenticate a media session set up to handle PDUs. This structure is described in Section 8.

Net2Display provides an infrastructure for managing authorization, so that, a method is supported to determine if an authenticated device is allowed to connect and to prevent the connection if appropriate. Net2Display allows an administrator to remotely manage the security configuration of a Net2Display Remoting device. It must be possible to lock the association between a specific Host and a specific Client, either by specifying at the Client, Host or Target Locator Address.

5.4.1 Device Authentication

Clients and hosts are authenticated as part of the IPsec, TLS, or HDCP key negotiation, and no further authentication is required.

5.4.2 User Authentication

User authentication is an application level function. The Net2Display Client may forward attached TPM, keyboard, fingerprint, or smartcard device data to the authentication application in support of the application based user level authentication.

5.5 Client Configuration

Net2Display Clients must have a capability defined for locking down Clients in a standards defined manner so that they can be managed by an IT organization and the user cannot change the configuration. There must, however be a way for the physical possessor to reset all secrets and configurations back to factory default. Any such reset will zero all secrets, including passwords and private keys, so the device will no longer be able to authenticate as properly configured.

- A Net2Display system will initially be configured as trusting and then after configuration security will be strengthened.
- A way for a Management Console to access the basic Client configuration information must be defined for Net2Display Remoting.

5.6 Content Protection

Implementations may support HD content protected by HDCP. This content will be transported over UDP, and will not require additional protection by IPsec or TLS.

5.7 Trusted Display

Net2Display Clients must have a trusted way of displaying information about the existing connection(s), including the name, address, and security characteristics. This connection information display must be trusted, in that the remote Host(s) cannot block, alter, or forge the information, so as to deceive the viewer. There are many ways to achieve trusted display of the connection information, including:

- The use of a secure attention key, such as the “info” button on an infrared remote control, which causes the client to overwrite the information on top of the Host provided content,
- The use of a separate “channel” display, containing the required information,
- The use of a reserved section of the display, such as in a window manager panel at the bottom of the screen,
- The use of informative window “chrome” in the case where the client provides windowed access to multiple concurrent remote host windows.

The displayed name must not be settable by the remote Host; it must be based on a trusted name source, such as the name from the authenticated TLS/IPsec certificate, inverse DNS lookup, or configuration information in the client. The security characteristics must include whether or not the connection has been authenticated, and may include the authentication method (TLS/IPsec/HDCP).

6 Net2Display Element Requirements

6.1 Net2Display Host Node Requirements

A Host Node configuration provides remoting to Net2Display Clients. A Net2Display Host Node must have the following components to be a conforming Net2Display instantiation:

1. An interface to a Host Server
2. An IP network interface for Net2Display Remoting traffic
3. TCP networking stack
4. IPsec security protocol support
5. Net2Display Host Node functions

The maker or provider of a Host Node configuration must state what Client Configurations its supports. It is recommended that the maker or provider of a Net2Display Host state what optional Net2Display features are supported by the Host configuration.

6.1.1 Secure Host Configuration

The Secure Host Configuration is a Host Node configuration that provides an environment where the transfer of data via removable media is prevented. A Secure Host Configuration must provide the following Virtual Channels: Display, Motion Video, Keyboard, Pointer, and Audio Out. A Secure Host Configuration must not allow the formation of the following Virtual Channels: USB and Audio In.

6.1.2 Host Configuration Summary

The different sets of capabilities of Host configurations are summarized as shown in Table 6-1. The features that a Host Node of a particular configuration type must have are stated as Required, those features that a Host Node may have are stated as Optional and those that a Host Node must not have are stated as Prohibited.

Table 6-1: Host Configuration Requirements

Key: **Rqd. – Required (Must have)** **Opt. – Optional (May have)** **None – Prohibited (Must not have)**	**Full Host**	**Secure Host**	**Net2Display Document Section**
Display or Display Connector and Net Display Virtual Channel	Rqd.	Rqd.	11
Motion Video Virtual Channel	Rqd.	Opt.	12
Motion Video Control Virtual Channel	Opt.	Opt.	D
USB Ports and USB Primary Virtual Channel	Rqd.	None	13
Additional USB Channels	Opt	None	13
Keyboard Support (Keyboard via PS/2 or USB Port)	Rqd.	Rqd.	14
Keyboard Virtual Channel Support	Rqd.	Rqd.	14
Pointer Support (Integrated, PS/2 or USB)	Rqd.	Rqd.	15
Pointer Virtual Channel Support	Rqd.	Rqd.	15
Local Cursor Support	Rqd.	Rqd.	15
Audio Out (Speakers or Jacks) and Virtual Channel	Rqd.	Opt.	16
Audio In (Microphone or Jack) and Virtual Channel	Rqd.	None	16
Parallel Port Virtual Channel	Opt.	None	F
Extended Virtual Channels	Opt.	None	17

6.2 Net2Display Client Node Requirements

A Net2Display Client Node must have the following components to be a conforming Net2Display instantiation:

1. An IP network interface for Net2Display Remoting traffic
2. TCP networking stack
3. IPsec security protocol support
4. One or more displays, display interfaces, or display connectors
5. Net2Display Client Node functions.

The Minimal Client Configuration is a special case of the Display Client Configuration. The absolute minimal configuration of a Net2Display Client must have display with at least one single pixel.

6.2.1 Client Configurations

Other Client Configurations include:

- **Workstation Client Configuration** - A Workstation Client is a Client Node that supports both Displays and USB I/O.

- **Display Client Configuration** - A Display Client is a Client Node that does not have any attached I/O and serves as a stand-alone display or serves to provide as an additional display to a Terminal Client. A Display Client may be connected to remoting from a Display Host, a Terminal Host or a Secure Host. A Display Client does not typically provide USB functionality.
- **Terminal Client Configuration** - A Terminal Client is a Client Node with attached I/O devices that typically allow user interaction. A Terminal Client provides USB and Audio functionality. The keyboard and pointer information can either be carried over separate Keyboard and Pointer Virtual Channels or over USB.
- **Secure Client Configuration** - The Secure Client Configuration is a Terminal Configuration designed to provide an environment where the transfer of data via removable media is prevented.

The different sets of capabilities of Client Configurations are summarized in Table 6-2. The features that a Client Node of a particular configuration type must have are stated as Required, those features that a Client Node may have are stated as Optional and those that a Client Node must not have are stated as Prohibited.

Table 6-2: Client Configuration Requirements

Key: **Rqd. – Required (Must have)** **Opt. – Optional (May have)** **None – Prohibited (Must not have)**	**Workstation Client**	**Terminal Client**	**Secure Client**	**Display Client**	**Net2Display Document**
Display or Display Connector and Net Display Virtual Channel	Rqd.	Rqd.	Rqd.	Rqd.	11
Motion Video Virtual Channel	Rqd.	Opt.	Opt.	Opt.	12
Motion Video Control Virtual Channel	Opt.	Opt.	Opt.	Opt.	D
USB Ports and USB Primary Virtual Channel	Rqd.	Opt	None	Opt.	13
Additional USB Channels	Opt	Opt	None	Opt.	13
Keyboard Support (Keyboard via PS/2 or USB Port)	Rqd.	Rqd.	Rqd.	Opt.	14
Keyboard Virtual Channel Support	Rqd.	Opt.	Rqd.	Opt.	14
Pointer Support (Integrated, PS/2 or USB)	Rqd.	Rqd.	Rqd.	Opt.	15
Pointer Virtual Channel Support	Rqd.	Opt.	Rqd.	Opt.	15
Local Cursor Support	Rqd.	Opt.	Opt.	Opt.	15
Audio Out (Speakers or Jacks) and Virtual Channel	Rqd.	Opt.	Opt.	Opt.	16
Audio In (Microphone or Jack) and Virtual Channel	Rqd.	Opt.	None	Opt.	16
Parallel Port Virtual Channel	Opt.	Opt.	None	Opt.	F
Extended Virtual Channels	Opt.	Opt.	None	Opt.	17

6.3 Other Host and Client Node Options

Net2Display allows a wide range of performance and configuration options for Hosts and Clients. These different options provide different performance, security and configuration features with different levels of client complexity. The support of these options in a Net2Display Host or Client is communicated during Association formation.. Some of the configuration options within Net2Display include:

- UDP Network Transport
- USB I/O
- Remoted USB Keyboard and Pointer, Client Controlled Keyboard and Mouse, or None
- Display Commands Supported: RawPixel, Copy, SolidFill, PatFill, BiFill
- Support for Motion Video Streams
- Compression Types Supported
- Client Aggregation or no Aggregation
- Client and User Authentication
- Single or Multiple Displays
- Client Local Mouse Support

6.4 Association Requirements

6.4.1 IP Addresses

An Association is formed between a Host Node and a Client Node. An Association is identified by its Host/Client pair of IP addresses and Security Identifier within a network. Thus, each Association must have a Host and Client pair of addresses and Security Identifier that is unique in the network or uniquely reachable through NAT. For locally assigned addresses, the Host and Client addresses and Security Identifier must only be unique on the local network. Thus, the pair of Host and Client IP addresses identifies the endpoints of the Association.

6.4.2 Host Port Number

The initial communication between the Client and the Host is performed using a default of Port Number 80 on the Host for the HTTP communication. If Port 80 is not used by the Host, then the Client and Host must provide other means for converging on the Host port that will be used.

After the Net2Display Association is formed, the Net2Display remoting default is to communicate using TCP and UDP ports 9086 for the Host Port end of the Association. If Port Number 9086 is not used by the Host Port, then the Client and Host must provide other means for converging on the Host Port that will be used for Net2Display communication. Thus, the Host Port Number is used for identifying the TCP and UDP communication as Net2Display traffic.

6.4.3 Client Port Numbers

The Client end may use any Port number that it chooses for communication that is not reserved or used for other purposes. For TCP, the Client Port is typically incremental, usually beginning at 1024 at boot time and wrapping at 4096. With UDP, Client Port selection depends on the application and may be incremental, fixed to a nonsensical value, or fixed equal to the Host Port. Thus, the Client Port is used to identify the Net2Display communications traffic at the Client end.

If multiple levels of IP Differentiated Services are used to provide multiple Virtual Channel Priorities to transport different types of Net2Display remoting traffic, then a different Client Port Number must be used

with each different IP Differentiated Service level that is used. Thus, multiple Client Port Numbers must be used to support multiple Net2Display priority levels.

6.5 Client Aggregation Requirements

Multiple Clients are Aggregated when they all Associate with the same Host Node and share the same display coordinate space, cursors and I/O devices. When multiple Clients are Associated to the same Host, they form separate Associations to the Host. These Associations may be grouped together to form an Aggregated Client.

The Net2Display Aggregation function is to allow displays to be assigned to a single global address space, the Aggregated Coordinate Space. Using the Aggregated Coordinate Space allows a Client's pointer control to identify which Net Display the pointer is on. Local Pointer coordination for multiple Aggregated Clients is a responsibility of the Client vendor and should be presented as a single control from one client to the Net2Display host.

Aggregated Clients are identifiable as Aggregated because they all communicate with the same Host using the same Host IP address and same Host Port Number. Thus, all Clients in an Aggregated Client configuration must communicate with the Host using the same Host IP address and the same Host Port Number.

7 Protocol Data Units

The Net2Display Protocol Data Unit (PDU) is the basic block of transfer for Net2Display across a network. All Net2Display traffic is communicated using Net2Display PDUs. Net2Display PDUs contains two major parts, a Net2Display PDU Header and Optional Net2Display PDU Command Data as shown in Figure 7-1.

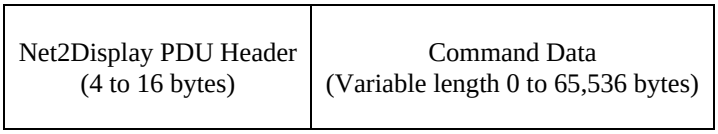


Figure 7-1: Net2Display PDU Structure

7.1 PDU Types

There are three major types of Net2Display PDUs: Control PDUs, Net2Display Defined Virtual Channels and Extended Virtual Channels. A Virtual Channel is a dedicated pipe that is established between a Host and a Client and is used to exchange a specific type of data, such as display, keyboard, pointer, or audio. The three major types of Net2Display PDUs are shown in Table 7-1 along with the settings of the Header bits.

Table 7-1: Net2Display PDU Types

PDU Type	Control Bit (C bit)	Extended Bit (E bit)	Use
Net2Display Defined Virtual Channel PDUs	0	0	Net2Display Defined Virtual Channel data traffic
Extended Virtual Channel PDUs	0	1	Extend Virtual Channel data traffic
Control PDUs	1	0	Used for managing Association and controlling Net2Display Defined Virtual Channels
Control PDUs	1	1	Used for controlling Extended Virtual Channels

7.1.1 Control PDUs

Control PDUs are used to initialize the Association and to setup and take down Virtual Channels. Both Net2Display Defined Virtual Channels and Extended Virtual Channels use Control PDUs for their setup and termination. Control PDUs are distinguished by having the Control Bit set in the PDU Header. Both Net2Display Defined and Extended Virtual Channels must conform to the same overall protocol for the setup and closing of Virtual Channels using Control PDUs as specified in Section 9.3.

7.1.2 Net2Display Defined Virtual Channels

Net2Display Defined Virtual Channels are the set of Virtual Channels that are defined and specified within the Net2Display standard. This includes the full set of data typically communicated between a Host and Client including display, Motion Video, pointer, keyboard, USB and audio in and out. Net2Display Defined Virtual Channel PDUs are distinguished by both the Control Bit and the Extended Bit not being set. Net2Display Defined Virtual Channels must always have the full 16 bytes of PDU Header as defined in Section 6.

7.1.3 Extended Virtual Channels

Extended Virtual Channels are used to encapsulate application or vendor specific data that is defined outside of the Net2Display standard. Extended Virtual Channels are negotiated between the Host and Client. An Extended Virtual Channel is distinguished by the setting of the Extended Bit and the Control Bit not being set.

7.2 PDU Header

The Net2Display PDU Header is organized into Header Field Groups ordered according to the typical processing order. Fields placed earlier in the Protocol Data Unit (PDU) header are used to identify how to process the later fields of the PDU. The Header Field Groups in the order that they appear in the PDU Header are:

1) **Net2Display Version Header Field Group** identifies how to interpret the Net2Display header and which version of the Net2Display protocol to use. This Header Field Group occurs first because it determines how all the other fields of the PDU header are to be interpreted.

2) **Virtual Channel Control Header Field Group** contains information used to setup and manage the Virtual Channels. This is used to indicate when control information is being sent for forming and terminating Virtual Channels. This is early in the PDU as it differentiates whether the PDU contains control or Virtual Channel data.

3) **Virtual Channel Identification Header Field Group** contains information used to identify an individual Virtual Channel.

4) **Virtual Channel Specific Control Header Field Group** contains information for managing the flow of data within a Virtual Channel. Once a channel is identified, the Virtual Channel Specific Control Header Field Group is used to pass information for adjusting the delivery of the specific Virtual Channel data. This Header Field Group is last because it is only relevant once the particular Virtual Channel has been identified.

Placing these Header Field Groups in the order in which they would be processed or accessed yields the following high level PDU Header Structure shown in Figure 7-2.

<table>
<tr><td>Word</td><td colspan="5">Byte 0</td><td colspan="2">Byte 1</td><td>Byte 2</td><td>Byte 3</td></tr>
<tr><td rowspan="2">0</td><td>Version Header Field Group</td><td colspan="4">Virtual Channel Control Group</td><td colspan="4">Virtual Channel Identification Header Field Group</td></tr>
<tr><td>Version Number</td><td>CBit</td><td>EBit</td><td colspan="2">Reserved Control Bits (RC Bits)</td><td colspan="4">Virtual Channel ID</td></tr>
<tr><td rowspan="2">1</td><td colspan="9">Virtual Channel Specific Control Header Field Group</td></tr>
<tr><td colspan="2">Virtual Channel Protocol Type</td><td>Reserved Header Bits (RH Bits)</td><td>Contin-uation / More Bits (CM Bits)</td><td>RsBit</td><td>Command Code</td><td colspan="3">PDU Length</td></tr>
<tr><td>2</td><td colspan="9">VC_Timestamp (Optional)</td></tr>
<tr><td>3</td><td colspan="7">Sequence Number</td><td colspan="2">Received Sequence Number</td></tr>
<tr><td>4 …</td><td colspan="9">CommandData (Optional)</td></tr>
</table>

Figure 7-2: Net2Display PDU Fields

7.2.1 Version Header Field Group

7.2.1.1 Version Number – 3 Bits

The Version bits allow the Net2Display Header to be improved in the future. The Version Number must be set to 0b000 for implementations conforming to this first version of the Net2Display standard.

7.2.2 Virtual Channel Control Header Field Group

7.2.2.1 Control Bit (C Bit) – 1 Bit

The Control bit indicates whether the PDU is a Control PDU used for setting up or passing parameters for a Virtual Channel. This field allows any Virtual Channel ID number to be associated with the Control stream, providing flexibility in the assignment of Virtual Channel IDs. The Control Bit is used for setting up both Net2Display Defined Virtual Channels and Extended Virtual Channels. Whenever the Control Bit is set, the PDU is a Control PDU.

7.2.2.2 Extended Bit (E Bit) – 1 Bit

When the Control Bit is set, the Extended Bit indicates whether the Control PDU is operating on a Net2Display Defined Virtual Channel or an Extended Virtual Channel.

When the Control Bit is not set and the Extended bit is not set, the PDU is part of aNet2Display Defined Virtual Channel. When the Control Bit is not set and the Extended bit is not set, the ProtocolCode must take on one of the values shown defined for Net2Display Defined Virtual Channels as shown in Table 9-1.

When the Control Bit is not set, the Extended bit indicates that the PDU is part of an Extended Virtual Channel used to carry application or vendor specific data that is defined outside of the Net2Display standard. The Extended Bit allows for ease of expansion to the Net2Display standard. When the Control Bit is not set and the Extended bit is set, the following fields are required: Response Bit, Command Code, and PDU Length. When the Extended bit is set, the PDU must conform to the general PDU format, but the contents of the CommandData is protocol specific and defined outside of this standard.

7.2.2.3 Reserved Control Bits (RC Bits) – 3 Bits

These bits are reserved for Control purposes. These bits must not be interpreted by a receiving implementation.

7.2.3 Virtual Channel Identification Header Field Group

7.2.3.1 Virtual Channel ID – 24 Bits

The Virtual Channel ID identifies the traffic in a particular Virtual Channel. The Virtual Channel ID is needed to separate the different types of traffic into different streams. For both TCP and UDP, Virtual Channel numbers are needed to separate different types of data and for assigning different priorities to the data. A Virtual Channel has only one Virtual Channel Protocol Type. The Virtual Channel Protocol Type defines the protocol carried by the PDU and defines how the Command Codes are to be interpreted.

7.2.3.2 Virtual Channel Protocol Type – 6 Bits

The ProtocolCode field indicates the protocol being passed within the PDU. Including the Virtual Channel Protocol Type allows the protocol to be self describing and requires no lookup to interpret the PDUs. A Virtual Channel has only one Virtual Channel Protocol Type. The Virtual Channel Protocol Type defines the protocol carried by the PDU and defines how the Command Codes are to be interpreted.

7.2.4 Virtual Channel Specific Control Header Field Group

7.2.4.1 Reserved Header Bits (RH Bits) – 2 Bits

Reserved bits allow additions to the standard in future versions. These bits must not be interpreted by a receiving implementation.

7.2.4.2 Continuation/More Bits (CM Bits) – 2 Bits

The Continuation/More Bits are used to pass data in multiple PDUs across the network so that a single PDU will not tie up the network for an extended time, but will allow the intervening transport of data from higher priority Virtual Channels. The different combinations of the settings of the Continuation/More Bits are shown in Table 7-2. When a PDU contains the full Command Data, the CM bits are set to 0b00.

Table 7-2: Continuation/More Bits

Condition	Continuation/ More Bits (CM Bits)	Continuation Bit	More Bit	Descrpition
PDU with Full Data	0b00	0	0	PDU has full Command Data associated with the PDU Command
PDU with Start of Data	0b01	0	1	PDU has the beginning of the Command Data associated with the PDU Command
PDU with Ending Data	0b10	1	0	PDU has the ending of the Command Data associated with the PDU Command
PDU with Intermediate Data	0b11	1	1	PDU has an intermediate portion of the Command Data associated with the PDU Command

When the Command Data is divided among multiple PDUs, the first PDU has the CM Bits set to 0b01, intermediate PDUs have the CM bits set to 0b11 and the final PDU with the endig of the Command Data has the CM bits set to 0x10. For all PDUs carrying portions of the same Command Data, the following fields must be set to the same value as in the first PDU with the Command Data:

- Version Number
- Control Bit
- Extended Bit
- Virtual Channel ID
- Virtual Channel Protocol Type
- Response Bit
- Command Code
- VC_Timestamp

The following fields are set differently for the separate PDUs containing portions of the Command Data for the same PDU Command:

- Continuation/More Bits – Set to reflect the portion of the Command Data carried in this PDU.
- Sequence Number – The Sequence Number is incremented for each PDU transmitted in a Virtual Channel.
- Received Sequence Number – Set based on the last received PDU as specified in Section 7.2.4.8.

Note: The Continuation/More Bits may be regarded as the inversion of Start/End Bits, which indicate the beginning and end of the Data. By using the inverstion of the Start/End Bits, the normal state of the bits is 0.

7.2.4.3 Response Bit (Rs Bit) – 1 Bit

The Response Bit differentiates requests and responses and is set for Response PDUs. This bit allows response codes to be sent back and associated with requests. This is particularly useful for setting up Virtual Channels or for commands that require resources that may be limited.

When the Response Bit is set, the first field in the PDU Command Data is the Response Code which indicates the status of the requested operation. The ResponseCode field uses a common format for many of the different Response PDUs to ease interpretation. For a Response PDU, the ResponseCode must take on one of the Response Code values shown in Table 7-3.

7.2.4.4 Command Code – 5 Bits

The Command Code specifies the specific command carried by the PDU. Including the Command Code in the PDU header gives an early cue on how to process the PDU data before the data is encountered. Since every PDU contains a Command Code, there is no disadvantage in including it in the header.

7.2.4.5 PDU Length – 16 Bits

The PDU Length is a 16 bit unsigned integer. While a PDU Length is not needed in UDP packets, it is needed when the PDUs are transported over TCP to separate the TCP byte stream into individual PDUs. The PDU Length is included in both TCP and UDP for providing a consistent interface to the Transport layer. Since TCP is a byte stream interface, it requires a size to be included in the PDUs so that the message boundaries can be located.

The PDU Length indicates the size of the entire Net2Display PDU in bytes, including both the header and CommandData. The PDU Length is limited to 65,536 bytes.

7.2.4.6 VC_Timestamp (Optional) – 32 Bits

The optional VC_Timestamp field may be used to provide timing information for this Virtual Channel where the ordering or timing of data is important. When the VC_Timestamp field is used, it starts with a randomly chosen value and monotonically increases, incrementing for each of the Host time intervals. If the VC_Timestamp field is not used, then it contents must be set to zero. If VC_Timestamps are to be used in a Virtual Channel, then the Timing Virtual Channel Parameters must be passed during the opening of the Virtual Channel. The full detailed protocol for the value of the VC_Timestamp field is given in Section 9.6.2.

7.2.4.7 Sequence Number – 16 Bits

The Sequence Number is used to uniquely identify every transmitted PDU. The Sequence Number must start out with a pseudo random value for each Virtual Channel to prevent known text attacks on encryption. Thus, the first PDU in each Virtual Channel sets the starting Sequence Number for that Virtual Channel. The Sequence Number is incremented by one for each PDU sent in that Virtual Channel and wraps to 0x0000 after it reaches its maximum value of 0xFFFF.

7.2.4.8 Received Sequence Number – 16 Bits

The Received Sequence Number field value is different depending upon whether the PDU being sent is a Request or Response.

For Request PDUs, the Received Sequence Number indicates the highest Sequence Number that has been successfully received in the current Virtual Channel. For Request PDUs, the Received Sequence Number is used to provide status of PDU delivery.

For Response PDUs, the Received Sequence Number indicates the Sequence Number of the Request that this Response PDU is associated with. Thus, for Response PDUs, the Received Sequence Number field is used for associating Requests with Responses.

7.3 PDU Command Data

The Net2Display PDU Command Data is the data associated with the specific Command Code contained in the Net2Display PDU Header. The Command Data is variable in length and may be from 0 to 65,519 bytes, with the size of the Command Data equal to sixteen less than the value of the PDU Length field of the Net2Display PDU Header. The Command Data is optional because some Net2Display Control Commands may not require associated Command Data.

7.3.1 Control Request PDUs

Control Request PDUs contain configuration information for setting up the new Virtual Channel in the Command Data field. The content of the Command Data field in the Control Request PDUs is defined in Section 9.5.

7.3.2 Net2Display Defined Virtual Channel Request PDUs

The content of the Command Data filed in Net2Display Defined Virtual Channel Request PDUs is dependent upon the specific Net2Display Defined Virtual Channel and is defined in the document section pertaining to the individual Net2Display Defined Virtual Channel.

7.3.3 Extended Virtual Channel PDUs

The Command Data in Extended Virtual Channel PDUs is defined by the particular application or vendor using the Extended Virtual Channel and is not defined or interpretable by Net2Display Remoting. The structure for defining and using Extended Virtual Channels is defined in Section 17.

7.4 Request PDUs

Request PDUs are used as the initial PDU in any Net2Display transaction. A Response PDU may be generated by the Net2Display Node receving the Request PDU depending upon the type of Response PDU. The rules for generating Responses for Request PDUs are given in Section 7.5.

7.5 Response PDUs

Response PDUs are used to indicate the status of a requested operation. Response PDUs are not sent in response to all Request PDUs, but when they are sent, they must use the format shown in Figure 7-3**,** with all fields matching those in the corresponding Request PDU except the R bit, the Sequence Number, The ResponseCode and the OtherCommandData. The R bit is set to one to indicate a Response, the Sequence Number is set to one more than the Sequence Number of the last transmitted PDU, The ResponseCode is set to indicate the status of the Request PDU operation and the OtherCommand Data is used to transmit any data associated with the Response. OtherCommandData may include response data sent in response to a query request.

The requirements for the sending of Response PDUs in response to Request PDUs are as follows:

- For all Control Request PDUs, a Response PDU must be sent in response. The Response PDU must contain a ResponseCode value, either indicating Successful (0x0) or an error code.
    - If the ResponseCode value is Successful (0x0), the OtherCommandData is set to the Response PDU CommandData.
    - If the ResponseCode value is an error code, a value other than Successful (0x0), then the OtherCommandData must be set to the CommandData of the Request PDU.
- For all Net2Display Defined Virtual Channel Request PDUs, no response is sent when the Request PDU action completes successfully.

- If the Request PDU action does not complete successfully, then a Response PDU is sent with a ResponseCode value set to a value other than Successful and the Other CommandData is set to the Response PDU CommandData

- For all Extended Virtual Channel PDUs,

While Control Responses are required and Net2Display Defined Virtual Channel Responses are only provided on an error, the format of the Command Data is the same, with the ResponseCode appearing as the first word in the Command Data as shown in Figure 7-3. This Response PDU format is used for Net2Display Defined Virtual Channels and Control PDUs, but is not required for Extended Virtual Channel PDUs, where only the first word is required to conform to the Net2Display Header structure. Thus, the E Bit is set to zero in Figure 7-3 to indicate that the PDU is not an Extended Virtual Channel PDU.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x0	C Bit	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type			0x1	Command Code	PDU Length	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	ResponseCode						
5…	OtherCommandData						

Figure 7-3: General Response PDU Format

Note: For Figure 7-3 and other figures that show PDU formats, all fields that may take on variable values are shown as variable and fixed value fields for that PDU type may be combined to simplify the figure.

The different values that the Response Code may take on are given in Table 7-3. ResponseCodes may be Generic, Channel and Command Specific, Application Specific or Vendor Specific. Generic are useful for all types of commands and protocols and can be universally interpreted. Channel and Command Specific are specifically defined for a given Virtual Channel or command of a particular protocol type and are defined within this Standard. Application and Vendor Specific ResponseCodes are application and vendor specific and are not defined in this Standard. OtherCommandData may include response data sent in response to a query request.

Table 7-3: ResponseCode Values

Value	Description
0x00 00 00 00	Successful
0x00 00 00 01	Requested Operation unauthorized
0x00 00 00 02	Insufficient Resources to Process Request
0x00 00 00 03	Temporary Failure
0x00 00 00 04	Erroneously Formatted Request
0x00 00 00 05	Requested service not registered
0x00 00 00 06	Requested service not available
0x00 00 00 07	Invalid Parameter
0x00 00 00 08 to 0x3F FF FF FF	Reserved for other Generic Error Codes
0x40 00 00 00 to 0x7F FF FF FF	Channel and Command Specific Error Code
0x80 00 00 00 to 0xBF FF FF FF	Application Specific Error Code
0xC0 00 00 00 to 0xFF FF FF FF	Vendor Specific Error Code

Vendor Specific Failure Codes: This set of ResponseCodes is available for vendors to use to indicate vendor specific errors. One of the standard specified error codes should be used whenever possible as it can be universally interpreted.

7.6 PDU Processing

The PDU Handler is the portion of the Net2Display instance that forms Net2Display PDUs and passes them to the appropriate transport protocol and receives Net2Display PDUs from transport protocols and delivers them to other facilities in a Net2Display instance for processing.

The PDU Header and other networking headers are processed to determine both where the PDU should be delivered and how it should be processed on delivery.

7.6.1 PDU Routing Processing

Different fields within a PDU ensure that it reaches the proper destination. These fields work together to enable proper PDU routing. The fields and what they specify are:

- IP Address – Specifies the Node address for the Net2Display data.
- Net2Display TCP/UDP Port Number – Specifies that the data is Net2Display protocol data and is to be passed to the Net2Display instance in the Node. Net2Display by default uses a port number of 9086 for the Host Port for Net2Display services. If Port 9086 is not used for the Host Port, then the Client and Host must provide other means for converging on the Host Port that will be used for Net2Display communication.
- ChannelID – Specifies which data pipe the data belongs within

7.6.2 PDU Header Processing

The processing of a PDU Header is as follows:

1) Separate PDU of size PDU_Length
2) The Version Code is examined.
    a) If the Version Code is 0, then the PDU must behave according to this specification

b) Version Codes of 1 through 7 are reserved for future versions and must not be used until a later version of the Net2Display standard is specified.

3) The C Bit is examined:
   a) If the C Bit is set, the PDU is a Control PDU used for setting up or tearing down a Virtual Channel
   b) If the C Bit is not set, the PDU is a Data PDU.
4) The E Bit is examined:
   a) If the E bit is set, the PDU is a Extended PDU and only the Word 0 must conform to the Net2Display Header structure
   b) If the E bit is not set, the PDU is a Net2Display Defined Virtual Channel and the entire Header must conform to this defined structure
5) The Virtual Channel Number is used to direct the PDU to the proper receiving channel
6) The Protocol Type is then examined to determining how to process the fields in the PDU.

This processing is illustrated in Figure 7-4.

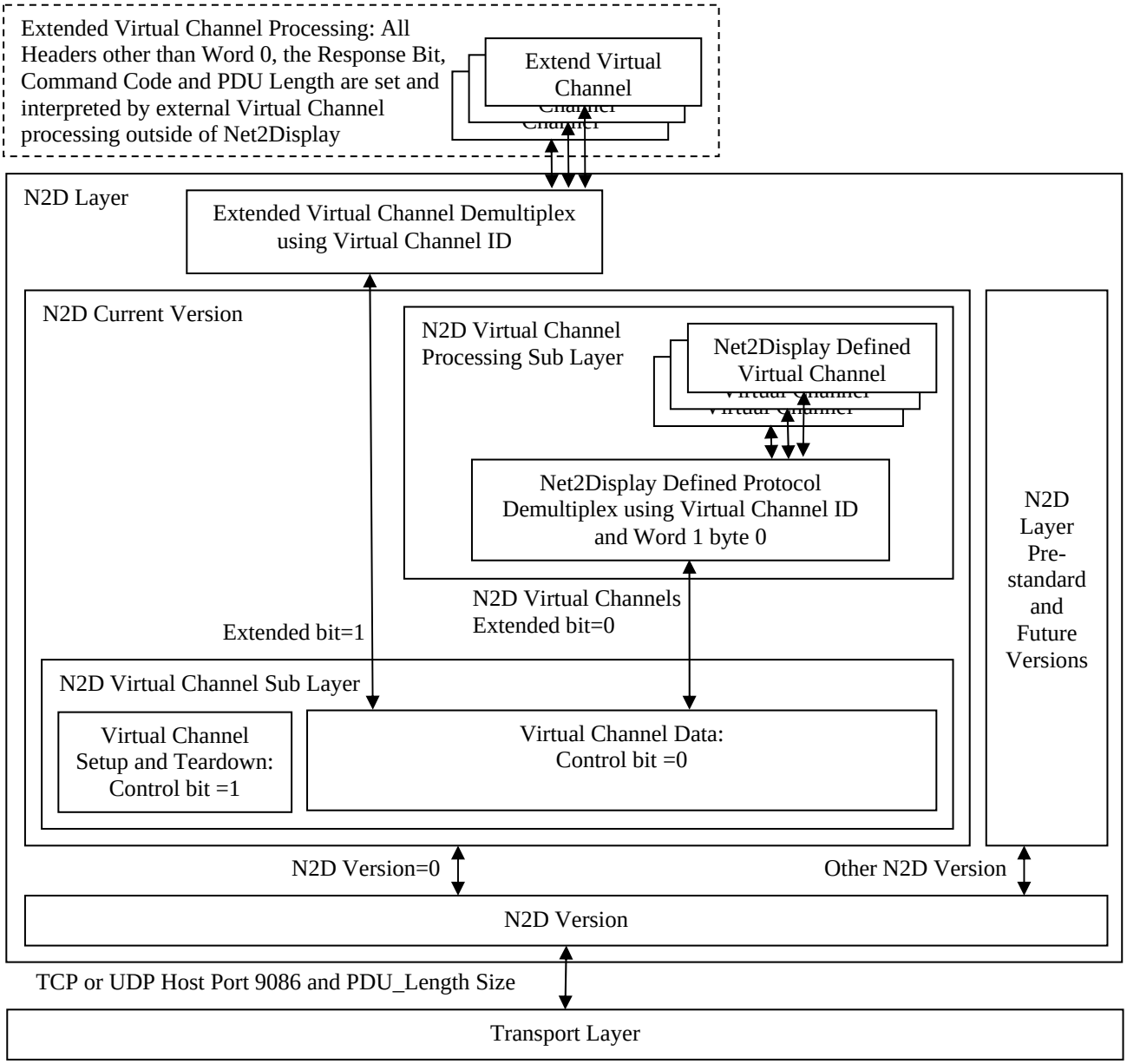


Figure 7-4: Net2Display PDU Header Processing Diagram

7.6.3 PDU Size

The Net2Display Node is responsible for selecting the size of the Net2Display PDU. While some PDU sizes are determined by the amount of data associated with the Command Code, some actions, such as drawing an image may be broken into multiple commands that can fit in multiple PDUs.

It may be advantageous to limit the size of the Net2Display PDUs for multiple reasons:

1. If PDUs are kept below the maximum size that fits in one transport packet (the network MTU), it helps to ensure prompt delivery since the PDU cannot be processed by the receiving Net2Display instance until the complete Net2Display PDU is present
2. In low bandwidth network connections PDU size should be limited. For example; four media streams (video, still image, audio and USB) sharing a TCP session and using 64k PDUs will introduce 200 msec of latency on a 10 Mb/s network connection

7.6.3.1 Combining Small PDUs into Single Transport Packet

The PDU Handler may put multiple Net2Display PDUs into a single transport packet, for efficient network bandwidth usage, and the receiving end PDU Handler will separate the Net2Display PDUs.

7.6.4 PDU Transport

Different Net2Display PDUs may be delivered across the network by both streaming transport protocols (TCP and SCTP) and datagram transport protocols (UDP). For a streaming transport protocol, the transported data is viewed as one continuous stream of bytes to be delivered to the receiving end; while for a datagram transport protocol, the transported data is viewed as a series of messages or datagrams to be delivered to the receiving end.

7.6.4.1 Streaming Transport

For a streaming transport protocol, after the data in one transport stream is transferred across the network, it is passed to the receiving Net2Display instance as a sequence of bytes always in the order in which it was presented at the transmitting Net2Display instance. Data transported by a streaming transport protocol may be delayed when a transport packet is dropped and all following data in the stream is then delayed until the dropped data is retransmitted and presented to the receiving Net2Display instance.

7.6.4.1.1 Chunking

In a streaming based session such as TCP, the PDUs form the chunking structure for multiplexing multiple media streams over a single TCP session. PDUs are indivisible, as in a PDU cannot be suspended to insert other channel data, other then by using a different priority and a different TCP session. Note that chunking at the PDU level is not the same as chunking at the TCP transport level.

7.6.4.1.2 PDU Integrity

In a TCP session, PDUs are delivered to the decoder in the same format they are sent by the encoder. Thus a codec may use the PDU as a reliable datagram inserting and receiving a command structure immediately following the PDU header. This PDU structure can be used by the Control PDUs to perform a sequence of commands in a defined order.

7.6.4.2 Datagram Transport

For a datagram transport protocol, the individual datagrams are typically presented to the receiving Net2Display instance in the order in which they arrive. Net2Display provides Sequence Numbers in PDUs to detect missing or out of order PDUs.

When a datagram transport is used such as UDP, it is the responsibility of the codecs to determine the maximum payload size and ensure PDUs do not exceed this limit. Furthermore, it is the responsibility of the codec to create and extended PDU header with sequencing numbers or transport information if these are required.

7.6.5 Data Synchronization

PDUs using a common TCP session will be delivered in the order they are sent. However, PDUs of different priority or using different network transport session are not synchronized. It is the responsibility of the media codecs to use time stamps or other means to synchronize data between PDUs using different transport sessions.

7.6.6 Data Streams in PDUs

Data streams may have no distinct chunking boundaries and may not have an implicit end. These can be handled by setting up a Virtual Channel and continuously communicating chunks of content in each PDU.

These chunks can be sent on PDUs without any additional command structure other then the PDU header. Note that this works because the virtual channel is dedicated to one media stream while control and commands use sub channels.

7.6.7 PDU Handling

The PDU Handler will not pass a PDU on to a decoder until all the data of the PDU is received. The PDU handler does not modify the PDU. The PDU handler only looks at the PDU header. All extended command structures are a responsibility of the codec or system controller.

7.6.8 PDU Ordering

PDUs that share a common priority are typically transmitted in the order they are received. If an implementation requires special ordering of different encoder content on a common TCP session, it is a responsibility of the codecs to coordinate the sending of the PDUs to the PDU Handler. For example: minimizing the queued video data to ensure timely response to audio data.

Alternatively the implementation may use a filter between the codecs and the PDU Handler to define some type of scheduling of PDUs.

8 Associations

An Association is a relationship formed between a Net2Display Host and a Net2Display Client that allows the Host display output to be transferred to the Client and other I/O to pass between the Host and Client systems. The Client will always initiate the Association with the Host, which allows the procedure to work when the Client is behind a NAT.

8.1 Association Security and Identification

An Association uses a Secure Transport to provide security across the network and to provide grouping of the Network Transports used within this Association. Any Secure Transport that provides these services may be used. Typically, IPsec is used for providing these services. Security Associations are typically one way, so a two-way connection requires at least 2 Security Associations.

For Net2Display Remoting, TLS will be used for Certificate exchange and IKEv2 will be used for IPsec Key Exchange.

Before IPsec sends authenticated or encrypted IP data, both the sender and receiver must agree on the protocols, encryption algorithms and keys to use. Net2Display IPSec uses the Internet Key Exchange (IKE) protocol to negotiate the encryption and authentication methods, and generate shared encryption keys. The IKE protocol also provides primary authentication - verifying the identity of the remote system before negotiating the encryption algorithm and keys.

When IPsec is used to provide the Secure Transport, a Security Association is formed which is uniquely identifiable by the combination of the IP Addresses, the IPsec Protocol type (ESP or AH) and the Security Parameters Index. The Security Parameters Index (SPI) is an IPsec parameter that helps the recipient select which of possibly many ongoing conversations this packet applies. Each AH-protected connection implies a hash algorithm (MD5, SHA-1, etc.), some kind of secret data, and a host of other parameters. The SPI can be thought of as an index into a table of these settings, allowing for easy association of packet with parameter.

8.2 Association Formation

The overall process leading to the formation of a Net2Display Association between a Client and Host is shown in Figure 8-1.

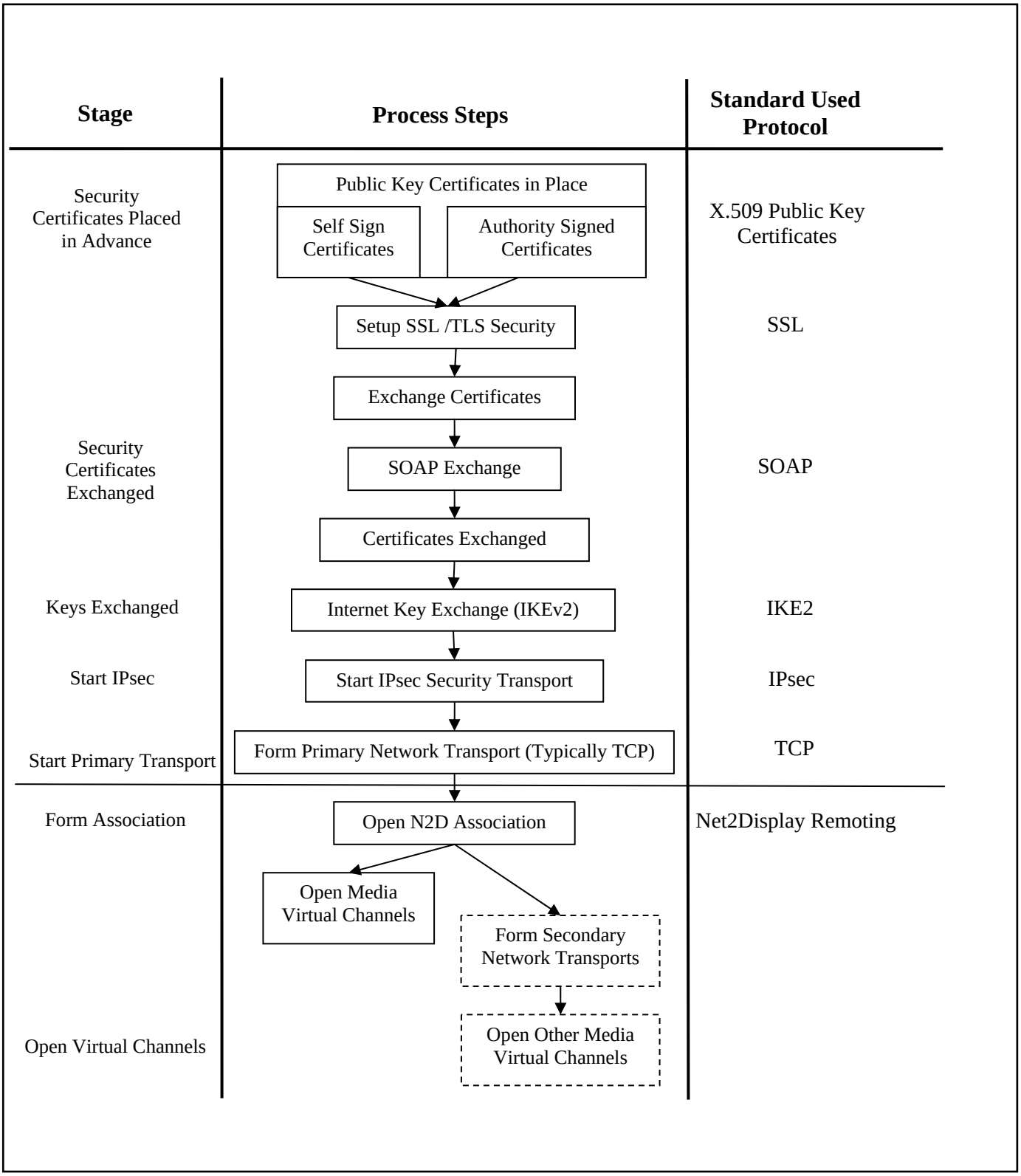


Figure 8-1: Steps in Net2Display Initialization

The procedure for starting up an Association is illustrated in Figure 8-1 and proceeds as follows:

1) **Setup SSL Security**
   a) Setup using the Net2Display port 8096
   b) Exchange security information
2) **SOAP Exchange**
   a) Client contacts Host via HTTP through SSL and requests connection.
   b) Security and Configuration Information Exchanged
   c) Agree to setup media session and exchange security information
   d) Pass all information qualifying the Socket to be used for Net2Display. Pickup the address off the HTTP request.
3) **Form Secure Transport Session**
   a) Typically, UDP encapsulated IPsec
   b) IPsec initialization
4) **Form Primary Network Transport and Association.**
   a) An Open Association Request is sent from the Client to the Host using the Primary Transport
   b) Primary Transport is setup using N2D Port on Host (Typically TCP).
   c) Client contacts Host via N2D Port Number and requests Association
   d) Host allocates for Association and responds with an accept, providing an Association_ID
5) **Additional Virtual Channels**
   a) Client or Host can now request that Virtual Channels be added to the Primary Transport using the Open Virtual Channel Request
6) **Additional Secondary Network Transports**
   a) Client or Host can then request other Virtual Channels with different Differentiated Services priorities.
      i) A new Transport may be opened by a direct request or it may be opened when a Virtual Channel with a different Differentiated Services value is requested

The different protocol stacks formed during the initialization of Net2Display remoting are shown in Figure 8-2.

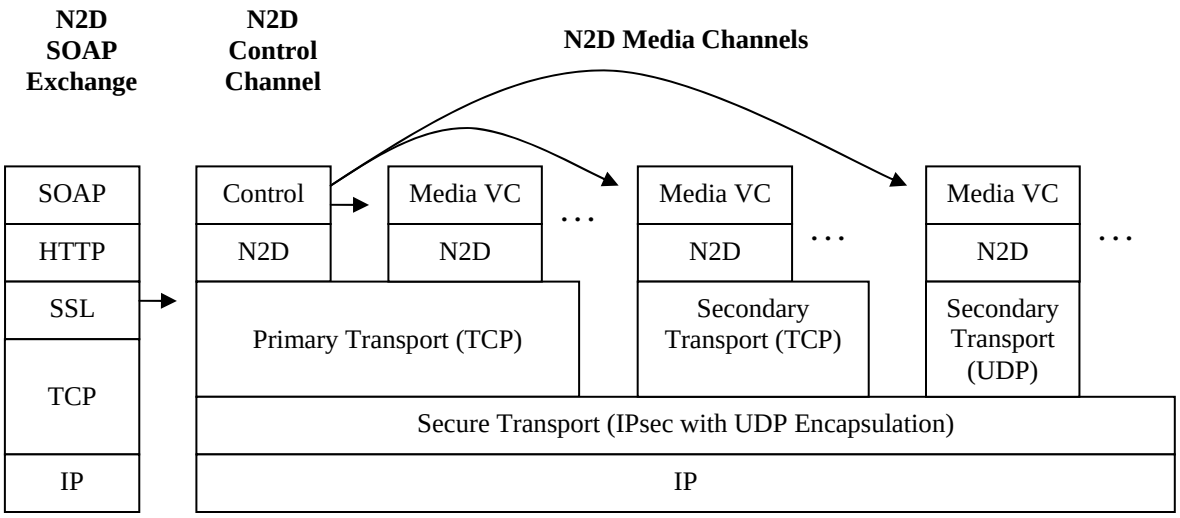


Figure 8-2: Protocol Stacks formed in Net2Display Initialization

8.3 Association Control PDUs

Net2Display uses Control PDUs for managing the formation and removal of Associations. The Command Code values for the Association Control PDUs are shown in Table 8-1. For the following Association Control PDUs that control multiple Virtual Channels with multiple Protocol Types, the Virtual Channel ID is set to zero (0x0000) and the Virtual Channel Protocol Type is set to zero (0x00).

Table 8-1: Association Control PDU CommandCode Values

Command Code	CommandCode Value	Description
Open_Association	0b01001	Sent by a Client requesting the Host to open a new Association with the Client
Pass_Association_Parameters	0b01010	Sent by a Host or a Client to pass or change one or more Association Parameters
Close Association	0b01011	Sent by a Host to open up a Virtual Channel
Reassociate	0b01100	Sent by a Client to a Host to reestablish an Association
Open New Transport	0b01101	Opens up a new Transport session
Close Transport	0b01110	Close a Transport

Note: The contents and operation of the Association Control PDUs will be further characterized by the reference implementation and will be more fully specified in the next version of this standard.

8.3.1 Open_Association

8.3.1.1 Open_Association Request

The Open_Association Request is sent by a Client as the first Net2Display PDU to request the Opening of an Association. The format of the Open_Association Request is given in Figure 8-3.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	0x00 00 00		
1	0x00			0x09	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4…	Association Parameters					

Figure 8-3: Open_Association Request PDU Format

Association Parameters: The Open Association Request PDU may include optional Association Parameters that are used to characterize the Association. The Association Parameters are defined in Section 8.4.

8.3.1.2 Open_Association Response

The Open_Association Response is sent by a Host in response to an Open_Association Request sent by a Client that is requesting the Opening of an Association. The format of the Open_Association Response is given in Figure 8-4.

The Open Association Response Association Parameters must include the Association_Identifier Parameter and the Association_Cookie Parameter as Association Parameters. The Association Identifer Parameter and Association_Cookie Paremeter are provided for later reconnection in the event that the Association is interrupted due to network failure.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	0x00 00 00		
1	0x00			0x29	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5 …	Association_Identifier Parameter					
	Association_Cookie Parameter					
	Other Association Parameters					

Figure 8-4: Open_Association Response PDU Format

Association_Identifier Paremeter: The Association_Identifier Paremeter is used to uniquely identify an Association. The Association_Identifier Paremeter is defined in Section 8.4.1.

Association_Cookie Parameter: The Association_Cookie Parameter is a randomly generated cookie as defined in Section 8.4.2.

Other Association Parameters: The Open Association Response PDU may include other optional Association Parameters that are used to characterize the Association. The Other Association Parameters are from the set of Association Parameters defined in Section 8.4.

8.3.2 Pass_Association_Parameters

8.3.2.1 Pass_Association_Parameters Request

The Pass_Association_Parameters_Request is sent to provide a new Association Parameter or to request the change of an Association Parameter. Other Association Parameters remain the same. The format of the Pass_Association_Parameter_Request is given in Figure 8-5.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x0A	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4…	Association Parameters					

Figure 8-5: Pass_Association_Parameters Request PDU Format

Association Parameters: The Pass_ Association_Parameters Request PDU may include optional Association Parameters that are used to characterize the Association. The Association Parameters are defined in Section 8.4.

8.3.2.2 Pass_Association_Parameters_Response

The Pass_Association_Paramters_Response is sent in response to an Pass_Association_Parameters Request passing Association Parameters. The format of the Pass_Association_Parameters Response is given in Figure 8-6.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x2A	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5…	Association Parameters					

Figure 8-6: Pass_Association_Parameters_Response PDU Format

ResponseCode: Data passed by the recipient to indicate the results of the Pass parameters request. The ResponseCode must take on one of the values defined in Table 7-2.

Association Parameters: The Pass_ Association_Parameters Response PDU may include optional Association Parameters that are used to characterize the Association. The Association Parameters are defined in Section 8.4.

8.3.3 Close_Association

8.3.3.1 Close_Association Request

The Close_Association Request is sent by a Client or Host to request the Closing of an Association. The format of the Close_Association Request is given in Figure 8-7. The Close Association Request includes the Association_Identifier Parameter.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x0B	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4…	Association_Identifier Parameter					

Figure 8-7: Close_Association Request PDU Format

Association_Identifier Paremeter: The Association_Identifier Paremeter uniquely identifies the Association. The Association_Identifier Paremeter is defined in Section 8.4.1.

8.3.3.2 Close_Association Response

The Close_Association Response is sent by a Host or Client in response to a Close_Association Request. The Association is closed on the sending or receiving of a Close_Association Response. The format of the Close_Association Response is given in Figure 8-8.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x2B	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5…	Association_Identifier Parameter					

Figure 8-8: Close_Association Response PDU Format

ResponseCode: Data passed by the recipient to indicate the results of the Close Association Request. The ResponseCode must take on one of the values defined in Table 7-2.

Association_Identifier Paremeter: The Association_Identifier Paremeter uniquely identifies the Association. The Association_Identifier Paremeter is defined in Section 8.4.1.

8.3.4 Reassociate

The Reassociate process is used to allow a Client to reconnect to a Host after a short term network failure or a change of network media.

8.3.4.1 Reassociate Request

The Reassociate Request is sent by a Client to request a reconnection of a dropped Association. The format of the Reassociate Request is given in Figure 8-9.

Note: The Reassociate process is based on the Automatic Recommnection Process with Enhanced RDP Security used by the Microsoft Windows Remote Desktop Protocol and described in the Microsoft Document: Remote Desktop Protocol: Basic Connectivity and Graphics Remoting Specification.

<table>
<tr><td>Word</td><td colspan="3">Byte 0</td><td>Byte 1</td><td>Byte 2</td><td>Byte 3</td></tr>
<tr><td>0</td><td>0x1</td><td>E
B
i
t</td><td>0x0</td><td colspan="3">0x00 00 00</td></tr>
<tr><td>1</td><td colspan="3">0x00</td><td>0x0C</td><td colspan="2">PDU Length</td></tr>
<tr><td>2</td><td colspan="6">Reserved Word</td></tr>
<tr><td>3</td><td colspan="4">Sequence Number</td><td colspan="2">Received Sequence Number</td></tr>
<tr><td rowspan="3">4…</td><td colspan="6">Association_Identifier Parameter</td></tr>
<tr><td colspan="6">Security_Verifier Parameter</td></tr>
<tr><td colspan="6">Other Association Parameters</td></tr>
</table>

Figure 8-9: Reassociate Request PDU Format

Association_Identifier Paremeter: The Association_Identifier Paremeter uniquely identifies the Association. The Association_Identifier Paremeter is defined in Section 8.4.1.

Security_Verifier Parameter: The Security_Verifier Parameter is generated by the Client as defined in Section 8.4.3.

Other Association Parameters: The Reassociate Request PDU may include other optional Association Parameters that are used to characterize the Association. The Other Association Parameters are from the set of Association Parameters defined in Section 8.4.

8.3.4.2 Reassociate Response

The Reassociate Response is sent by a Host in response to an Reassociate Request sent by a Client that is requesting the reforming of an Association. The format of the Reassociate Response is given in Figure 8-10.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x2C	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5…	Association_Identifier					
	Other Association Parameters					

Figure 8-10: Reassociate Response PDU Format

ResponseCode: Data passed by the recipient to indicate the results of the Reassociate request. The ResponseCode must take on one of the values defined in Table 7-2.

Association_Identifier Paremeter: The Association_Identifier Paremeter uniquely identifies the Association. The Association_Identifier Paremeter is defined in Section 8.4.1.

Other Association Parameters: The Reassociate Response PDU may include other optional Association Parameters that are used to characterize the Association. The Other Association Parameters are from the set of Association Parameters defined in Section 8.4.

8.3.5 Open_New_Transport

8.3.5.1 Open_New_Transport Request

The Open_New_Transport Request is sent by a Client or a Host to request the opening of a new transport session within the Association. The format of the Open_New_Transport Request is given in Figure 8-11.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x0D	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4…	Transport Parameters					

Figure 8-11: Open_New_Transport Request PDU Format

Transport Parameters: The Transport Parameters are one or more of the transport parameters defined in Section 9.6.1.

8.3.5.2 Open_New_Transport Response

The Open_New_Transport Response is sent by a Host or Client in response to an Open_New_Transport Request. The format of the Open_New_Transport Response is given in Figure 8-12.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x2D	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5…	Transport Parameters					

Figure 8-12: Open_New_Transport Response PDU Format

ResponseCode: Data passed by the recipient to indicate the results of the open tranport request. The ResponseCode must take on one of the values defined in Table 7-2.

Transport Parameters: The Transport Parameters are one or more of the transport parameters defined in Section 9.6.1.

8.3.6 Close_Transport

8.3.6.1 Close_Transport Request

The Close_Transport Request is sent by a Client or Host to request the Closing of a Transport Session. The format of the Close_Transport Request is given in Figure 8-13.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E B i t	0x0	0x00 00 00		
1	0x00			0x0E	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4…	TransportParameters					

Figure 8-13: Close_Transport Request PDU Format

Transport Parameters: The Transport Parameters are one or more of the transport parameters defined in Section 9.6.1.

8.3.6.2 Close_Transport Response

The Close_Transport Response is sent by a Host or Client in response to a Close_Transport Request. The format of the Close_Transport Response is given in Figure 8-14. The Transport is closed on the transmission or reception of the Close_Transport Response PDU.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	0x00 00 00		
1	0x00			0x2E	PDU Length	
2	Reserved Word					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5…	Transport Parameters					

Figure 8-14: Close_Transport Response PDU Format

ResponseCode: Data passed by the recipient to indicate the results of the Close Transport request. The ResponseCode must take on one of the values defined in Table 7-2.

Transport Parameters: The Transport Parameters are one or more of the transport parameters defined in Section 9.6.1.

8.4 Association Control PDU Parameters

8.4.1 Association_Identifier

The Association_Identifier is a unique identifier for the Association. The Association_Identifier is passed on the opening of the Association and on reconnections of the Association.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8012		0x0004	
1	Association_Identifier			

Figure 8-15: Association_Identifier Parameter Representation

Parameter_Type: A 16 bit value set to 0x8012.

Parameter_Length: A 16 bit value set to the length of the Association_Identifier in bytes (0x0004).

Association_Identifier: A 32 bit unique identifier for this Association

8.4.2 Association_Cookie Parameter

The Association_Cookie is a cookie randomly generated by the Host and used for Reassociation. The Association_Cookie Parameter is passed on the opening of the Association and is used to generate the Security_Verifier used on reconnections of the Association. The Association_Cookie containes a 16 byte cryptographically secure random number and is formatted as shown in Figure 8-16. When a Client receives an Association_Cookie, the Client never allows programmatic access to it.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8012		0x0010	
1-4	Association_Cookie			

Figure 8-16: Association_Cookie Parameter Representation

Parameter_Type: A 16 bit value set to 0x8012.

Parameter_Length: A 16 bit value set to 0x0010

Association_Cookie: A 128 bit random number generated by the Host

8.4.3 Security_Verifier Parameter

The Secirity Verifier Parameter is a parameter generated by the Client derived from the Association_Cookie and used for Reassociation. The Security_Verifier Parameter is generated and placed in a Reassociate Request PDU when a reconnection attempt is required. The format of the Security_Verifier Parameter is shown in Figure 8-17.

The Client derives the 16-byte Seciruty Verifier Parameter from the Association_Cookie Parameter that it received from the Host when the Association was formed. This Security_Verifier Parameter is placed in the Reassociate Request PDU and sent to the Host.

The Association_Cookie Parameter is used to key the HMAC function (defined in IETF RFC 2104), which uses MD5 as the iterative hash function. The Security_Verifier is derived by applying the HMAC to a series of 16 zero bytes:

Security_Verifier = HMAC(Association_Cookie, 16 zero bytes)

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8012		0x0010	
1-4	Security_Verifier			

Figure 8-17: Security_Verifier Parameter Representation

Parameter_Type: A 16 bit value set to 0x8012.

Parameter_Length: A 16 bit value set to 0x0010

Security_Verifier: A 128 bit (16 byte) verifier value derived using cryptographic methods (as specified above) from the Association_Cookie Parameter (See Section 8.4.2) of the Open Association Response PDU (See Section 8.3.1.2).

8.4.4 Maximum Association Bandwidth

The Maximum Association Bandwidth specifies the total bandwidth available for this Association when there is no congestion experienced on the network.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8013		0x0004	
1	Maximum_Association_Bandwidth			

Figure 8-18: Maximum Association Bandwidth Parameter Representation

Parameter_Type: A 16 bit value set to 0x8013.

Parameter_Length: A 16 bit value set to 0x0004

Maximum_Association_Bandwidth: A 32 bit value that specifies the bandwidth available for the Association in kbps

8.4.5 Allocated Association Bandwidth

The Allocated Association Bandwidth specifies the bandwidth available for this Association when the network path is congested. This

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8014		0x0004	
1	Allocated_Association_Bandwidth			

Figure 8-19: Allocated Association Bandwidth Parameter Representation

Parameter_Type: A 16 bit value set to 0x8014.

Parameter_Length: A 16 bit value set to 0x0004

Allocated_Association_Bandwidth: A 32 bit value that specifies the bandwidth available for the Association in kbps

9 Virtual Channels

Virtual Channels are used for remoting media from the Host to the Client. Virtual Channels allow different types of media to be remoted independently without blocking and using different transport services at different priority levels. A Virtual Channel operates like an independent connection between the Host and the Client as shown in **Figure 9-1**. Depending upon the type of media being carried, some of these Virtual Channels use a reliable transport mechanism, while other media can be carried using unreliable transport.

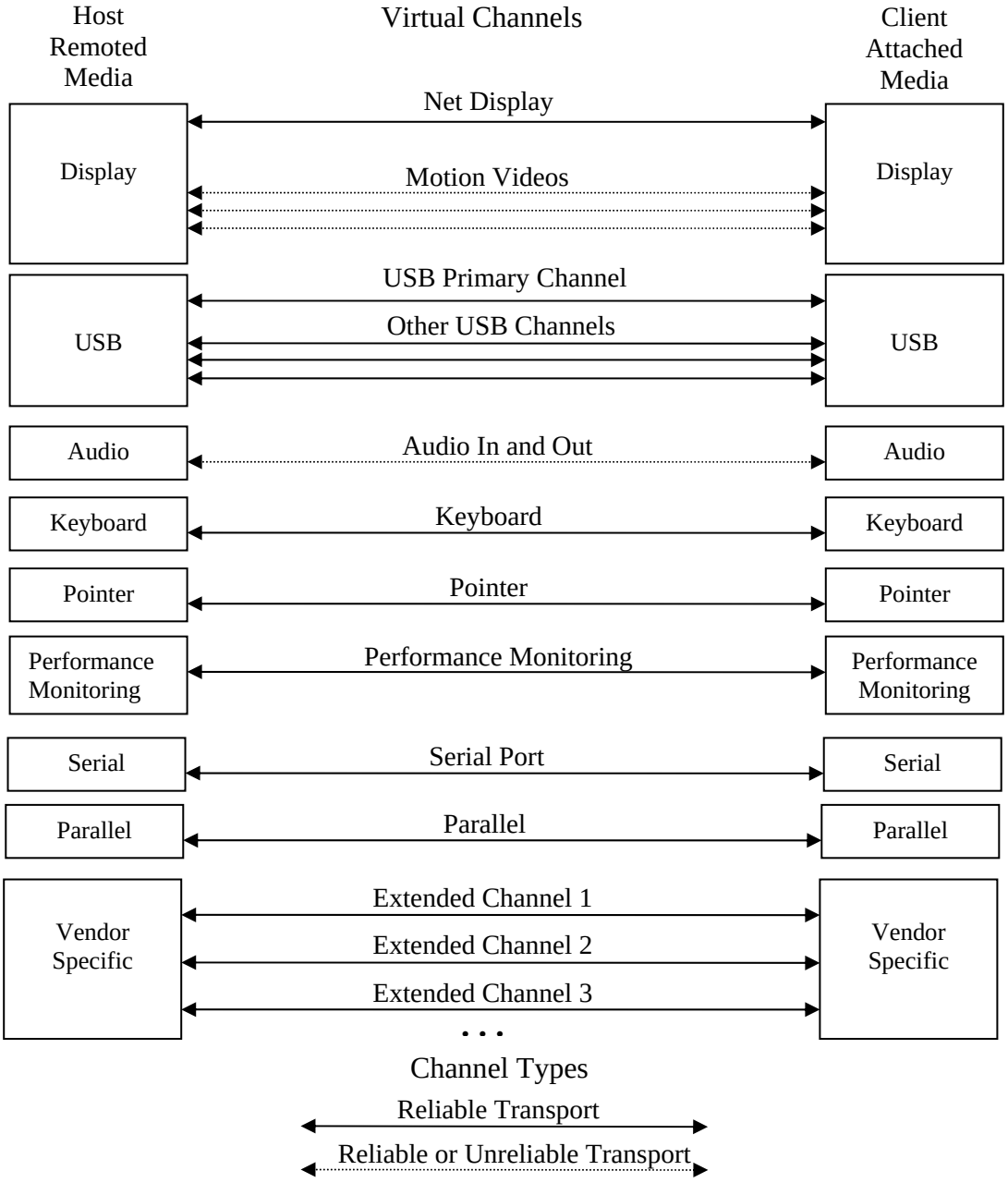


Figure 9-1: Virtual Channel Remoting

As seen in **Figure 9-1**, many different types of information can be remoted using Virtual Channels. Some of the different types of information that can be separated into Virtual Channels include:

- Net Display – used for sending still display content or slowly updated display content. Motion Video can be sent using the Net Display Virtual Channel if the Motion Video Virtual Channel is not supported or available. Since Display content is relatively static, it is sent using a reliable transport protocol so that the results of transmission errors are not persistent on the display
- Motion Video – used for sending a single motion video stream that fits within a rectangular area or a masked rectangular area
- Motion Video Control – used for controlling a single motion video stream carried by a single Motion Video Virtual Channel
- USB Primary Channel – used for carrying USB Control type information and carrying all USB traffic when only one Virtual Channel is used for USB
- USB Other Channels – used for carrying additional USB data
- USB Isochronous – used for carrying USB Isochronous traffic, such as audio or captured video
- USB Bulk – used for carrying USB Bulk traffic, such as disk and flash storage data
- Audio Out – used for carrying an audio stream for output on the Client from speakers or headphones
- Audio In – used for carrying an audio input stream as would be acquired from a microphone
- Keyboard – used for carrying keyboard input and control.
- Pointer – used for carrying pointer movement and button presses from mice, touchscreens
- Serial Port – provided for backwards compatibility for systems requiring serial ports
- Parallel Port – provided for backwards compatibility for systems requiring parallel ports
- Performance Monitoring – used for passing periodic alternating short and long data packets for measuring the bandwidth and latency between the Host and the Client
- Extended Channels – used for carrying vendor defined data, negotiated by Net2Display Remoting, but not defined explicitly in this Net2Display Remoting standard.

9.1 Virtual Channel Types

Net2Display Remoting defines a number of different Virtual Channel types for handling different kinds of data with different delivery requirements and priorities. A summary of the required and optional Net2Display Defined Virtual Channels are shown in Table 9-1.

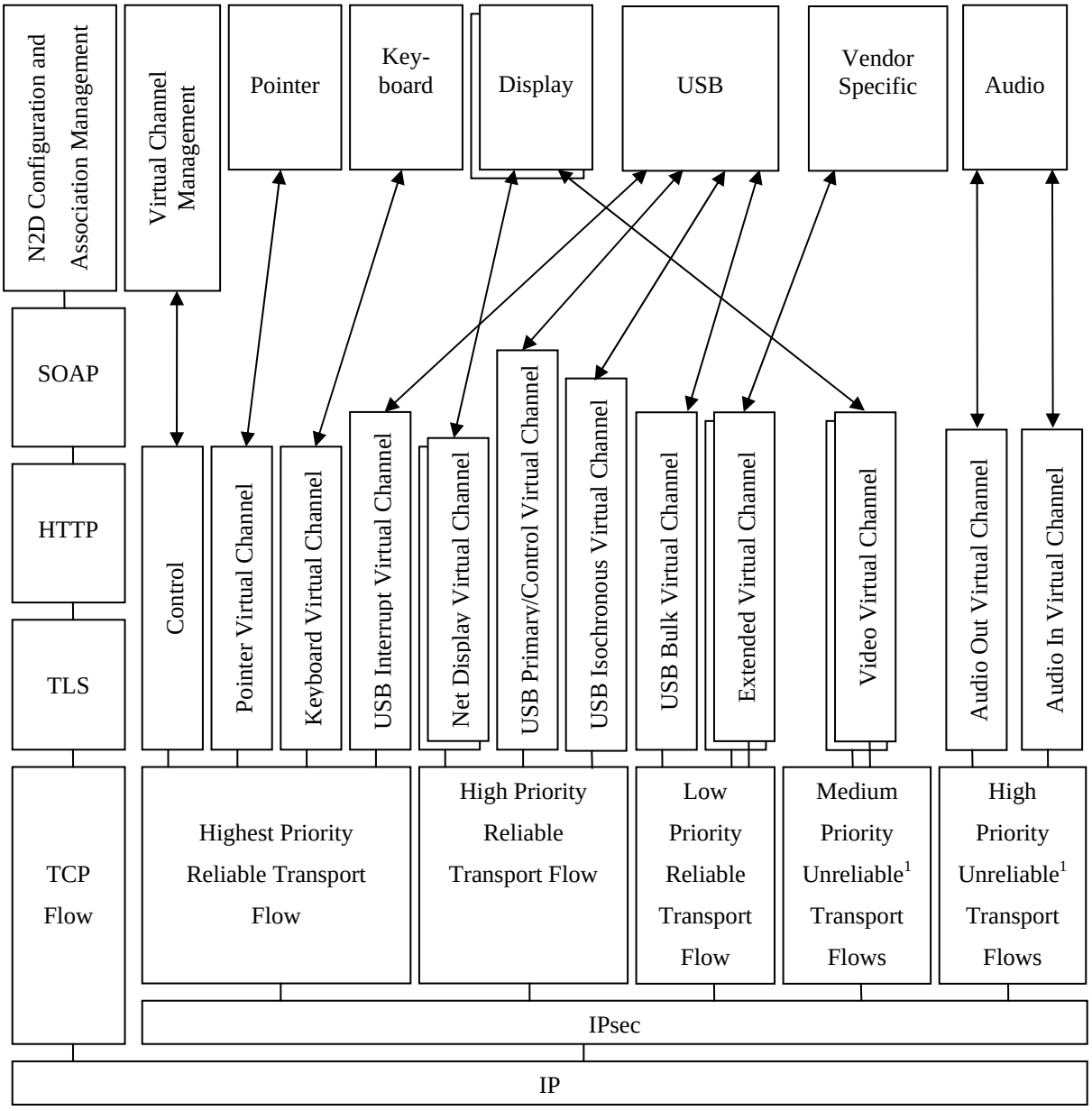
Table 9-1: Net2Display Defined Virtual Channel Protocol Types

Virtual Channel Type	Document Section	Virtual Channel Protocol Type Field	Number of Channels	Default Transport Protocol	Default Virtual Channel Priority	Example
Association Control	8.3	0b000000	One	TCP	High	Open Association PDU
Net Display	11	0b000001	One per Net Display	TCP	High	Windows
Keyboard	13	0b000010	One	TCP	Very High	Keyboard
Pointer	0	0b000011	One	TCP	Highest	Mouse, Mouse Clicks & Pointer Icons
USB	13	0b0001XX (See Section 13)	Number mutually supported by Host and Client	TCP	Low to Very High (See Section 13)	USB Traffic: USB mouse, keyboard, audio, captured motion video, Disk, Flash
Audio Out	16	0b001000	One	UDP	High	Speaker Out
Audio In	16	0b001001	One	UDP	High	Microphone
Motion Video	12	0b001010	One per motion video stream	UDP	Medium	Motion Video Stream
Motion Video Control	C	0b001011	One per motion video stream	TCP	Medium	Video Stream Control: Play, Pause, Fast Forward
Serial Port	E	0b001100	One per Port	TCP	Medium	Legacy Serial Port
Parallel Port	F	0b001101	One per Port	TCP	Medium	Legacy Parallel Port
Performance Monitoring	C	0b001110	One	TCP	Medium	Test packets to measure bandwidth and latency
Reserved	N/A	0b001111 to 0b011111	N/A	TCP	Low	N/A

Note: While defaults for transport types and Virtual Channel Priorities are defined in this table, their values may be changed by issuing a command to pass the updated Virtual Channel parameters

9.2 Virtual Channel Transport

The relationship of these Virtual Channels to the different network layers are shown in Figure 9-2. Typically, the reliable transport will be TCP and the unreliable transport will be UDP, but other transports may be negotiated and used. A single TCP session may carry multiple different Virtual Channels, so that a simple Net2Display implementation may use a single TCP session to carry all the different Net2Display Virtual Channels together.



[1]**Note:** If UDP or other unreliable transport is not available for use, then Reliable Transport must be used with the same priority level.

Figure 9-2: Transport of Virtual Channels

9.3 Virtual Channel Priorities

Net2Display Remoting provides the option of using different levels of Virtual Channel Priorities for the processing and transport of different types of data. Virtual Channel Priorities are established and communicated during the formation of the Virtual Channels. If the Virtual Channel Priority is not communicated as a parameter during the opening of the Virtual Channel, then the Virtual Channel will use Virtual Channel Priority as specified in Table 9-1. The Virtual Channel Priority must be set to one of the priorities defined in Table 9-2. As seen in Table 9-2, for each Virtual Channel Priority that is defined, there is an associated IP Differentiated Services value and a setting for the DSCP Bits that are placed in the IP Header.

Note: The use of the Virtaul Channel Priority to prioritize the queuing and processing of different types of data within a Net2Display Node is implementation specific. Multiple queues may be used internal to a Net2Display impelemantion to provide different levels of priorities of processing, transmission and reception.

Table 9-2: Virtual Channel Priorities

Virtual Channel Priority Name	Virtual Channel Priority Numeric Value	Differentiated Services Class Designation	DSCP Bits in IP Header
Highest	16	EF	0x3E
Very High	12	EF	0x3E
High	8	AF33	0x30
Medium	4	AF22	0x20
Low	0	Default	0x00

9.4 Virtual Channel Formation

This section describes the different processes that contribute to the formation of Virtual Channels.

9.4.1 Configuration Information Exchange during Association Formation

During the exchange of configuration information, the Host and the Client exchange the following information that affects the configuration of Virtual Channels:

- Maximum number of TCP connections supported. The number of supported Transport connections determines how many different DiffServ values can be used to support different priorities of TCP across the Association. The minimum of the number of supported Transport connections communicated by both the Host and the Client sets the maximum on the number of DiffServ values that may be used for TCP transport.
- Set of Data Channels that require Virtual Channels – Determines the Initial number of Virtual Channels that are to be setup.
- Bandwidth Allocation requested for each Virtual Channel

9.4.2 Initial Virtual Channel Startup

Initially, all of the Virtual Channels are setup from the Host to the Client. The Host sends a Virtual_Channel_Open_Request for each Virtual Channel that is required. In the Virtual_Channel_Open_Request, the Host includes:

1) A Sequence Number so that multiple Virtual Channel open requests can be outstanding at the same time between the same IP addresses and ports. This prevents requiring all open requests for multiple displays to be serialized.
2) A Requested Virtual Channel ID. This will be confirmed as available for use by the Client
3) The Priority to be used by this Virtual Channel for both internal processing and the priority to be used for IP Differentiated Services
4) The type of Transport required: Reliable, In-Order, Low Latency
5) The Bandwidth Allocation requested on the Virtual Channel
6) Other Virtual Channel specific control information

A Client responds to a request to open a Virtual Channel, with a Virtual_Channel_Open_Response including:

1) The Sequence Number to associate with the Host request.
2) The Virtual Channel ID
3) The Bandwidth Allocation from the Client
4) A ResponseCode to indicate the results of the operation
5) Other Virtual Channel specific control information

9.4.3 Additional Virtual Channels Requested by the Client

When additional channels are needed by a Client beyond what was initially established, a Virtual_Channel_Needed_Request is sent for each new Virtual Channel that is needed by the Client. In the Virtual_Channel_Needed_Request, the Client includes much of the information to be included in the Virtual_Channel_Open_Request:

1) A Sequence Number so that multiple Virtual Channel open requests can be outstanding at the same time between the same IP addresses and ports. This prevents requiring all open requests for multiple displays to be serialized.
2) The Priority to be used by this Virtual Channel for both internal processing and the priority to be used for IP Differentiated Services
3) The type of Transport requested: Reliable, In-Order, Low Latency
4) The Bandwidth Allocation requested on the Virtual Channel
5) Other Virtual Channel specific control information

The Host will then follow the Virtual_Channel_Needed_Request with a Virtual_Channel_Needed_Response which only includes a Sequence Number and a Response Code.

Following the Virtual_Channel_Needed_Response, the Host will send a Virtual_Channel_Open_Request with all of the information and the process detailed in Section 9.4.2.

9.4.4 Additional Virtual Channels Added by the Host

When a Host requires an additional Virtual Channel after all the initial Virtual Channels have been setup, the Host initiates this process by sending a Virtual_Channel_Open_Request. This Virtual_Channel_Open_Request is sent with all of the information and the process detailed in Section 9.4.2.

9.5 Virtual Channel Control PDUs

Net2Display Control PDUs are used for setting up and managing Virtual Channels. There is a separate Control bit (C Bit) that is set to indicate Control PDUs so that control information does not require a Virtual Channel for communication. All Net2Display Control PDUs are sent using TCP to ensure reliable delivery of control information and to ensure consistency of control state. The same set of Command Code values are used for request and response PDUs and may be used to associate responses with the outstanding request. The Command Code values for the Virtual Channel Control PDUs are shown in Table 9-3.

Table 9-3: Virtual Channel Control PDU CommandCode Values

Command Code	CommandCode Value	Description
Virtual_Channel_Needed	0b00001	Sent by a Client requesting the Host to open a new Virtual Channel
Virtual_Channel_Open	0b00010	Sent by a Host to open up a Virtual Channel
Virtual_Channel_Change_Parameters	0b00011	Sent by a Host or a Client to change one or more Virtual Channel parameters
Virtual_Channel_End	0b00100	Ends an open Virtual Channel

9.5.1 Virtual Channel Open PDUs

Different types of command and data are divided into separate communications channels, called Virtual Channels. Virtual Channels are used for passing display, video, pointer, keyboard, USB data and control and other types of data across the network. The division of the different Commands types into different Channels may require more networking resources, but provides higher performance service by allowing interactive traffic to take priority over bulk data traffic. These different PDUs can also be processed with different priorities to give the fastest display responsiveness.

The Virtual Channel Open PDUs are used for initializing a Virtual Channel the different PDUs involved in the Virtual Channel Open Process are:

- Virtual_Channel_Needed_Request – Used by a Client to request the Host to open a new Virtual Channel
- Virtual_Channel_Open_Request – Used by a Host to open a new Virtual Channel
- Virtual_Channel_Open_Response – Used by a Client to indicate the status of the Virtual_Channel_Open_Request to the Host

These different PDUs that form the Virtual Channel Open Process contain Virtual Channel Parameters that are used for setting up the properties of the Virtual Channel.

9.5.1.1 Virtual_Channel_Open_Request

The Virtual_Channel_Open_Request is sent by a Host that requires a new Virtual Channel to be formed. The format for the Virtual_Channel_Open_Request PDU is given in Figure 9-3. The E Bit will be set if it is an Extended Virtual Channel that is requested.

<table>
<tr><td>Word</td><td colspan="3">Byte 0</td><td>Byte 1</td><td>Byte 2</td><td>Byte 3</td></tr>
<tr><td>0</td><td>0x1</td><td>E Bit</td><td>0x0</td><td colspan="3">Virtual Channel ID</td></tr>
<tr><td>1</td><td colspan="2">Virtual Channel Protocol Type</td><td>0x0</td><td>0x02</td><td colspan="2">PDU Length</td></tr>
<tr><td>2</td><td colspan="5">VC_Timestamp (Optional)</td></tr>
<tr><td>3</td><td colspan="4">Sequence Number</td><td>Received Sequence Number</td></tr>
<tr><td>4…</td><td colspan="5">VirtualChannelOpenParameters</td></tr>
</table>

Figure 9-3: Virtual_Channel_Open_Request PDU Format

VirtualChannelParameters: The set of parameters needed to negotiate the start of a Virtual Channel.

ChannelAllocation: The ChannelAllocation specifies the resources that are allowed to be used by this channel.

9.5.1.2 Virtual_Channel_Open_Response

The format of the Virtual_Channel_Open_Response PDU is given in Figure 9-4. The E Bit will be set if it is an Extended Virtual Channel that was requested.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type		0x0	0x22	PDU Length	
2	VC_Timestamp (Optional)					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5 …	VirtualChannelParameters					

Figure 9-4: Virtual_Channel_Open_Response PDU Format

The Virtual_Channel_Open_Response PDU must set the Virtual Channel ID equal to the fields in the Virtual_Channel_Open_Request.

ResponseCode: Data passed by the Client to indicate the results of open request. The ResponseCode must take on one of the values defined in Table 7-3.

VirtualChannelParameters: The set of parameters needed to negotiate the start of a Virtual Channel. If the recipient of a Virtual_Channel_Open_Request is capable of supporting the requested channel allocation, then it responds with the same value of Channel Allocation as in the request PDU. If the recipient of a Virtual_Channel_Open_Request does not have the resources to support the requested channel allocation as requested, it responds with ChannelAllocation set to the channel allocation that it is capable of supporting.

9.5.2 Virtual Channel Pass Parameter PDUs

The Virtual Channel Pass Parameters PDUs are used to change or pass new parameters on a Virtual Channel without having to shut down the Virtual Channel. The Virtual Channel Pass Parameter exchange can be initiated by either a Host or a Client.

9.5.2.1 Virtual_Channel_Pass_Parameters_Request

The Virtual_Channel_Pass_Parameters_Request PDU is sent by a Host or a Client to change or pass one or more Virtual Channel parameters. The format for the Virtual_Channel_Pass_Parameters_Request PDU is given in Figure 9-5. The E Bit will be set if it is an Extended Virtual Channel that is changing parameters.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type		0x0	0x03	PDU Length	
2	VC_Timestamp (Optional)					
3	Sequence Number				Received Sequence Number	
4…	VirtualChannelParameters					

Figure 9-5: Virtual_Channel_Pass_Parameters_Request PDU Format

VirtualChannelParameters: The set of parameters requested to be passed or changed for the Virtual Channel.

9.5.2.2 Virtual_Channel_Pass_Parameters_Response

The format of the Virtual_Channel_Pass_Parameters PDU is given in Figure 9-4. The E Bit will be set if it is an Extended Virtual Channel that is changing parameters.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type		0x0	0x23	PDU Length	
2	VC_Timestamp (Optional)					
3	Sequence Number				Received Sequence Number	
4	ResponseCode					
5 …	VirtualChannelParameters					

Figure 9-6: Virtual_Channel_Pass_Parameters_Response PDU Format

The Virtual_Channel_Pass_Parameters_Response PDU must set the Virtual Channel ID equal to the fields in the Virtual_Channel_Pass_Parameters_Request.

ResponseCode: Data passed by the recipient to indicate the results of the Pass parameters request. The ResponseCode must take on one of the values defined in Table 7-3.

VirtualChannelParameters: The set of parameters that were requested to be passed or changed and their latest values as allowed by the recipient of the pass request.

9.5.3 Virtual Channel End PDUs

The Virtual Channel End is used to terminate a Virtual Channel. A Virtual Channel may be terminated by either the Host or the Client end of the Association. A Virtual Channel is not fully terminated until the Virtual_Channel_End_Response PDU is received.

9.5.3.1 Virtual_Channel_End_Request PDU

The format of the Virtual_Channel_End_Request PDU is given in Figure 9-7. The E Bit will be set if it is an Extended Virtual Channel that is being ended.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type		0x0	0x04	PDU Length	
2	VC_Timestamp (Optional)					
3	Sequence Number				Received Sequence Number	
4…	ReasonCode					

Figure 9-7: Virtual_Channel_End_Request PDU Format

The Virtual Channel ID field of the Virtual_Channel_End_Request PDU must match the fields of the Virtual_Channel_Open_Request PDU that initiated the Virtual Channel.

ReasonCode: The ReasonCode gives the reason for terminating the Virtual Channel. The set of possible ReasonCodes are given in Table 9-4.

Table 9-4: Virtual Channel End Request Reason Codes

ReasonCode	Description
0x00 00 00 03	Temporary Failure
0x00 00 00 09	Virtual Channel no longer needed
0x80 00 00 00 to 0xFF FF FF FF	Vendor Specific Reason Codes

9.5.3.2 Virtual_Channel_End_Response PDU

The format of the Virtual_Channel_End_Response PDU is given in Figure 9-8.

Word	Byte 0			Byte 1	Byte 2	Byte 3
0	0x1	E Bit	0x0	Virtual Channel ID		
1	Virtual Channel Protocol Type		0x0	0x24	PDU Length	
2	VC_Timestamp (Optional)					
3	Sequence Number				Received Sequence Number	
4…	ResponseCode					

Figure 9-8: Virtual_Channel_End_Response PDU Format

The Virtual Channel ID field of the Virtual_Channel_End_Response PDU must match the fields of the Virtual_Channel_End_Request PDU.

ResponseCode: Data passed by the recipient to indicate the results of the end request. The ResponseCode must take on one of the values defined in Table 7-3.

9.6 Virtual Channel Parameters

There are four classes of Virtual Channel Parameters:

- **General Virtual Channel Parameters** are applicable to all Virtual Channels and include transport type, available bandwidth, and priority. General Virtual Channel Parameters may be optionally present in both Net2Display Defined Virtual Channel Open and Extended Virtual Channel Open requests. General Virtual Channel Parameters are defined in Section 9.6.1. The most significant bit in the first byte is set for General Virtual Channel Parameters to distinguish them from Net2Display Defined and Extended Virtual Channel Parameters.
- **Timing Virtual Channel Parameter is** applicable to all Virtual Channels that use timestamps, potentially both Net2Display Defined Virtual Channels and Extended Virtual Channels. The Timing Virtual Channel Parameter is defined in Section 9.6.2.
- **Net2Display Defined Virtual Channel Parameters** are specific to a Net2Display Defined Virtual Channel Protocol Type. Net2Display Defined Virtual Channel Parameters are only found in Virtual Channel Open Requests for Net2Display Defined Protocols. The Net2Display Defined Virtual Channel Parameters are defined in each of the document sections describing the Net2Display Defined Virtual Channel protocols.
- **Extended Virtual Channel Parameters** are specific to the Extended Virtual Channel. Extended Virtual Channel Parameters are only present in Virtual Channel Open Requests for Extended Virtual Channels. There are two Extended Virtual Channel Parameters that are required in all Extended Virtual Channel Open requests which are defined in Section 9.6.3. The remaining Extended Virtual Channel Parameters are vendor protocol specific and are not defined in this specification. However, the representation of the Extended Virtual Channel Parameters must conform to the Virtual Channel Open Parameter Representation specified in this document section.

The same format is used for the three different Virtual Channel Open Parameter types. All parameters in the body of Virtual Channel Open PDUs are expressed in a type-length-value (TLV) format. These parameters, called VirtualChannelParameters, are each represented using 16 bits to express the type of parameter, 16 bits to express the length of the value in bytes and a variable size field for the specific value being passed. This representation is shown in Figure 9-9.

Additional parameters beyond those specified in this standard are not required to be interpreted by an implementation conforming to this version of the Net2Display standard. A Net2Display instance must be capable of parsing the Virtual Channel Parameter format and skipping parameters that it is not capable of processing.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Parameter_Type		Parameter_Length	
1…	Parameter_Value			

Figure 9-9: Virtual Channel Parameter Representation

Parameter_Type: A 16 bit value that represents the specific type of parameter that is being encoded.

Parameter_Length: A 16 bit value that represents the length of the encoded value in bytes.

Parameter_Value: The value of the parameter being encoded. If the Parameter_Value is not a multiple of four bytes, the Parameter_Value is followed by padding so that it ends on a word boundary.

9.6.1 General Virtual Channel Parameters

The following Virtual Channel Parameters may be optionally present in all Virtual Channel Open PDUs. If these parameters are not included, the default Virtual Channel characteristics specified in Table 9-1 are used for Net2Display Defined Virtual Channels.

9.6.1.1 Virtual Channel Transport

The Virtual Channel Transport specifies the Value that will be used in the Protocol field of the IP Header for the Virtual Channel being opened. Standard IP Transport protocols are specified by IANA at http://www.iana.org/assignments/protocol-numbers.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8001		0x0004	
1	Virtual_Channel_Transport_Value			

Figure 9-10: Virtual Channel Transport Parameter Representation

Parameter_Type: A 16 bit value set to 0x8001.

Parameter_Length: A 16 bit value set to 0x0004

Virtual_Channel_Transport_Value: The value of the Transport Protocol identifier as given by IANA at http://www.iana.org/assignments/protocol-numbers. For example, if the Virtual Channel Transport is TCP, then the Virtual_Channel_Transport_Value is 0x0000 0006 (equal to decimal 6) or if the Virtual Channel Transport is UDP, then the Virtual_Channel_Transport_Value is 0x0000 0011 (equal to decimal 17).

9.6.1.2 Virtual Channel Priority

The Virtual Channel Priority specifies the priority value that will be used for this Virtual Channel. By default, the Virtual Chanel Priority is set to the Virtual Channel Priorities specified in Table 9-1. The Virtual Channel Parameter is a facility to change the Virtual Channel Priority from its default value for a specific Virtual Channel.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8002		0x0004	
1	Virtual_Channel_Priority_Numeric_Value			

Figure 9-11: Virtual Channel Priority Parameter Representation

Parameter_Type: A 16 bit value set to 0x8002.

Parameter_Length: A 16 bit value set to 0x0004

Virtual_Channel_Priority_Numeric_Value: The Virtual_Channel_Priority_Numeric_Value must take on one of the values shown in Table 9-2.

9.6.1.3 Virtual Channel Bandwidth

The Virtual Channel Bandwidth specifies the bandwidth available for this Virtual Channel.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8003		0x0004	
1	Virtual_Channel_Bandwidth			

Figure 9-12: Virtual Channel Bandwidth Parameter Representation

Parameter_Type: A 16 bit value set to 0x8003.

Parameter_Length: A 16 bit value set to 0x0004

Virtual_Channel_Bandwidth: A 32 bit value that specifies the bandwidth available for the Virtual Channel in kbps

9.6.1.4 Virtual Channel Codec Capability List

The Virtual Channel Codec Capability List provides the set of codecs that this end of the Virtual Channel is capable of using on the Virtual Channel with a representation as given in Figure 9-13. The sending end can only use a codec that is supported by both ends of the Virtual Channel. The Codec_Types for each different Virtual Channel are defined in each Virtual Channel section. If a common Codec Type is not supported by both ends, then no codec is used.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8004		Parameter_Length	
1…	Virtual_Channel_Codec_Capability_List			

Figure 9-13: Virtual Channel Codec Capability List Parameter Representation

Parameter_Type: A 16 bit value set to 0x8004.

Parameter_Length: A 16 bit value set to equal to the sum of all of the lengths of the Virtual Channel codecs given in the Virtual_Channel_Codec_Capability_List.

Virtual_Channel_Codec_Capability_List: The Virtual_Channel_Codec_Capability_List provides a list of the codec capabilities, where each item in the list is expressed in the Virtual Channel Codec Type Parameter Block representation as defined in Section 9.6.1.5.

9.6.1.5 Virtual Channel Codec Type

The Virtual Channel Codec Type specifies a codec for a Virtual Channel with the representation given in Figure 9-14.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8005		Parameter_Length	
1…	Codec_Type			

Figure 9-14: Virtual Channel Codec Type Parameter Representation

Parameter_Type: A 16 bit value set to 0x8005.

Parameter_Length: A 16 bit value set to the length of the Codec_Type variable.

Codec_Type: A variable length parameter that specifies the codec that is to be used on the Virtual Channel expressed in ASCII format. The different Codec_Types are defined with each Virtual Channel definition. The Codec Types for different Virtual Channels are specified in the following sections:

- Net Display - See Section 11.4
- Motion Video – See Section 12.1
- USB – See Section 13.3
- Audio – See Section 16.2.3

9.6.1.6 Virtual Channel Vendor Specific Codec Type

The Virtual Channel Vendor Specific Codec Type specifies a vendor defined codec for a Virtual Channel with the representation given in Figure 9-15. The combination of the Extended Protocol Organization and the Codec Type fully specify a vendor specific codec.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8006		Parameter_Length	
1…	Extended Protocol Organization Parameter Block			
N…	Codec Type Block			

Figure 9-15: Virtual Channel Vendor Specific Codec Parameter Representation

Parameter_Type: A 16 bit value set to 0x8006.

Parameter_Length: a 16 bit value set to the length of the combined sizes of the Extended Protocol Organization Parameter Block and the Codec Type Block.

Extended_Protocol_Organization Parameter Block: The Extended Protocol Organization Parameter Block is defined in Section 9.6.3.1 and provides the URL of the organization that defined this protocol in ASCII format

Codec_Type: A variable length parameter that specifies the codec that is to be used on all the data in the Virtual Channel expressed in ASCII format

9.6.2 Timing Virtual Channel Parameters

The Timing Virtual Channel Parameters are applicable to Virtual Channels that use the optional timestamps in the PDU Header as defined in Section 7.2.4.6. Timestamps are optionally available to provide finer control over the timing of the presentation of related remoted information. The approach to Timestamp transmission and processing is based on that used for the Real-Time Transport Protocol (RTP) as described in RFC 3550.

Timestamps may be used to synchronize audio and video and may also be used to allow audio and video to be played at an even pace and to prevent out of order and late received packets from being played. The Timestamp should be included in all the headers in a Virtual Channel to provide consistency in headers and header processing. For both TCP and UDP, a timestamp field may be useful for the real time playing of video and audio to ensure that material is not replayed late.

Timestamps are optionally included in Motion Video, Audio, Display, Pointer, Keyboard, USB and other Virtual Channel PDUs. Timestamps may be used for a number of different purposes including:

- Reordering of Motion Video and Audio transported over an unreliable transport such as UDP.
- Synchronization of Audio and Video outputs together and with a time base
- Matching Keyboard input with resulting display output

Timestamps may be also be used for other vendor and application specific purposes not defined in Net2Display Remoting.

If Timestamps are to be used in a Virtual Channel, then the Timing Virtual Channel Parameters must be passed during the opening of the Virtual Channel. These parameters provide the timestamp frequency value and may provide an absolute time reference.

The Timing Parameters specify the required parameters for using the optional VC_Timestamp field in a Virtual Channel. When the Timing Parameters are specified in the opening of the Virtual Channel, then the Virtual Channel will include the VC_Timestamp in the data PDUs passed from the end that specified the Timing Parameters. The Timing Virtual Channel Parameters consist of the Virtual Channel Timing Parameter and the Virtual Channel Media Source Parameter.

9.6.2.1 Virtual Channel Timing Parameter Representation

The Virtual Channel Timing Parameter gives the relationship between the relative VC_Timestamp that is placed in Net2Display PDUs and the reference Wallclock_Timestamp used by the source specified in the Media Source Parameter.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8007		0x0010	
1	Wallclock_Timestamp (Most Significant Word)			
2	Wallclock_Timestamp (Least Significant Word)			
3	VC_Timestamp			
4	Timestamp_Frequency			

Figure 9-16: Virtual Channel Timing Parameter Representation

Parameter_Type: A 16 bit value set to 0x8007.

Parameter_Length: A 16 bit value set to 0x0010

Wallclock_Timestamp: The Wallclock_Timestamp is a 64 bit field that indicates the Wallclock Time when the Timing Parameters were sent so that it may be used with VC_Timestamps returned in reception reports from other receivers to measure round-trip propagation to those receivers. The Wallclock time is represented using the full resolution timestamp format of the Network Time Protocol (NTP), which is in seconds relative to 0h UTC on 1 January 1900. The full resolution NTP timestamp is a 64-bit unsigned fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits.

An implementation is not required to run the Network Time Protocol to produce the Wallclock_Timestamp; it may use a local clock or none at all. However, running NTP may be useful for synchronizing streams transmitted from separate hosts. Note that the NTP timestamp will wrap around to zero some time in the year 2036, causing the Wallclock Timestamp to also wrap. However, for Net2Display Remoting purposes only differences between pairs of Wallclock Timestamps are used. So long as the pairs of timestamps can be assumed to be within 68 years of each other, using modular arithmetic for subtractions and comparisons makes the wraparound irrelevant.

On a system that has no notion of a reference Wallclock Time but does have some system-specific clock such as "system uptime", a sender may use that clock as the Wallclock time reference to calculate relative timestamps. It is important to choose a commonly used clock so that if separate implementations are used to produce the individual streams of a multimedia session, all implementations will use the same clock. Until the year 2036, relative and absolute timestamps will differ in the high bit so invalid comparisons will show a large difference; by then one hopes relative timestamps will no longer be used. A sender that has no notion of reference clock or elapsed time may set the Wallclock_Timestamp to zero.

VC_Timestamp: The VC_Timestamp is a 32 bit field that corresponds to the same time as the Wallclock_Timestamp, but in the same units and with the same random offset as the VC_Timestamps in the Virtual Channel data packets. This correspondence may be used for intra- and inter-media synchronization for sources whose Wallclock times are synchronized using NTP. Note that in most cases this VC_Timestamp will not be equal to the VC_Timestamp in any adjacent data packet. Rather, it must be calculated from the corresponding Wallclock Timestamp using the relationship between the RTP timestamp counter and real time as maintained by periodically checking the reference clock time at a sampling instant.

Timestamp_Frequency: The Timestamp_Frequency is a 32 bit field that indicates the number of times the VC_Timestamp is incremented in a second. This field allows for a wide range of increment rates to as low as once per second

9.6.2.2 Media Source Parameter

The Media Source Parameter specifies the source of the media and timing reference being transferred across the Virtual Channel.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8008		CNAME_Length	
1 …	CNAME			

Figure 9-17: Virtual Channel Media Source Parameter Representation

Parameter_Type: A 16 bit value set to 0x8008.

CNAME_Length: A 16 bit value set to the length of the CNAME parameter

CNAME: A parameter that specifies the source of the Virtual Channel data and timing reference. The canonical name is an identifier that uniquely identifies a device that sources a Virtual Channel. This definition of the canonical name is from the RTP protocol as defined in RFC 3550. The CNAME identifier has the following properties:

- The CNAME item must be included to provide the binding to an identifier for the source (sender or receiver) that remains constant.
- The CNAME identifier should also be unique among all participants in the media communication session.
- To provide a binding across multiple media tools used by one participant in a set of related Net2Display Virtual Channels or RTP sessions, the CNAME should be fixed for that participant.
- To facilitate third-party monitoring, the CNAME should be suitable for either a program or a person to locate the source.

Therefore, the CNAME should be derived algorithmically and not entered manually, when possible. To meet these requirements, the following format should be used unless a profile specifies an alternate syntax or semantics. The CNAME item should have the format "user@host", or "host" if a user name is not available as on single-user systems. For both formats, "host" is either the fully qualified domain name of the host from which the real-time data originates, formatted according to the rules specified in RFC 1034, RFC 1035 and Section 2.1 of RFC 1123; or the standard ASCII representation of the host's numeric address on the interface used for the Net2Display communication. For example, the standard ASCII representation of an IP Version 4 address is "dotted decimal", also known as dotted quad, and for IP Version 6, addresses are textually represented as groups of hexadecimal digits separated by colons (with variations as detailed in RFC 3513). Other address types are expected to have ASCII representations that are mutually unique. The fully qualified domain name is more convenient for a human observer and may avoid the need to send a NAME item in addition, but it may be difficult or impossible to obtain reliably in some operating environments. Applications that may be run in such environments should use the ASCII representation of the address instead.

Examples are "doe@sleepy.example.com", "doe@192.0.2.89" or "doe@2201:056D::112E:144A:1E24" for a multi-user system. On a system with no user name, examples would be "sleepy.example.com", "192.0.2.89" or "2201:056D::112E:144A:1E24".

The user name should be in a form that a program such as "finger" or "talk" could use, i.e., it typically is the login name rather than the personal name. The host name is not necessarily identical to the one in the participant's electronic mail address.

This syntax will not provide unique identifiers for each source if an application permits a user to generate multiple sources from one host. Such an application would have to rely on the Virtual Channel Number to

further identify the source, or the profile for that application would have to specify additional syntax for the CNAME identifier.

If each application creates its CNAME independently, the resulting CNAMEs may not be identical as would be required to provide a binding across multiple media tools belonging to one participant in a set of related RTP sessions. If cross-media binding is required, it may be necessary for the CNAME of each tool to be externally configured with the same value by a coordination tool.

Note: Application writers should be aware that private network address assignments may create network addresses that are not globally unique. This would lead to non-unique CNAMEs if hosts with private addresses and no direct IP connectivity to the public Internet have their RTP packets forwarded to the public Internet through an RTP-level translator. To handle this case, applications may provide a means to configure a unique CNAME, but the burden is on the translator to translate CNAMEs from private addresses to public addresses if necessary to keep private addresses from being exposed.

9.6.2.3 Time Stamp Enable Parameter

The Time Stamp Enable Parameter is set to indicate that all data PDUs in the Virtual Channel are to contain a valid Time Stamp in the VC_Timestamp field. The Time Stamp Enable Parameter is used to start and terminate the sending of time stamps on a Virtual Channel. The format of the Time Stamp Enable Paramter is shown in Figure 9-18.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8009		0x0004	
1	Time_Stamp_Enable			

Figure 9-18: Time Stamp Enable Parameter Representation

Parameter_Type: A 16 bit value set to 0x8009.

Parameter_Length: A 16 bit value set to 0x0004, the length of the Time_Stamp_Enable field.

Time_Stamp_Enable: A parameter that specifies the starting and stopping of the sending of Time Stamp in data PDUs. When the Time_Stamp_Enable field is set to 0x0001, all following data PDUs will include a valid value in the VC_Timestamp field. When the Time_Stamp_Enable field is set to 0x0000, the VC-Timestamp field in all following data PDUs will not be considered valid.

9.6.2.4 Timestamp Processing

The VC_Timestamp field, defined in Section 7.2.4.6, indicates the sampling instant of the first data in the Net2Display PDU. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations. The initial value of the VC_Timestamp should be random. The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient). The clock frequency is dependent on the format of data carried as payload and is specified at the opening of the Virtual Channel in the Timing Parameters.

Several consecutive Net2Display PDUs will have equal VC_Timestamps if they are (logically) generated at once, e.g., belong to the same video frame. Consecutive Net2Display packets may contain VC_Timestamps that are not monotonic if the data is not transmitted in the order it was sampled, as in the case of MPEG interpolated video frames. (The sequence numbers of the packets as transmitted will still be monotonic.)

VC_Timestamps from different media streams in different Virtual Channels may advance at different rates and usually have independent, random offsets. Therefore, although these VC_Timestamps are sufficient to reconstruct the timing of a single stream, directly comparing Net2Display VC_Timestamps from different media is not effective for synchronization. Instead, for each medium the Net2Display VC_Timestamp is related to the sampling instant by pairing it with a Timestamp from a reference clock that represents the time

when the data corresponding to the VC_Timestamp was sampled. The reference clock is shared by all media to be synchronized. The Timestamp pairs are not transmitted in every PDU, but at a lower rate in Control PDUs as described in Section 9.6.2.1.

The sampling instant is chosen as the point of reference for the Net2Display VC_Timestamp because it is known to the transmitting endpoint and has a common definition for all media, independent of encoding delays or other processing. The purpose is to allow synchronized presentation of all media sampled at the same time.

Applications transmitting stored data rather than data sampled in real time typically use a virtual presentation timeline derived from reference clock time to determine when the next frame or other unit of each medium in the stored data should be presented. In this case, the Net2Display VC_Timestamp would reflect the presentation time for each unit. That is, the Net2Display VC_Timestamp for each unit would be related to the reference clock time at which the unit becomes current on the virtual presentation timeline. Actual presentation occurs some time later as determined by the receiver.

An example describing live audio narration of prerecorded video illustrates the significance of choosing the sampling instant as the reference point. In this scenario, the video would be presented locally for the narrator to view and would be simultaneously transmitted using Net2Display Remoting. The "sampling instant" of a video frame transmitted in Net2Display would be established by referencing its VC_Timestamp to the reference clock time when that video frame was presented to the narrator. The sampling instant for the audio Net2Display packets containing the narrator's speech would be established by referencing the same reference clock time when the audio was sampled. The audio and video may even be transmitted by different hosts if the reference clocks on the two hosts are synchronized by some means such as NTP. A receiver can then synchronize presentation of the audio and video packets by relating their VC_Timestamps using the Timestamp pairs in Control PDUs.

Note: The initial value of the VC_Timestamp Clock should be random to avoid known text attacks.

9.6.3 Extended Virtual Channel Parameters

The Extended Virtual Channel Parameters may only be present in Extended Virtual Channels, i.e. when the E Bit is set in the PDU Header.

9.6.3.1 Extended Protocol Organization

The Virtual Channel Extended Protocol Organization specifies the URL of the organization that specified the protocol that will be used on the Virtual Channel being opened. This parameter must only be present for Extended Virtual Channels. The format of the Extended Protocol Organization parameter is shown in Figure 9-19.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8010		Parameter_Length	
1…	Extended_Protocol_Organization			

Figure 9-19: Virtual Channel Extended Protocol Organization

Parameter_Type: A 16 bit value set to 0x8010.

Parameter_Length: A 16 bit values set to the length of the Extended_Protocol_Organization in bytes.

Extended_Protocol_Organization: The URL of the organization that defined this protocol in ASCII format

9.6.3.2 Extended Protocol Type

The Virtual Channel Extended Protocol Type specifies the Protocol Type that will be used on the Virtual Channel being opened. This parameter must only be present for Extended Virtual Channels. This parameter follows the Virtual_Channel_Extended_Protocol_Organization and is only interpretable after the Extended_Protocol_Organization has been processed. The format of the Extended Protocol Type parameter is shown in Figure 9-20.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x8011		Parameter_Length	
1…	Extended_Protocol_Type			

Figure 9-20: Extended Protocol Type

Parameter_Type: A 16 bit value set to 0x8011.

Parameter_Length: A 16 bit values set to the length of the Extended_Protocol_Type in bytes.

Extended_Protocol_Organization: The URL of the organization that defined this protocol in ASCII format

9.6.4 General Virtual Channel Parameter Summary

The currently defined General Virtual Channel Parameters are summarized in Table 9-5.

Table 9-5: General Virtual Channel Parameters

Parameter	Class	Parameter Type	Parameter Length	Default Value
Virtual_Channel_Transport_Value	General	0x8001	0x0004	See Table 9-1
Virtual_Channel_Priority	General	0x8002	0x0004	See Table 9-1
Virtual_Channel_Bandwidth	General	0x8003	0x0004	0xFFFF FFFF FFFF FFFF
Virtual_Channel_Codec_Capability_List	General	0x8004	Variable	None
Virtual_Channel_Codec	General	0x8005	Variable	None
Virtual_Channel_Vendor_Specific_Codec	General	0x8006	Variable	None
Timing Parameters	Timing	0x8007	0x0010	None
Media Source Parameter	Timing	0x8008	Variable	None
Time Stamp Enable Parameter	Timing	0x8009	0x0004	None
Extended_Protocol_Organization	Extended	0x8010	Variable	None
Extended_Protocol_Type	Extended	0x8011	Variable	None

10 Aggregated Coordinate Space

An Aggregated Coordinate Space is defined for Net2Display Remoting that is coordinated between the Host and the Client. The Aggregated Coordinate Space defines a space for:

- Locating multiple display areas within the Host display addressing space
- Navigating a cursor between displays for either a Local Cursor or Host Cursor
- Placing Motion Videos within and across displays
- With Aggregated Clients, locating different Clients in the display space and navigating the Cursor between the different Clients and Displays

These different elements are shown in the Aggregated Coordinate Space of Figure 10-1. Figure 10-1 shows the display space for a Host that is has one local display area for viewing material on the Host system and three displays that are remoted to Clients A and B. Clients A and B make up an Aggregated Client, acting in concert to appear to the user as a single Client system. Client A has two displays attached to it and Client B has a single display. Motion Video 1 is mapped to appear across both displays of Client A and Motion Video 2 is being displayed on Client B. The remoting coordinates for these different objects are communicated in the PDUs that communicate these data. The coordinates are expressed using the vertical and horizontal Aggregated Coordinate Space coordinates.

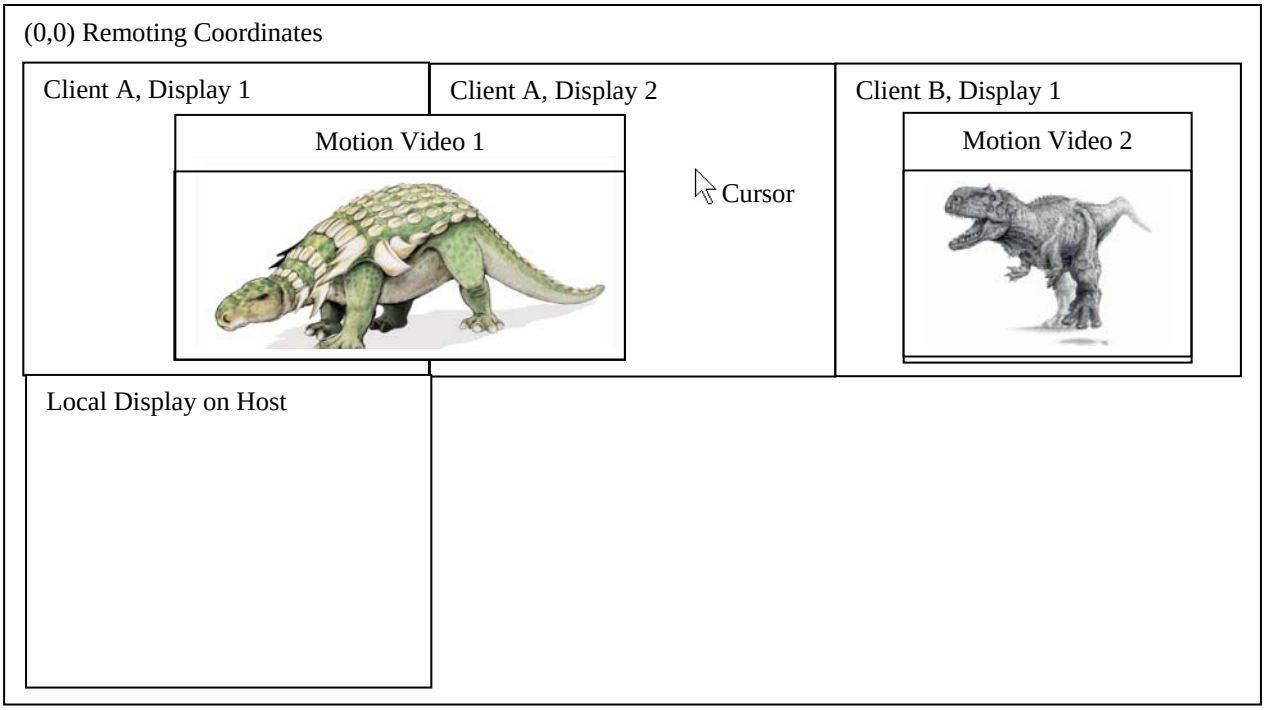


Figure 10-1: Aggregated Coordinate Space

10.1 Display Remoting Elements

The Aggregated Coordinate Space of Figure 10-1 illustrates the mapping of the different remote elements at a high level. For the actual mapping to take place, Net2Display has defined a set of Display Remoting elements to characterize the display mappings that are used. The Display Remoting elements used in remoting the display space are defined and illustrated in this section. The different terminology used for representing the scenario of Figure 10-1 is shown in Figure 10-2.

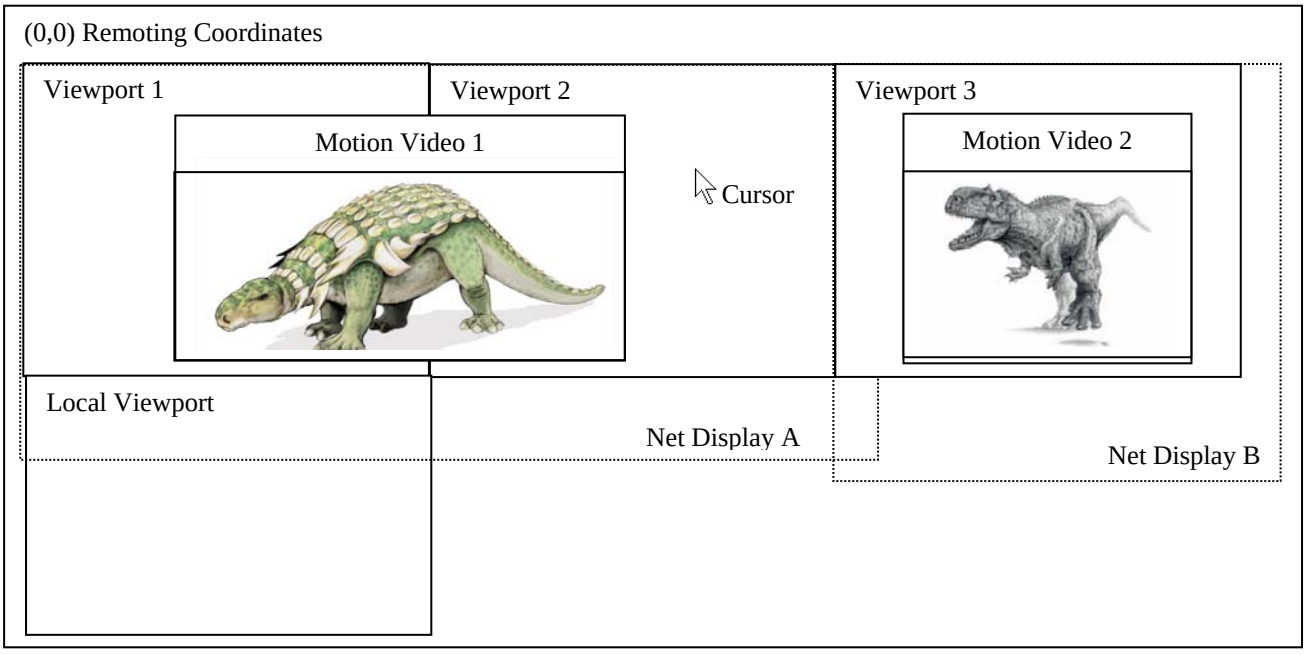


Figure 10-2: Viewports in Aggregated Coordinate Space

10.1.1 Net Displays

A Net Display is a display surface remoted from a Host to a Client. A Net Display space includes both visible and invisible regions, similar to the mapping for video RAM, which includes both visible and invisible regions. While the visible regions are mapped to Client display areas, the invisible regions are typically used for offscreen drawing, caching offscreen display content and composing display material before moving to the visible area. The invisible regions may be used at the discretion of the Host direction.

A Net Display region may contain multiple Viewports that represent multiple Client displays as illustrated in Figure 10-3. A Net Display is sent across a single Virtual Channel and goes from a single Host to a single Client. Thus, the number of Net Display Virtual Channels is equal to the number of remoted Net Displays. If a Host is connected to multiple Clients acting as an Aggregated Client, one or more Net Displays surfaces must be remoted across Net Display Virtual Channels to each individual Client. The size of the Net Display area is coordinated between the Host and the Client, with the Client informing the Host of the area that is available for remoting and the Host determining if it will use offscreen drawing. Net Displays to two different Clients may overlap and may be used for providing overlapping Viewports.

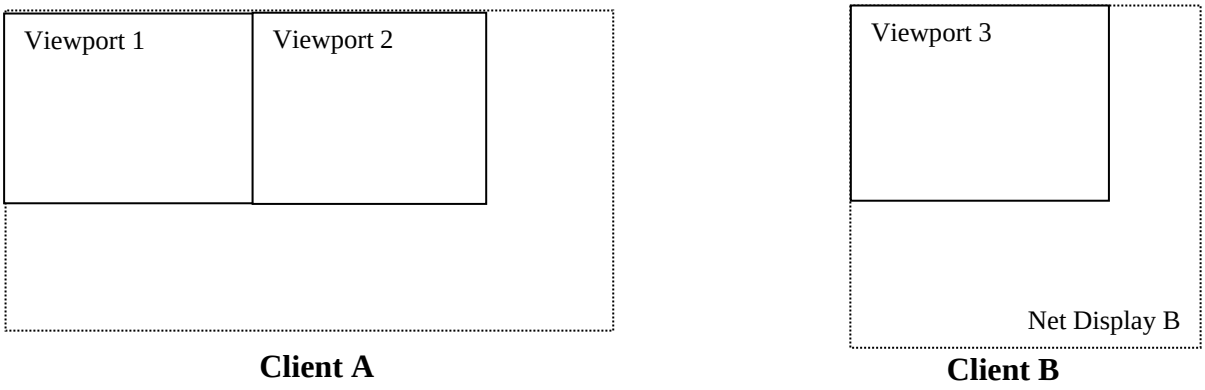


Figure 10-3: Net Display Areas

10.1.2 Viewports

A Viewport is a visible area within a Net Display area and generally corresponds to a single Client Display. A Viewport may be placed at any location within the Aggregated Coordinate Space. Multiple Viewports may be mapped within the same Net Display area. Placing two Viewports of the same size at the same location in the Aggregated Coordinate Space results in the same display information being displayed on the displays that is mapped to the overlapping Viewport areas. Viewports may completely overlap or may partially overlap. The mapping of Viewports to represent the display space of Figure 10-1 is shown in Figure 10-2. Figure 10-4 shows the groupings of these Viewports by Net2Display Client.

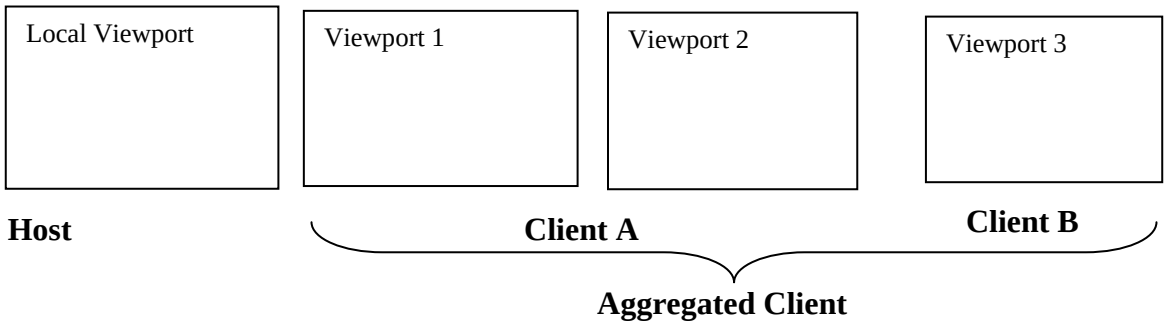


Figure 10-4: Viewports Grouped by Location

10.1.3 Motion Videos

A Motion Video is a stream for placing a motion video within a Net Display region. The Motion Video may appear on multiple Viewports, but all of the Motion Video must be within one Net Display region. The Motion Video is essentially a drawing primitive to a Net Display surface and must stay within the one Net Display Surface. The Motion Video is typically decoded in a separate space and placed within the visible Viewport. The mapping of Motion Videos to the Aggregated Coordinate Space is shown in Figure 10-2.

10.1.4 Multiple Display Clients

A Client with multiple displays may be handled in multiple different ways in the mapping of Net Displays to physical displays: one Net Display may be used to cover all the display surfaces of the Displays in the Client or multiple Net Display surfaces may be used to cover the different displays in the Client either in a one to one relationship or with multiple displays comprising one or more Net Displays. A representation of a Client with multiple displays is shown in Figure 10-3.

10.1.5 Aggregated Client

An Aggregated Client is when multiple Clients are grouped together and operate for the user as one Client as illustrated in Figure 10-4. Since a Net Display can only remote to one Client device, each Client in an Aggregated Client must be mapped to a different Net Display.

10.1.6 Display Location Addressing

All display location addressing is done with signed integers. This becomes important in global addressing where additional displays need to be added above and to the left of existing displays.

10.2 Net Display Remoting Examples

Since there are a range of different Client instantiations and configurations, there are a number of different scenarios for remoting displays to Clients. Since one Net Display only remotes to one Client, only the single Client configurations will be enumerated here. These configurations remoting a Net Display to a single Client include:

- Remoting to a single connected display
- Remoting to a windowed area on a single display
- Remoting to the full area on multiple displays
- Remoting to a windowed area on multiple displays

The configurations are detailed in the following sections.

10.2.1 Remoting to a Single Connected Display

A single Net Display can also be used to remote a single display that supports offscreen drawing as shown in Figure 10-5. For such a configuration, the Viewport matches the Client display in size and is contained within the Net Display area, which includes both visible and invisible regions.

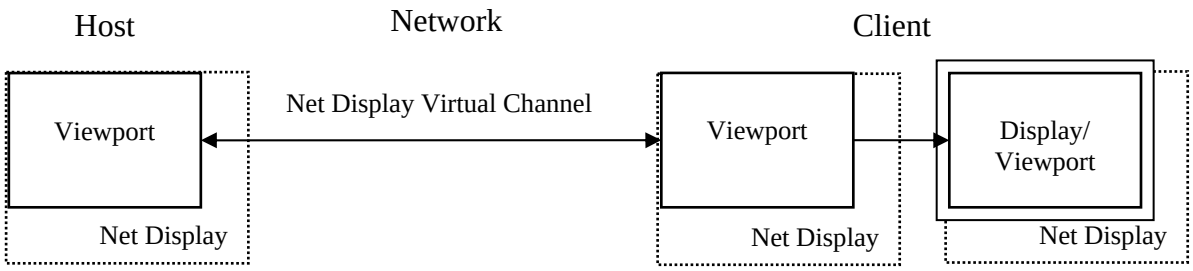


Figure 10-5: Entire Display Remoted as a Single Net Display with Offscreen Area

A special case of this configuration is the simplest configuration where the Client has a single display that is remoted without allowing offscreen drawing. That configuration is shown in Figure 10-6. For such a configuration, the Display, Viewport and Net Display all map to the same area.

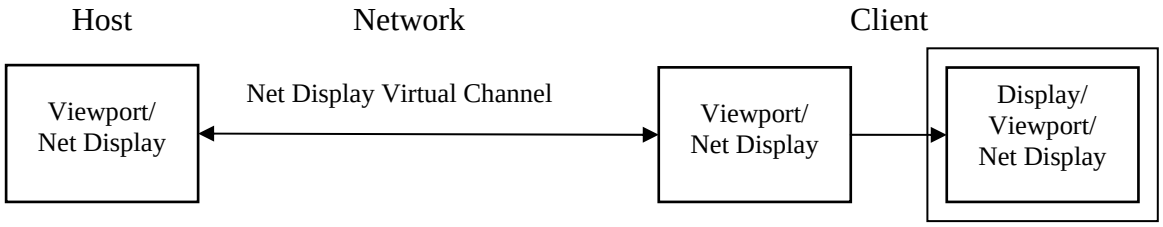


Figure 10-6: Entire Display Remoted as a Single Net Display with no Offscreen Area

10.2.2 Remoting to a Windowed area on a Single Display

A single Net Display can be used to remote a window in a display as shown in Figure 10-7. Figure 10-7 has a larger Net Display region that includes offscreen drawing.

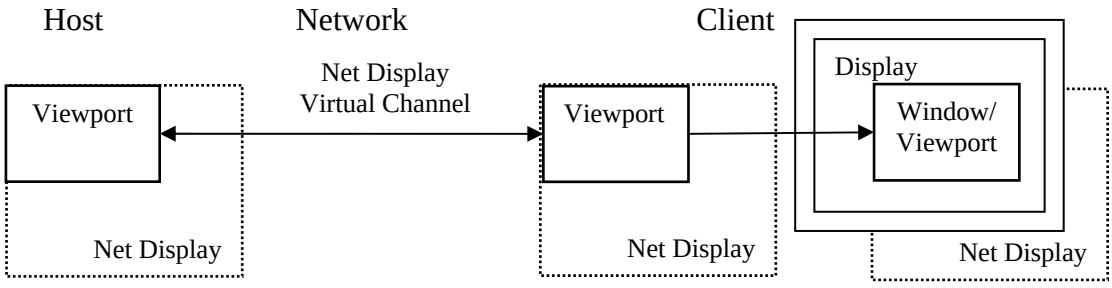


Figure 10-7: Client Display Window Remoted as a Single Net Display with Offscreen Area

A single Net Display can be used to remote to a window in a Client display without an offscreen area as shown in Figure 10-8. If no offscreen drawing, then the display window, Viewport and Net Display all map to the same region.

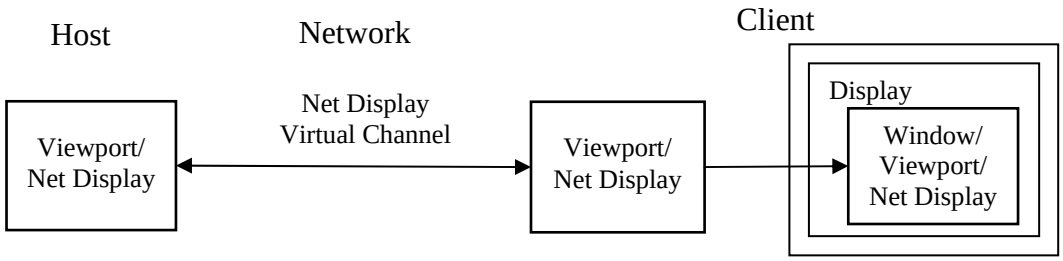


Figure 10-8: Client Display Window Remoted as a Single Net Display with no Offscreen Area

10.2.3 Remoting to the full area on Multiple Displays

A single Net Display can be used to remote an area composed of multiple displays as shown in Figure 10-9. The offscreen area extends outside of the area of the visual displays.

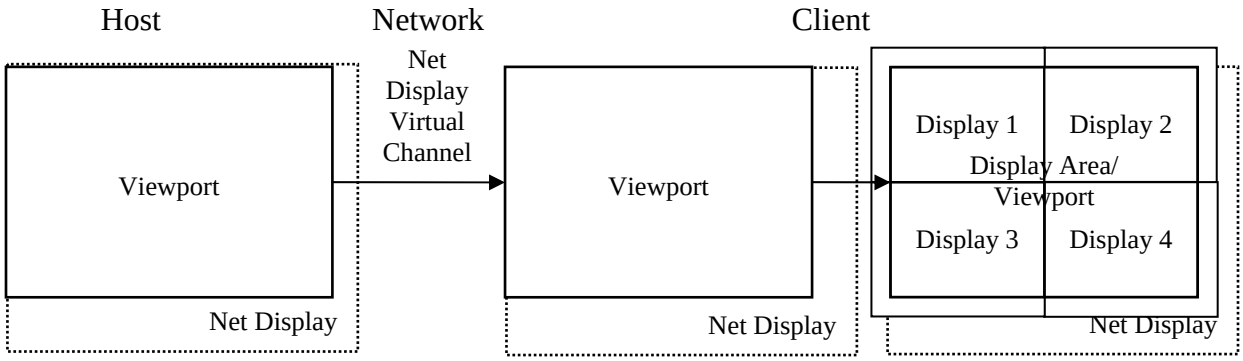


Figure 10-9: Multiple Displays Remoted as a Single Net Display with Offscreen Area

A single Net Display can be used to remote an area composed of multiple displays as shown in Figure 10-10 without an offscreen area.

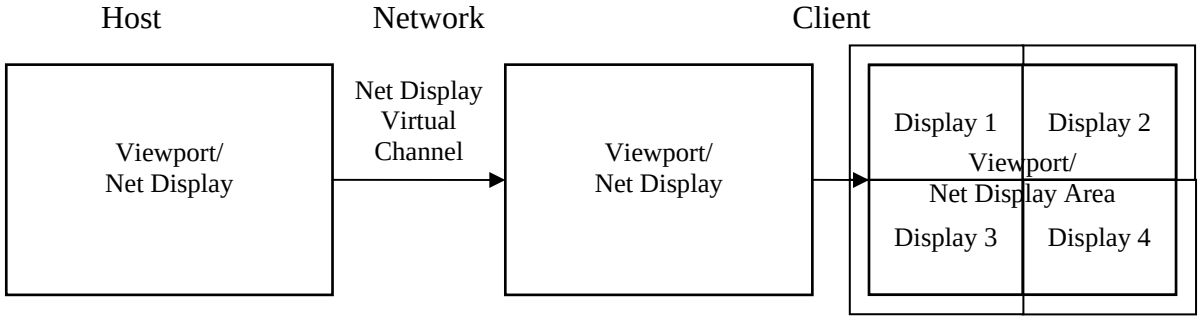


Figure 10-10: Multiple Displays Remoted as a Single Net Display with no Offscreen Area

10.2.4 Remoting to a Windowed Area on Multiple Displays with no Offscreen Writes Allowed

A single Net Display can be used to remote a window area that spans across multiple displays and provides an offscreen area as shown Figure 10-11.

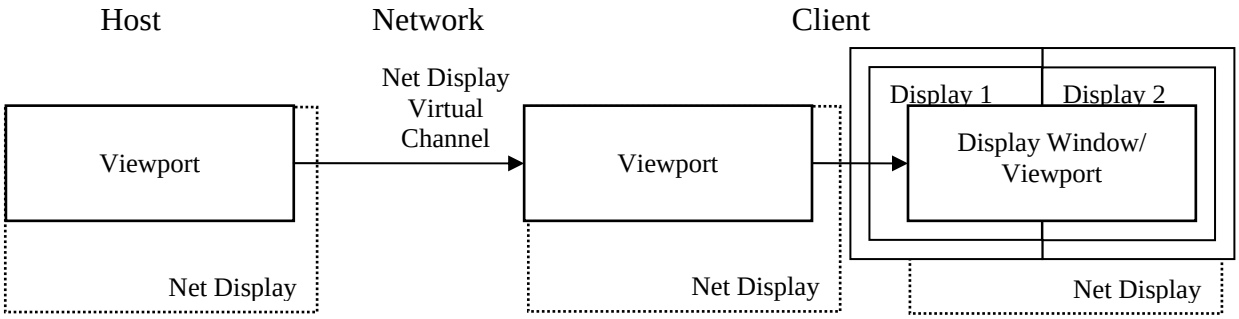


Figure 10-11: One Net Display Remoting Window Spanning Multiple Displays with Offscreen Area

A single Net Display can be used to remote an area that spans across multiple displays without an offscreen region as shown Figure 10-12.

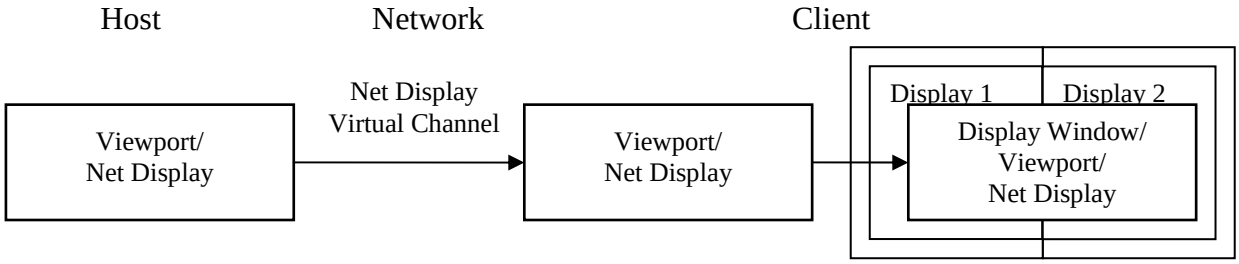


Figure 10-12: One Net Display Remoting Window Spanning Multiple Displays with no Offscreen Area

10.3 Display Colorimetry

The Display Colorimetry is expressed separately for each remoted display and each motion video.

10.3.1 Host Display Colorimetry

Net2Display Host devices must support sourcing of both RGB and YCbCr Colorimetry formats as shown in Table 10-1. A Net2Display Host device must indicate the Colorimetry format (including the dynamic range) of the transmitted stream in the Net2Display Display Parameters.

In determining the Colorimetry format, the Net2Display Host device must check the capability of the Net2Display Client device via the EDID passed from the Client in the Viewport EDID parameter as defined in Section 11.6.3.4. When the Net2Display Client device capability is unknown, for example due to the corruption of EDID, the Net2Display Host device must fall back to 18 bpp RGB, with full dynamic range.

When a Net2Display Host device is transmitting a RGB stream with a video timing format called out in CEA-861C Section 5 (except 640 x 480p) as using CEA range RGB, it should use CEA range RGB.

When a Net2Display Host device is transmitting 640 x 480p 24 bit RGB, it will always use the full dynamic range.

Table 10-1: Net2Display Colorimetry Format Support

Colorimetry Format	Bit-depth per Pixel (bpp)	Bit-depth per Component (bpc)	Dynamic Range, Coefficients	Mandatory vs. Optional
RGB	18	6	"VESA range" only	Mandatory. Used in "fall-back" modes when Net2Display Client device capability are unknown.
	24	8	"VESA range" or "CEA range"	Mandatory
	30	10		Optional
	36	12		
	48	16		
YCbCr 4:2:2	16	8	"CEA range". For CEA range, either 601 or 709 coefficients	Mandatory if YCbCr is supported on any other display interface
	20	10		Optional
	24	12		
	32	16		
YCbCr 4:4:4	24	8		Mandatory if YCbCr is supported on any other display interface
	30	10		Optional
	36	12		
	48	16		

Note: See the following sub-sections for definitions of VESA range and CEA range.

10.3.1.1 RGB Colorimetry

All Net2Display Host devices must support RGB Colorimetry with pixel depths of 18 and 24 bpp. Support for 30, 36 and 48 bpp RGB is optional.

"VESA range" and "CEA range" are defined as follows:

- "VESA range" must have:
  - Nominal zero intensity level at code value zero
  - Maximum intensity level at maximum code value allowed for bit depth. i.e. 63 for 18 bpp RGB, 255 for 24 bpp RGB, 1023 for 30 bpp RGB, 4095 for 36 bpp RGB, and 65,535 for 48 bpp RGB.
- "CEA range" must have:
  - Nominal zero intensity level at 16 for 24 bpp, 64 for 30 bpp, 256 for 36 bpp, and 4,096 for 48 bpp.
  - Maximum intensity level at maximum code value allowed for bit depth, namely, 235 for 24 bpp RGB, 940 for 30 bpp RGB, 3760 for 36 bpp RGB, and 60160 for 48 bpp RGB.

    Note: The RGB CEA range is defined for 24, 30, 36, 48 bpp RGB only, not for 18-bpp RGB.

When a Host device is transmitting a RGB stream with a video timing format called out in CEA-861C Section 5 as using CEA range RGB, it must use the CEA range RGB.

However, a Net2Display Host device may transmit all code values from zero to the maximum even when it declares the CEA range in the Main Stream Attributes. It is the responsibility of the Net2Display Client device to limit the pixel value range as needed.

Note: The Net2Display Host device falls back to 18 bpp, VESA range RGB when the sink capability is unknown.

10.3.1.2 YCbCr Colorimetry

Support for YCbCr Colorimetry is required for Host devices that support YCbCr or YPbPr on any other display interface, except where the sourcing Host would be required to convert RGB video to YCbCr in order to meet this requirement.

Source Hosts that support YCbCr must support at least 24 bpp YCbCr 4:4:4 and 16 bpp YCbCr 4:2:2 in both 601 (defined in ITU-R BT.601-5 section 3.5 or EIA/CEA-770.2-C section 3.3) and 709 (defined by ITU-R BT.709-4 Part 1, Section 4 or EIA/CEA-770.3-C Sections 5.4 to 5.7).

In addition to the required minimum above, the pixel depth may optionally be 30, 36 and 48 bpp for YCbCr 4:4:4 and 20, 24 and 32 bpp for YCbCr 4:2:2.

YCbCr dynamic range is recommended to be as defined in CEA-861C Section 5 (CEA range):

- Y has nominal zero intensity level at 16 for 8 bits, 64 for 10 bits, 256 for 12 bits, 4,096 for 16 bits per component
- Y has nominal maximum intensity level at 235 for 8 bits, 940 for 10 bits, 3760 for 12 bits, and 60160 for 16 bits per component
- Cb and Cr have their zero levels at 128 for 8 bit, 512 for 10 bit, 2048 for 12 bits, and 32,768 for 16 bits per component
- Cb and Cr have nominal ranges of 16 to 240 for 8 bits, 64 to 960 for 10 bits, 256 to 3840 for 12 bits, and 4,096 to 61,440 for 16 bits per component.

However, a source Host may transmit all code values from zero to the maximum value. It is the responsibility of the Client device to limit the pixel value range as needed.

10.3.2 Client Display Colorimetry

Net2Display Client devices support sinking of both RGB and YCbCr Colorimetry formats as shown in Table 10-1. Client devices must read the Colorimetry format of the transmitted stream from the DisplayPort Main Stream Attributes.

When receiving a CEA range video stream, the Net2Display Client device should anticipate that all the code values may be used by the Net2Display Host device and clamp the dynamic range if needed.

11 Display Remoting Process

Display Remoting passes static display images and video sequences across the Net2Display connection. Display Remoting content is passed from the Host to the Client and control information is passed back from the Client to the Host. Display Remoting supports a number of different performances of implementations by specifying a range of Display Remoting primitives from passing simple pixel bitmaps to using different forms of compression. A high level model of the Display Remoting is shown in Figure 11-1.

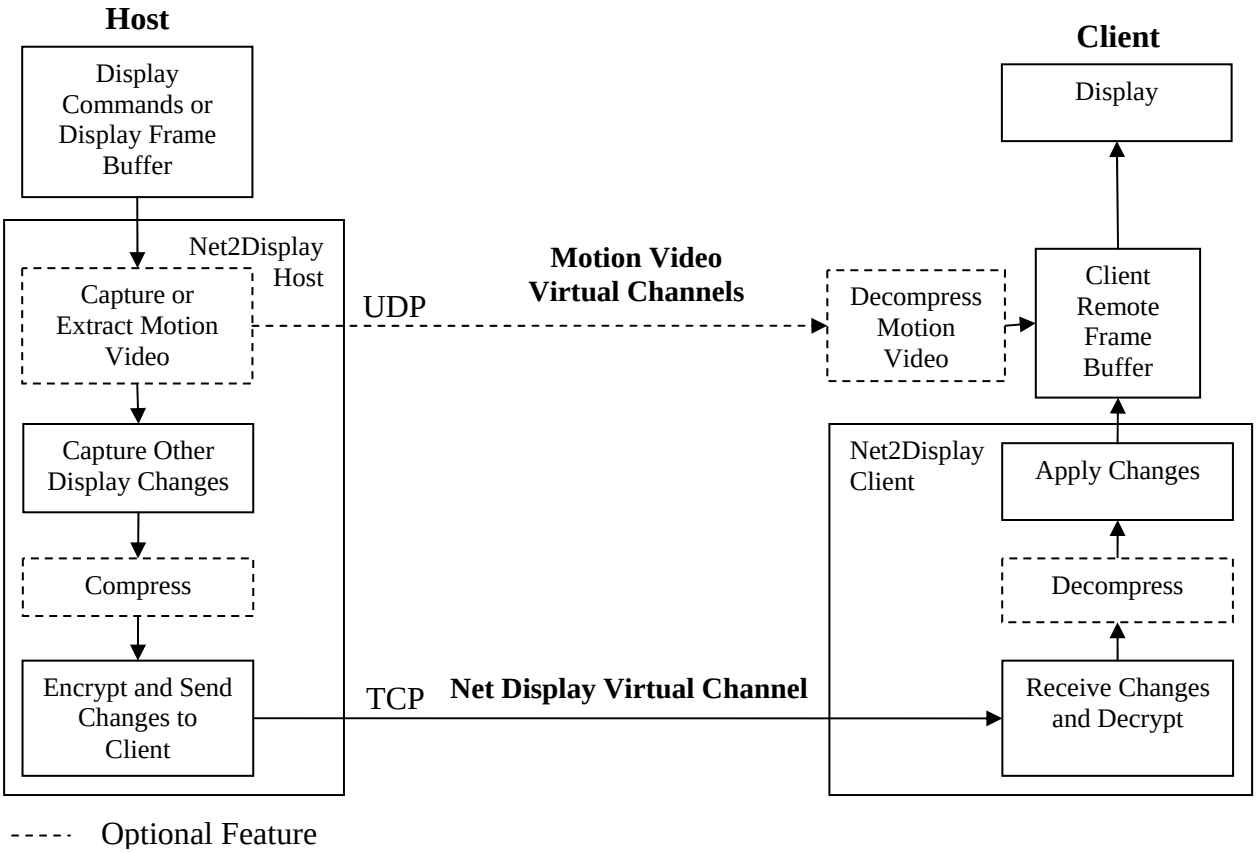


Figure 11-1: High Level Display Remoting Model

Displays are remoted by:

1) Setting up a Virtual Channel to pass the remoting information from the Host to the Client
2) Capturing the display contents or changes in contents on a Host Display Interface,
3) Interpreting the information at that interface to represent the updates to the display contents with a set of Display Remoting Primitives,
4) Transmitting the Display Remoting Primitives across the network to the Client on a Virtual Channel,
5) Receiving the Display Remoting Primitives at the Client and interpreting and using them to update the display.

Each of these steps in the remoting is described in within this section.

11.1 Host Remoting Model

In specifying interoperable Client Display Remoting, Net2Display Remoting specifies the commands and data that pass across the network or interconnection and the interpretation of these commands at the Client. Such a level of specification is sufficient for interoperability. This section provides guidance in selecting a Host interface for remoting to encourage consistent and efficient implementations for different significant Host configurations and operating systems.

In a Host computer system with a graphics processor and a display output, there are a number of different interfaces where Net2Display Remoting may acquire the graphics information for Display Remoting, as shown in Figure 11-2. All of these interfaces are shown to illustrate the full scope of the interfaces that could be used for remoting; not all of these interfaces will exist in all systems. Remoting can also be accomplished using a single one of these interfaces or by using the information from multiple interfaces.

Potential interfaces for remoting include the 2D graphics interface, designated as the Graphics Device Interface, found in computers using Microsoft® Windows® operating systems released before the Windows Vista operating system. The 3D graphical interfaces, OpenGL application programming interface and the Direct3D application programming interface are also potential interfaces for remoting. These interfaces capture the graphical functions that are sent to the Graphics Driver.

Remoting could also be performed after processing by the Host Graphics processor at several different locations. Changes to the Framebuffer could be tracked and remoted using Buffer Update Remoting. The Framebuffer itself may be monitored for changes and the changes captured and send as designated by Framebuffer Remoting. Additionally, the output of the Graphics Processor that would normally be sent to a local display may be captured, processed and remoted at the location designated as the Display Signal Remoting Interface. There may be a display connector where the signals are captured or the signals may exist on an internal Host interface. Finally, an application interface could be used in the Host that allows the capture of video content or compressed video streams from within applications, as designated by the Application Interface.

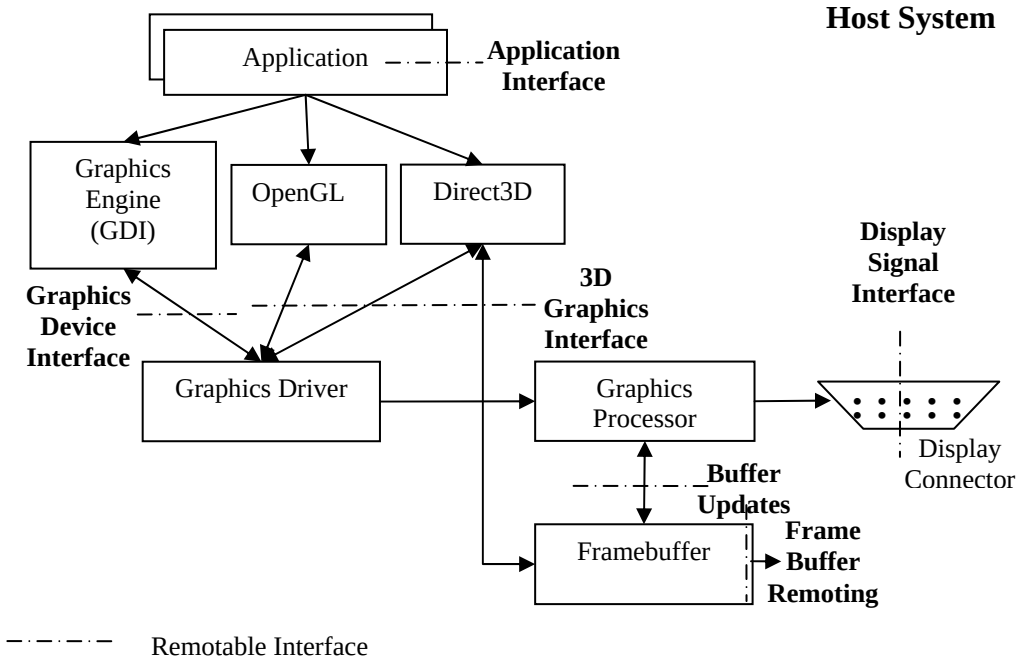


Figure 11-2: Host System Interfaces Available for Graphics Remoting

The different interfaces available for remoting are described in this section. Different operating system configurations and considerations are also described. The configurations and operating systems considered in this section are: Graphics Device Interface (GDI) Remoting, 3D Graphics Remoting, Unix Remoting, Framebuffer Remoting, and Display Signal Remoting. Other interfaces for remoting Net2Display may exist that are not described in this section, but that does not preclude those interfaces from being used for Net2Display Display Remoting.

11.1.1 Graphics Device Interface

Graphics Device Interface (GDI) is the main graphics library used for Windows®-based applications at the writing of this standard. GDI provides 2D graphics and text functionality, with limited imaging functionality. GDI acts as an intermediary between a Windows®-based application and the graphics driver as shown in Figure 11-3. GDI+ was introduced with the Windows® XP operating system and added anti-aliased 2D graphics, gradients, and per-pixel alpha value support. The Windows®-based applications call Win32® GDI functions with graphics output requests, which are routed to the kernel-mode GDI, a system-supplied module. The kernel-mode GDI then sends the requests on to the proper graphics driver.

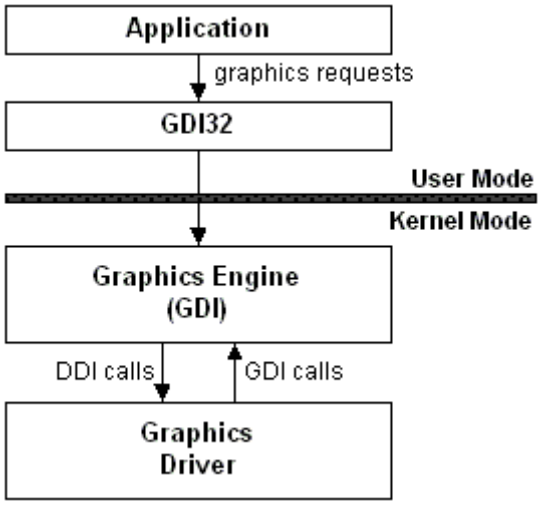


Figure 11-3: GDI and Graphics Driver

GDI communicates with the graphics driver through the graphics device driver interface (DDI) functions. The graphics DDI functions are identified by a *Drv* prefix. The graphics driver must support certain *DrvXxx* functions for the GDI to call, but is not required to support all of the DDI functions, allowing simplification of graphics drivers. GDI supports many of these graphics functions itself, which are designated with an *Eng* prefix. When a graphics driver is unable to support a graphics call, it calls back to the GDI with the *EngXxx* function. The *DrvXxx* functions have a corresponding *EngXxx* function with the same arguments that is called when the graphics driver does not support a function.

The graphics DDI functions can be divided into those that are required, conditionally required and optional. The required functions are used to enable and disable drivers, data structures and drawing surfaces. Three conditionally required functions are needed when the graphics device is managing the surface:

- DrvCopyBits
- DrvTextOut
- DrvStrokePath.

The *DrvCopyBits* function is not used to perform a copy operation on a display surface, but is used to translate between the GDI standard format bitmaps and graphics device managed surfaces. *DrvTextOut* is used to render a set of character images at specified positions and *DrvStrokePath* is used to render a line on a display surface. Optional DDI functions include additional drawing functions such as *DrvBitBlt*, for bit block transfers with raster operations (ROP), and *DrvAlphaBlend*, for bit block transfers with alpha blending. Supporting additional conditionally required and optional DDI functions in the graphics driver can provide higher performance.

Implementations of remoting Windows®-based operating system computers have also found that if certain other DDI functions are not implemented in the graphics driver, but are returned to the GDI, the screen image will not be complete.[1] These additional DDI functions that must be implemented in the graphics driver are:

- DrvAlphaBlend
- DrvGradientFill
- DrvStretchBlt
- DrvLineTo
- DrvTransparentBlt

11.1.1.1 GDI Remoting with a Net2Display Driver

When a Net2Display Driver is present on the Host computer, it needs to support the DDI functions, just like any other graphics driver. With this approach, the Net2Display Driver typically supports a display surface locally in the Host computer system. The Host, then, always maintains the current state of the display, keeping the Client stateless, supporting bit block transfer operations on the Host end, and allowing for rapid recovery. The DDI calls to the Net2Display Driver are translated into a set of Net2Display Remoting Commands that are sent across the network to the Net2Display Client. The Net2Display Display Remoting Commands are a subset of the operations available on the DDI interface.

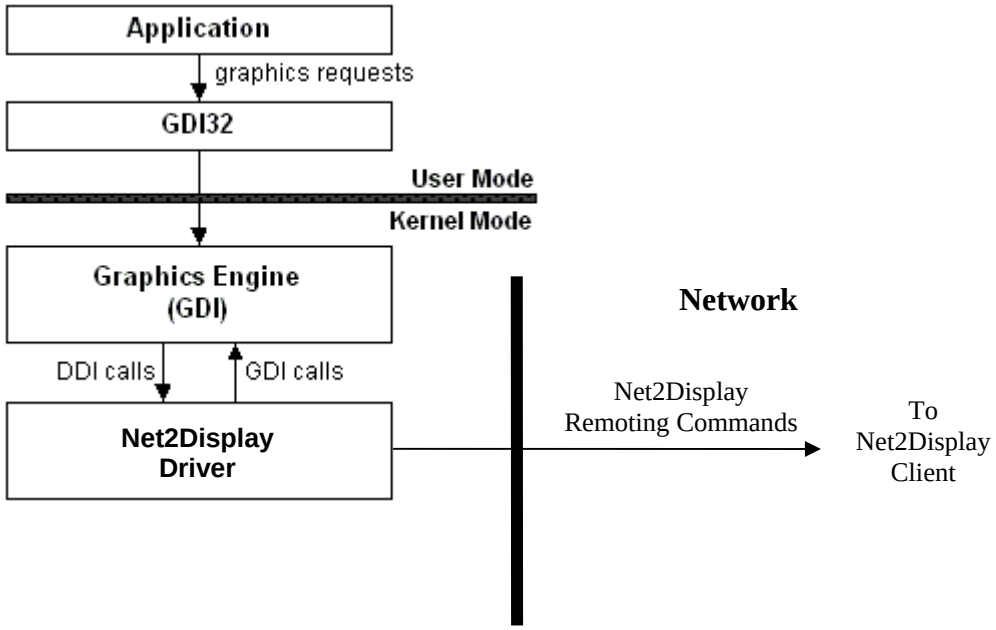


Figure 11-4: GDI and Net2Display Driver

[1] Lei Zheng, "Implementing Remote Display on Commodity Operating Systems," M.S. Thesis, Columbia University, January 2006, http://ncl.cs.columbia.edu/publications/zhangthesis.pdf

11.1.2 Windows Vista Operating System Interfaces

In the Windows Vista operating system, GDI has been replaced by the Windows Presentation Foundation (WDF), which then performs all of its rendering through a 3D interface, either using the Direct3D application programming interface or the OpenGL application programming interface. The GDI/GDI+ interfaces will still be supported to provide support for current hardware and application compatibility. The 3D graphics interface will be the primary graphics interface for most drawing functions. The support for 3D graphics as is used in the Windows Vista operating system is described in Section 11.1.3.2.

11.1.3 3D Graphics Remoting Interfaces

High performance 3D graphics is typically provided using either the OpenGL application programming interface or the Direct3D application programming interface. The OpenGL application programming interface is used by Windows, Linux and MAC OS, while the Direct3D application programming interface is available only for the Windows Operating System. At the time of writing, the OpenGL application programming interface is used predominantly for commercial 3D applications, while the Direct3D application programming interface is prevalent in 3D Gaming

11.1.3.1 OpenGL Application Programming Interface Remoting

The remoting of OpenGL application programming interface function calls across Net2Display Remoting will not be directly supported for the following reasons:

- OpenGL application programming interface consists of over 250 different function calls, which is too many to be supported in a remoting interface
- OpenGL application programming interface maintains a considerable amount of state information, which would require significant state to be maintained at the Client
- OpenGL application programming interface is being updated on a regular basis. This makes it difficult to track a mapping from Net2Display to the OpenGL application programming interface functions, requires upgrades to the Client system to support updates to OpenGL application programming interface and shortens the expected life of the Client system.

Thus, for remoting content generated using the OpenGL application programming interface, it is recommended that the Host create the OpenGL application programming interface content in a Framebuffer and use a suitable compression algorithm to remote changes to the Framebuffer to the Client. This allows a graphics processor located in the Host to perform the OpenGL application programming interface functions and the updates to be rapidly reflected on the Client.

11.1.3.2 Direct3D Application Programming Interface Remoting

Likewise, the remoting of Direct3D application programming interface function calls across Net2Display Remoting will also not be directly supported for the following reasons:

- Direct3D application programming interface consists of a large number of different function calls, which is too many to be supported in a remoting interface
- Direct3D application programming interface maintains a considerable amount of state information, which would require significant state to be maintained at the Client
- Direct3D application programming interface is being updated on a regular basis. This makes it difficult to track a mapping from Net2Display to the Direct3D application programming interface functions, requires upgrades to the Client system to support updates to theDirect3D application programming interface and shortens the expected life of the Client system.

Thus, for remoting content generated using the Direct3D application programming interface, it is recommended that the Host create the Direct3D application programming interface content in a Framebuffer

and use a suitable compression algorithm to remote changes to the Framebuffer to the Client. This allows a graphics processor located in the Host to perform the Direct3D application programming interface functions and the updates to be rapidly reflected on the Client.

11.1.4 X Windows System® Remoting

X Window System remoting is accomplished by converting the X remoting primitives into Net2Display remoting commands.

11.1.5 Display Signal Remoting

Net2Display Remoting can be accomplished by monitoring the output of a display connector from the Host and determining and remoting the areas of change to the Display. This display connector could be a VGA connector, DVI connector, other display connector or internal interface. A physical display connector is not necessary for the Display Signal Remoting approach to be used. For this configuration, the Net2Display Remoting may:

1. Monitor all of the traffic across the display signal interface
2. Optionally determine which regions of the display surface have changed and remote only the changes
3. Optionally compress the changed regions and remote to the Client

The display signal capture of Display Signal Remoting is effectively one method to acquire Framebuffer data. As a result, Display Signal Remoting is detailed within the Framebuffer Remoting Section.

11.1.6 Framebuffer Remoting

Net2Display Remoting can be accomplished by providing a complete graphics system in the Host that creates the image to be remoted in a Host-located Framebuffer. Software or hardware in the Host monitors the Framebuffer for changes and remotes the changes to the Client optionally using compression. The sensing of the changes to the Framebuffer may be done at periodic intervals to lessen the Display Remoting network traffic, while slightly increasing the Client response time. This section describes some of the methods that can be used to capture and transfer a display image rendered into a Framebuffer.

11.1.6.1 Obtaining a Framebuffer Image

There are a number of methods to obtain a Framebuffer image. The method for obtaining the Framebuffer should not affect the compatibility of the resulting system and vendors may develop proprietary access methods. Each method for obtaining the Framebuffer image has its advantages and disadvantages. The four different methods considered are:

- Framebuffer Direct Memory Access
- Framebuffer Copy Command
- Framebuffer Rendering to Accessible Memory
- Framebuffer Raster Capture (Display Signal Remoting)

11.1.6.1.1 Framebuffer Direct Memory Access

For some hardware and operating systems the structure and physical location of the Framebuffer can be determined and accessed. This may require locking the Framebuffer and accessing it from the application space or the Framebuffer may be at a fixed location in hardware.

11.1.6.1.2 Framebuffer Copy Command

If the Framebuffer physical location cannot be determined directly, the operating system may provide an API for copying the contents of the Framebuffer. These commands can provide application protection and physical address translation for paged Framebuffer memory. Once copied out of the Framebuffer, the data can be encoded and transmitted

11.1.6.1.3 Framebuffer Rendering to Accessible Memory

If the Framebuffer normally rendered by the operating system is unavailable or the performance of the access is too slow, an alternative is to render the image into system memory at a known and accessible location. This can be done by creating a duplicate image using a mirror driver, or replacing the existing graphics driver with a graphics driver that renders the image at an accessible location. Note that if the image is to be remoted and a new image is being created, there is no need for the originally rendered image.

11.1.6.1.4 Framebuffer Raster Capture (Display Signal Remoting)

Raster capture is a method of obtaining a copy of the Framebuffer by capturing Framebuffer as it is transferred from the Framebuffer to the display. Raster capture requires additional hardware to capture the image as it is being sent to what the host considers a local display. Raster capture removes compatibility and performance issues at the expense of additional hardware.

11.1.6.2 Issues Obtaining a Framebuffer Image

11.1.6.2.1 Multiple Framebuffers and Display Controllers

In some environments, the display system assembles the final display image as it is being sent to the display, so a single stored Framebuffer may not represent the full image seen at the desktop. One example of this is the pointer overlay using alpha blending. It is important that the system account for these limitations by assembling a final Framebuffer at the host or presenting the same final assembly and the client.

11.1.6.2.2 Disruption of the Graphics Stack

If the Framebuffer is to be rendered locally, there maybe limitations on the graphics stack that modify the image. For example, to use the DirectX® application programming interface require hardware shaders so if the system is to render the image in software, the system must remove the requirement for DirectX application programming interface support, thereby limiting the image content.

11.1.6.3 Determining Updated Framebuffer Pixels

Because the performance of a Framebuffer's remoting system is dependent on the Framebuffer access bandwidth, the access to the Framebuffer should be minimized to only access what has been modified. This requires the system to determine which pixels have been updated.

11.1.6.3.1 Rendering Command Trapping

One method to determine which pixels of the Framebuffer have been modified is to trap the rendering commands, using a means like a mirror driver, and keep track of the areas in the Framebuffer being update. This allow the system only access the Framebuffer when it is modified and only the area of the Framebuffer that has been modified

11.1.6.3.2 Framebuffer Sub-sampling

An alternative method for determining Framebuffer updates is to sample a limited number of pixels of the Framebuffer. Since changes usually happen in large areas, one can quickly determine a fair guess of the area that needs to be copied to the encoder. The method should rotate through a sub-sampling structure that eventually reviews all pixels on the screen to ensure no updates are missed.

11.1.6.4 Encoding and Transmitting a Framebuffer Image

After accessing the Framebuffer and determining which pixel need to be transmitted to the client, the pixel information needs to be encoded and transmitted. As the data is visual there are a number of items that should be considered for selecting the encoding method.

11.1.6.4.1 Two Dimensional Data - Blocks

Display represents a 2D space which has data correlation in both dimensions. Taking this 2D correlation into account improves the compression of the data. This is why image content is typically encoded a defined area at a time. For simplification the area is normally a grid of pixels on a power of 2 spacing, like 8x8 or 16x16 blocks.

11.1.6.4.2 Block Sequencing

Block sequencing is the order in which defined areas of the display image are processed. Three items make block sequencing important. One, the 2D correlation of image data does not end at the block boundary. Processing blocks adjacent to each other at the same time provides opportunities for improved correlation and improved compression. Two, memory and data transfer structures can look ahead and effectively manage the transfer of image content more quickly. And three, a predefined block sequence means the block address does not need to be encoded for each block.

11.1.6.4.3 Encoding Methods

As the image content that is to be encoded is in block pixel format, the encoding methods are primarily bitblts with some sort of lossless or lossy compression. Selecting the sequencing encoding methods as well as the lossless and lossy encoding methods is for another section of this document.

11.1.6.5 Framebuffer Remoting Summary

Framebuffer remoting can be implemented on the host using any one of a number of methods. Since the result is to encode areas of the Framebuffer, the Net2Display protocol can be independent of the Framebuffer acquisition method. The focus needs to be on how to define the area of the screen that has been encoded and which encoding method to use.

11.2 Client Display Model

Net2Display Remoting provides Display Remoting to match several different display models: Display, Terminal, and Windowed. These models are:

- Display - The Client behaves as a raw display device connected remotely over the network to the Net2Display Host. The remoted session occupies the full screen of the remoted Display. This approach could be used either with:
  - Integrated Display - The Display passes display size across Net2Display Remoting during the formation of the Virtual Channel
  - Thin Client Adapter Box – The adapter gathers the display information from the display and passes it across Net2Display Remoting to the Host. The adapter may need the capability to change the display size for a Net Display Virtual Channel if a display is unplugged and a different display is plugged in or alternatively, it may terminate the Net Display Virtual Channel and open a new one of the new resolution.
- Terminal - The Client behaves as a terminal, with the contents from the Host occupying the full area of the Client display
- Windowed – The remoted material from the Host is displayed in a window on the Client. This may be the model for a full PC acting as a Net2Display Client.

Depending upon the model of the Display Remoting, it may be desirable to pass different types of configuration information to set up the Net Display Virtual Channel. The different information that would be passed according to each model is:

- Display – The Client pulls the EDID and extended EDID from the display and passes it to the Host. One of the resolutions specified in the EDID may then be used for the Display. The Host may then set attributes of the display using DDC type commands.
- Terminal – The Client presents the full dimensions of the display in pixels to the Host and the Host configures the display for the specified size. The Client may also present a set of possible resolutions through EDID or a configuration information structure.
- Windowed – The size of the window on the Client is passed from the Client to the Host. The window size on the Client may be changed by dragging it to a different size, which either:
    1. Causes the Client to display the Display data sent from the Host to a different size on the Client, accessible with scroll bars. This approach does not notify the Host and is independent of Net2Display Remoting functionality.
    2. Causes the Client to resize the Display data to fit within the new Client window size. This approach does not notify the Host and is independent of Net2Display Remoting functionality.
    3. Causes the Client to notify the Host of the new display area and the Host then generate a display size to match the new area. This approach uses Net2Display Remoting to communicate the change of the display size to the Host.

Note: The Client is expected to communicate its Display Window size to the Host. If the Client does not communicate a Display Window Size, the Host will use a default Display Window size of VGA (640x480).

11.3 Display Remoting Primitives

The video command set consists of categories of commands which:

- Send graphical objects to the display device
- Manipulate graphical objects - Instruct the display as to where graphical objects should be placed and how they should be combined with other graphical objects.
- Setup and manage Video streams

The video command set is intended to be unambiguous in its interpretation. The host should, at all times, be able to maintain a completely accurate internal model of what is being displayed. Commands which might produce varying results according to the algorithm employed have been specifically excluded. Examples of commands which fall into this category are drawing of geometric primitives such as circles, triangles, etc. and the rendering of fonts.

As a general remoting architecture, Net2Display Remoting supports the remoting of displays for different operating systems. Microsoft's Windows operating systems require that certain conditional required functions be supported for remoting. These conditional required functions are DrvCopyBits, DrvStrokePath and DrvTextOut. These are in addition to other enabling and configuration functions that are required to be supported. While, the enabling and configuration functions can be supported indirectly by the initialization of the remoting Association, the support of conditional required functions is important for efficient remoting. These conditional required functions are supported through the efficient mapping to Net2Display Display Remoting Commands. These mappings are as follows:

- DrvCopyBits – The DrvCopoyBits function is used to copy from one region to another region on the display. The Net2Display Copy Display Remoting Command provides equivalent functionality to the DrvCopyBits function and each DrvCopyBits function can be mapped to a single Copy Display Remoting Command
- DrvStrokePath - This function strokes a path when called by GDI.

- DrvTextOut

Display content consists of several different types of data with different delivery requirements. Display data includes moved and changed cursors, text that has been moved or manipulated and modified graphics.

For best responsiveness, the cursor movements should be made at the display. Cursor movements sent from the Host to the Client should be done with low latency.

The set of Display Remoting Primitives are as follows:

- RawPixel – Used to transmit pixel data to be displayed in a rectangular region on the screen
- Copy - Copy Framebuffer contents from one region of buffer to another
- SolidFill – Fill a region with a single solid color
- PatFill – Fill a region with a pattern
- BiFill – Fill a region with one foreground and one background color with a bitmap specifying the foreground fill region

11.4 Image Data Format

The Image Data contained within the Display Remoting PDUs is mapped differently depending upon the Colorimetery used. These different mappings are based on the mappings used in DisplayPort Version 1.1a. The mappings for all the different Colorimeteries are given in this section. For all different Colorimetries used, if the Image Data does not end on a word boundary, the end of the data is padded with zeros to end the data on a word boundary.

11.4.1 RGB Encodings

11.4.1.1 18 bpp RGB (6 Bits/Component)

The mapping of 18 bpp RGB is shown in Figure 11-5. Bit 5 of R0 is mapped to bit 7 of Byte 1, while bit 4 of G0 is mapped to bit 0 of Byte 1.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	R0-5:0|G0-5:4	G0-3:0|B0-5:2	B0-1:0|R1-5:0	G1-5:0|B1-5:4
1	B1-3:0|R2-5:2	R2-1:0|G2-5:0	B2-5:0|R3-5:4	R3-3:0|G3-5:2
2	G3-1:0|B3-5:0	R4-5:0|G4-5:4	G4-3:0|B4-5:2	B4-1:0|R5-5:0
. . .	. . .	. . .	. . .	. . .

Figure 11-5: Image Data Format with 18 bpp RGB

11.4.1.2 24 bpp RGB (8 Bits/Component)

The mapping of 24 bpp RGB is shown in Figure 11-6. Bit 7 of each color is mapped to bit 7 in each byte.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	R0-7:0	G0-7:0	B0-7:0	R1-7:0
1	G1-7:0	B1-7:0	R2-7:0	G2-7:0
2	B2-7:0	R3-7:0	G3-7:0	B3-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-6: Image Data Format with 24 bpp RGB

11.4.1.3 30 bpp RGB (10 Bits/Component)

The mapping of 30 bpp RGB is shown in Figure 11-7.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	R0-9:2	R0-1:0|G0-9:4	G0-3:0|B0-9:6	B0-5:0|R1-9:8
1	R1:7-0	G1-9:2	G1-1:0|B1-9:4	B1-3:0|R2-9:6
2	R2-5:0|G2-9:8	G2-7:0	B2-9:2	B2-1:0|R3-9:4
. . .	. . .	. . .	. . .	. . .

Figure 11-7: Image Data Format with 30 bpp RGB

11.4.1.4 36 bpp RGB (12 Bits/Component)

The mapping of 36 bpp RGB is shown in Figure 11-8.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	R0-11:4	R0-3:0|G0-11:8	G0-7:0	B0-11:4
1	B0-3:0|R1-11:8	R1-7:0	G1-11:4	G1-3:0|B1-11:8
2	B1-7:0	R2-11:4	R2-3:0|G2-11:8	G2-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-8: Image Data Format with 36 bpp RGB

11.4.1.5 48 bpp RGB (16 Bits/Component)

The mapping of 48 bpp RGB is shown in Figure 11-9.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	R0-15:8	R0-7:0	G0-15:8	G0-7:0
1	B0-15:8	B0-7:0	R1-15:8	R1-7:0
2	G1-15:8	G1-7:0	B1-15:8	B1-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-9: Image Data Format with 48 bpp RGB

11.4.2 YCbCr 4:4:4 Encodings

11.4.2.1 24 bpp YCbCr 4:4:4 (8 Bits/Component)

The mapping of 24 bpp YCbCr 4:4:4 is shown in Figure 11-10. Bit 7 of each color is mapped to bit 7 in each byte.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cr0-7:0	Y0-7:0	Cb0-7:0	Cr1-7:0
1	Y1-7:0	Cb1-7:0	Cr2-7:0	Y2-7:0
2	Cb2-7:0	Cr3-7:0	Y3-7:0	Cb3-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-10: Image Data Format with 24 bpp YCbCr 4:4:4

11.4.2.2 30 bpp YCbCr 4:4:4 (10 Bits/Component)

The mapping of 30 bpp YCbCr 4:4:4 is shown in Figure 11-11.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cr0-9:2	Cr0-1:0|Y0-9:4	Y0-3:0|Cb0-9:6	Cb0-5:0|Cr1-9:8
1	Cr1:7-0	Y1-9:2	Y1-1:0|Cb1-9:4	Cb1-3:0|Cr2-9:6
2	Cr2-5:0|Y2-9:8	Y2-7:0	Cb2-9:2	Cb2-1:0|Cr3-9:4
. . .	. . .	. . .	. . .	. . .

Figure 11-11: Image Data Format with 30 bpp YCbCr 4:4:4

11.4.2.3 36 bpp YCbCr 4:4:4 (12 Bits/Component)

The mapping of 36 bpp YCbCr 4:4:4 is shown in Figure 11-12.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cr0-11:4	Cr0-3:0|Y0-11:8	Y0-7:0	Cb0-11:4
1	Cb0-3:0|Cr1-11:8	Cr1-7:0	Y1-11:4	Y1-3:0|Cb1-11:8
2	Cb1-7:0	Cr2-11:4	Cr2-3:0|Y2-11:8	Y2-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-12: Image Data Format with 36 bpp YCbCr 4:4:4

11.4.2.4 48 bpp YCbCr 4:4:4 (16 Bits/Component)

The mapping of 48 bpp YCbCr 4:4:4 is shown in Figure 11-13.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cr0-15:8	Cr0-7:0	Y0-15:8	Y0-7:0
1	Cb0-15:8	Cb0-7:0	Cr1-15:8	Cr1-7:0
2	Y1-15:8	Y1-7:0	Cb1-15:8	Cb1-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-13: Image Data Format with 48 bpp YCbCr 4:4:4

11.4.3 YCbCr 4:2:2 Encodings

11.4.3.1 16 bpp YCbCr 4:2:2 (8 Bits/Component)

The mapping of 16 bpp YCbCr 4:2:2 is shown in Figure 11-14.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cb0-7:0	Y0-7:0	Cr0-7:0	Y1-7:0
1	Cb2-7:0	Y2-7:0	Cr2-7:0	Y3-7:0
2	Cb4-7:0	Y4-7:0	Cr4-7:0	Y5-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-14: Image Data Format with 16 bpp YCbCr 4:2:2

11.4.3.2 20 bpp YCbCr 4:2:2 (10 Bits/Component)

The mapping of 20 bpp YCbCr 4:2:2 is shown in Figure 11-15.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cb0-9:2	Cb0-1:0|Y0-9:4	Y0-3:0|Cr0-9:6	Cr0-5:0|Y1-9:8
1	Y1-7:0	Cb2-9:2	Cb2-1:0|Y2-9:4	Y2-3:0|Cr2-9:6
2	Cr2-5:0|Y3-9:8	Y3-7:0	Cb4-9:2	Cb4-1:0|Y4-9:4
. . .	. . .	. . .	. . .	. . .

Figure 11-15: Image Data Format with 20 bpp YCbCr 4:2:2

11.4.3.3 24 bpp YCbCr 4:2:2 (12 Bits/Component)

The mapping of 24 bpp YCbCr 4:2:2 is shown in Figure 11-16.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cb0-11:4	Cb0-3:0|Y0-11:8	Y0-7:0	Cr0-11:4
1	Cr0-3:0|Y1-11:8	Y1-7:0	Cb2-11:4	Cb2-3:0|Y2-11:8
2	Y2-7:0	Cr2-11:4	Cr2-3:0|Y3-11:8	Y3-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-16: Image Data Format with 24 bpp YCbCr 4:2:2

11.4.3.4 32 bpp YCbCr 4:2:2 (16 Bits/Component)

The mapping of 32 bpp YCbCr 4:2:2 is shown in Figure 11-17.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	Cb0-15:8	Cb0-7:0	Y0-15:8	Y0-7:0
1	Cr0-15:8	Cr0-7:0	Y1-15:8	Y1-7:0
2	Cb2-15:8	Cb2-7:0	Y2-15:8	Y2-7:0
. . .	. . .	. . .	. . .	. . .

Figure 11-17: Image Data Format with 32 bpp YCbCr 4:2:2

11.5 Net Display Compression Facilities

The different types of compression that are defined by Net2Display Remoting for use on Net Display Virtual Channel data are defined in Table 11-1. These codecs are communicated in the Virtual Channel Codec

Capability List as defined in Section 9.6.1.4, using the format defined in Section 9.6.1.5 to express each codec capability.

Note: The specific codecs and data structures to be used by Net2Display Remoting will be refined by the reference implementation and defined in the next version of the standard.

Table 11-1: Net Display Compression Types

Name	Codec_Type Value	Reference/ Standard	Lossless/ Lossy	IP Free	Type	Comments
JPEG baseline	JPEG	ISO 10918-1	Lossless or Lossy	Yes[1]	DCT	Baseline
JPEG-LS	JPEG-LS	ISO 14495-1	Lossless or Lossy	Yes	LOCO-I	Better lossless, Poor on highly lossy
JPEG 2000	JPEG 2000	ISO 15444-1:2000	Lossless or Lossy	Yes for Core	DWT Wavelet based	Better on lossy than JPEG, Codecs available
PNG	PNG		Lossless	Yes	LZ-77	Better for sharp transitions
GIF	GIF		Lossless	Yes[2]		Limited to 256 colors
RLE	RLE					Run Length Encoding
RDP RLC	RDP RLC					RDP Protocol with RLE Compression

11.6 Net Display Virtual Channel Initialization

11.6.1 Net Display Initialization

For the remoting of display information, all of the display elements, including Net Displays, Viewports and Motion Videos must be initialized between the Client and the Host. For these Display Remoting elements to be used, their sizes and properties need to be communicated between the Host and the Client. The initialization of Net Display Virtual Channels and surfaces, Viewports and Motion Videos proceeds in the following order:

1. **Net Display** Virtual Channels are opened, one per Net Display surface.
    a. The Client opens a Net Display Virtual Channel to the Host to initialize the Net Display surface. In opening the Net Display Virtual Channel to the Host, it passes the Net Display properties including size and color space.
2. **Viewport** surfaces are opened, typically, one for each visible display surface
    a. The Client informs the Host of each Viewport, typically representing an attached display, passing along EDID information and initial display resolution. If the Viewport is just a window on a display, the equivalent EDID information for that surface is passed.

[1] Forgent has an applicable patent on JPEG baseline that expired in October 2006.
[2] Software Freedom Law Center stated that after 1 October 2006, there are no significant interfering patent claims on the GIF compression.

b. The Host now places the Viewport in the Aggregated Coordinate Space, since only the Host is aware of all the Aggregated Coordinate Space placement of all the displays from all the different Clients that it manages.

c. The Host may then change the Viewport size either to match available driver capabilities or based on user input. The Viewport size may also be changed based on Client input.

3. **Motion Video** Virtual Channels are opened with one for each independent Motion Video. The Host opens any Motion Video windows to the Client and uses any available graphics space that it needs that was allocated in the Net Display region.

11.6.2 Net Display Specific Parameters

The Net Display Specific Parameters are passed on the opening of a Net Display Virtual Channel or on the changing of these parameters.

The Client initializes the opening of a Net Display surface and a ViewPort. The Host is responsible for placing the Viewport within the Aggregated Coordinate Space. The Client would request a Net Display area (Horizontal Pixel Count and Vertical Pixel Count) and color depth that it handles. The Host would then respond accepting that request or giving an error.

11.6.2.1 Net Display Window Placement Coordinates

The Display Window Placement Coordinates is passed from the Host to the Client to specify the location of Display Window in the Aggregated Coordinate Space. The location of the upper left corner of the Display Window is specified.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0001		0x0008	
1	Horizontal_Placement			
2	Vertical_Placement			

Figure 11-18: Net Display Window Placement Coordinates Representation

Parameter_Type: A 16 bit value set to 0x0001.

Parameter_Length: A 16 bit value set to 0x0008

Horizontal_Placement: A 32 bit signed integer representing the horizontal location of the upper left corner of the Display Window in the Aggregated Coordinate Space.

Vertical_Placement: A 32 bit signed integer representing the vertical location of the upper left corner of the Display Window in the Aggregated Coordinate Space.

11.6.2.2 Net Display Pixel Count

The Net Display Pixel Count communicates the size of the display window by communicating by the horizontal and vertical sizes of the display window in pixels. This is the size in pixels that the remoted display will appear on the Client.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0002		0x0008	
1	Horizontal_Pixel_Count			
2	Vertical_Pixel_Count			

Figure 11-19: Net Display Pixel Count Parameter Representation

Parameter_Type: A 16 bit value set to 0x0002

Parameter_Length: A 16 bit value set to 0x0008

Horizontal_Pixel_Count: A 32 bit unsigned integer that specifies the horizontal size of the display in pixels.

Vertical_Pixel_Count: A 32 bit unsigned integer that specifies the vertical size of the display in pixels.

11.6.2.3 Colorimetry

The Colorimetry specifies the color encoding used by the Host for this Net Display region. The format of the Colorimetry parameter is as specified in Figure 11-20.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0003		0x0004	
1	Colorimetry			
	Colorimetry Byte 0	Colorimetry Byte 1	Colorimetry Byte 2	Colorimetry Byte 3

Figure 11-20: Colorimetry Parameter Representation

Parameter_Type: A 16 bit value set to 0x0003.

Parameter_Length: A 16 bit value set to 0x0004

Colorimetry: The Colorimetry is described in Section 10.3 and is expressed in the same format as communicated by Display Port.

Byte 0 of the Colorimetry is the DisplayPort Miscellaneous0 Data (MISC0, 8 bits)

- Synchronous Clock (bit 0)
  - 0 = Link clock and stream clock asynchronous
  - 1 = Link clock and stream clock synchronous
  - (When 1, the value M must be constant unless link clock down-spread enabled)
- Component format (bits 2:1)
  - 00 = RGB
  - 01 = YCbCr 4:2:2
  - 10 = YCbCr 4:4:4
  - 11 = Reserved
- Dynamic range (bit 3)
  - 0 = VESA range (from 0 to the maximum)
  - 1 = CEA range
- YCbCr Colorimetry (bit 4)
  - 0 = ITU-R BT601-5
  - 1 = ITU-R BT709-5
- Bit depth per color / component (bits 7:5)
  - 000 = 6 bits
  - 001 = 8 bits
  - 010 = 10 bits
  - 011 = 12 bits
  - 100 = 16 bits
  - 101, 110, 111 = Reserved

Byte 1 of the Colorimetry is the DisplayPort Miscellaneous1 Data (MISC1, 8 bits)

- Interlaced vertical total even (bit 0)
  - 0 = Number of lines per interlaced frame (consisting of two fields) is an odd number.
  - 1 = Number of lines per interlaced frame (consisting of two fields) is an even number.
- Stereo video attribute (bits 2:1)
  - 00 = No stereo video transported
  - 01

        - For progressive video, the next frame is RIGHT eye
        - For interlaced video, TOP field is RIGHT eye and BOTTOM field is LEFT eye
    - 10 = RESERVED and must not be used
    - 11
        - For progressive video, the next frame is LEFT eye
        - For interlaced video, TOP field is LEFT eye and BOTTOM field is RIGHT eye
- Bits 7:3 = Reserved
    - Set to 0's.

Byte 2: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

Byte 3: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

11.6.3 Viewport Specific Parameters

11.6.3.1 Viewport Placement Coordinates

The Viewport Placement Coordinates is passed from the Host to the Client to specify the location of Viewport in the Aggregated Coordinate Space. The location of the upper left corner of the Viewport is specified.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x1001		0x0008	
1	Horizontal_Placement			
2	Vertical_Placement			

Figure 11-21: Viewport Placement Coordinates Representation

Parameter_Type: A 16 bit value set to 0x1001.

Parameter_Length: A 16 bit value set to 0x0008

Horizontal_Placement: A 32 bit signed integer representing the horizontal location of the upper left corner of the Display Window in the Aggregated Coordinate Space.

Vertical_Placement: A 32 bit signed integer representing the vertical location of the upper left corner of the Display Window in the Aggregated Coordinate Space.

11.6.3.2 Viewport Pixel Count

The Viewport Pixel Count communicates the size of the display window by communicating by the horizontal and vertical sizes of the display window in pixels. This is the size in pixels that the remoted display will appear on the Client.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x1002		0x0008	
1	Horizontal_Pixel_Count			
2	Vertical_Pixel_Count			

Figure 11-22: Viewport Pixel Count Parameter Representation

Parameter_Type: A 16 bit value set to 0x1002

Parameter_Length: A 16 bit value set to 0x0008

Horizontal_Pixel_Count: A 32 bit unsigned integer that specifies the horizontal size of the display in pixels.

Vertical_Pixel_Count: A 32 bit unsigned integer that specifies the vertical size of the display in pixels.

11.6.3.3 Colorimetry

The Colorimetry specifies the color encoding used by the Host for this Viewport region. The format of the Colorimetry parameter is as specified in Figure 11-23.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x1003		0x0004	
1	Colorimetry			
	Colorimetry Byte 0	Colorimetry Byte 1	Colorimetry Byte 2	Colorimetry Byte 3

Figure 11-23: Colorimetry Parameter Representation

Parameter_Type: A 16 bit value set to 0x1003.

Parameter_Length: A 16 bit value set to 0x0004

Colorimetry: The Colorimetry is described in Section 10.3 and is expressed in the same format as communicated by Display Port.

Byte 0 of the Colorimetry is the DisplayPort Miscellaneous0 Data (MISC0, 8 bits)

- Synchronous Clock (bit 0)
  - 0 = Link clock and stream clock asynchronous
  - 1 = Link clock and stream clock synchronous
  - (When 1, the value M must be constant unless link clock down-spread enabled)
- Component format (bits 2:1)
  - 00 = RGB
  - 01 = YCbCr 4:2:2
  - 10 = YCbCr 4:4:4
  - 11 = Reserved
- Dynamic range (bit 3)
  - 0 = VESA range (from 0 to the maximum)
  - 1 = CEA range
- YCbCr Colorimetry (bit 4)
  - 0 = ITU-R BT601-5
  - 1 = ITU-R BT709-5
- Bit depth per color / component (bits 7:5)
  - 000 = 6 bits
  - 001 = 8 bits
  - 010 = 10 bits
  - 011 = 12 bits
  - 100 = 16 bits
  - 101, 110, 111 = Reserved

Byte 1 of the Colorimetry is the DisplayPort Miscellaneous1 Data (MISC1, 8 bits)

- Interlaced vertical total even (bit 0)
  - 0 = Number of lines per interlaced frame (consisting of two fields) is an odd number.
  - 1 = Number of lines per interlaced frame (consisting of two fields) is an even number.
- Stereo video attribute (bits 2:1)
  - 00 = No stereo video transported
  - 01
    - For progressive video, the next frame is RIGHT eye
    - For interlaced video, TOP field is RIGHT eye and BOTTOM field is LEFT eye
  - 10 = RESERVED and must not be used
  - 11
    - For progressive video, the next frame is LEFT eye

        - For interlaced video, TOP field is LEFT eye and BOTTOM field is RIGHT eye
- Bits 7:3 = Reserved
    - Set to 0's.

Byte 2: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

Byte 3: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

11.6.3.4 EDID

The EDID Parameter must be passed from the Client to the Host. If the Client is not capable of providing the full EDID capabilities to the Host, it must generate and pass an equivalent EDID to the Host that gives the known capabilities of the Viewport to the Host for Colorimetry and resolution negotiation. When the EDID is passed as a parameter, the Client is responsible for reading all of the Display EDID and extended EDID blocks and passing them as one parameter to the Host Net2Display Node. The procedure for reading the EDID and extended EDID is given in the VESA Enhanced Display Data Channel (E-DDC) Standard and the VESA Enhanced Extended Display Identification (E-EDID) Standard. The Host Net2Display Node instance may then pass on the EDID and Extended EDI information to the Host computer when requested. The EDID field in the EDID parameter contains the complete EDID information that is read including the Extended EDID information. The representation of the EDID Parameter is given in Figure 11-24.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x1004		EDID_Length	
1…	EDID and Extended EDID			

Figure 11-24: EDID Parameter Representation

Parameter_Type: A 16 bit value set to 0x1004.

EDID Length: Set to the total length of the EDID in bytes

EDID: Complete EDID information including all EDID and Extended EDID blocks

11.6.3.5 Viewport Size

The Viewport Size is an optional parameter specifies the dimensions of the size of the display in millimeters. The Viewport Size Parameter format is specified in Figure 11-25.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x1005		0x0008	
1	Max Horizontal Image Size			
	Max Vertical Image Size			

Figure 11-25: Viewport Size Parameter Representation

Parameter_Type: A 16 bit value set to 0x1005.

Parameter_Length: A 16 bit value set to 0x0008

Max Horizontal Image Size: the horizontal size of the display area in millimeters

Max Vertical Image Size: the vertical size of the display area in millimeters

11.7 Net Display Viewport Control PDUs

11.7.1 Open Viewport Request PDU

The Open Viewport Request PDU is sent by the Client to the Host to initialize a Viewport for display content. The format of the Open Viewport Request PDU is given in Figure 11-26.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x04	0x06	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Viewport ID	0x00 00	
5…	Viewport Parameters			

Figure 11-26: Open Viewport Request PDU Format

Command Code: The Command code is set to 0x06

Reserved Byte: The first byte of the PDU data area is reserved for future definition

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport number that is to be assigned to the Viewport being opened

Viewport Parameters: The set of Viewport parameters that characterize the Viewport being opened. The Viewport Parameters are defined in Section 11.6.3.

11.7.2 Open Viewport Response PDU

The Open Viewport Response PDU is sent by the Host to the Client in response to an Open Viewport Request. The format of the Open Viewport Response PDU is given in Figure 11-27.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x04	0x26	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Viewport ID	0x00 00	
5	ResponseCode			
6…	Viewport Parameters			

Figure 11-27: Open Viewport Response PDU Format

Command Code: The Command code is set to 0x06

Reserved Byte: The first byte of the PDU data area is reserved for future definition

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport number that is to be assigned to the Viewport being opened

ResponseCode: The ResponseCode gives a status for the Open Request. The different values that the Response Code may take on are given in Table 7-3.

Viewport Parameters: The set of Viewport parameters that characterize the Viewport being opened. The Viewport Parameters are defined in Section 11.6.3.

11.7.3 Close Viewport Request PDU

The Close Viewport Request PDU is sent by the Client or the Host to close a Viewport. The format of the Close Viewport Request PDU is given in Figure 11-28.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x04	0x07	0x00 18	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Viewport ID	0x00 00	
5	Viewport Close Reason Code			

Figure 11-28: Close Viewport Request PDU Format

Command Code: The Command code is set to 0x07

Reserved Byte: The first byte of the PDU data area is reserved for future definition

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport number of the Viewport being closed

Viewport Close Reason Code: An optional field that communicates the reason for closing the Viewport

11.7.4 Close Viewport Response PDU

The Close Viewport Response PDU is sent by the Client or the Host in response to the Close Viewport Request to close a Viewport. The Viewport is closed on the sending or receiving of the Close Viewport Response PDU. The format of the Close Viewport Response PDU is given in Figure 11-29.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x04	0x27	0x00 18	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Viewport ID	0x00 00	
5	ResponseCode			

Figure 11-29: Close Viewport Response PDU Format

Command Code: The Command code is set to 0x07

Reserved Byte: The first byte of the PDU data area is reserved for future definition

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport number of the Viewport being closed

ResponseCode: The ResponseCode gives a status for the Close Request. The different values that the Response Code may take on are given in Table 7-3.

11.8 Net Display Virtual Channel Transfer PDU

Net Display Data commands are used for transporting display data. The general format of Net Display Transfer Request PDUs is shown in Figure 11-30. There are no Responses required for Net Display Transfer Request PDUs, but responses may be optionally generated to indicate error conditions.

Word	Byte 0				Byte 1		Byte 2	Byte 3
0	0x00				Virtual Channel ID			
1	0x04				0x0	Command Code	PDU Length	
2	VC_Timestamp (Optional)							
3	Sequence Number						Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID			Codec Index	
5	Image Width							
6	Image Height							
7	Destination X							
8	Destination Y							
9	Source X							
10	Source Y							
11…	Image Data (If present)							

Figure 11-30: Net Display Transfer Request PDU Format

Command Code: The set of Display Remoting commands are as follows:

- RawPixel – Used to transmit pixel data to be displayed in a rectangular region on the screen
- Copy - Copy Framebuffer contents from one region of buffer to another
- SolidFill – Fill a region with a single solid color
- PatFill – Fill a region with a pattern
- BiFill – Fill a region with one foreground and one background color with a bitmap specifying the foreground fill region

Display Remoting commands share the same basic format. The fields that are present in the different Display Remoting commands are shown in Table 11-2. All display commands in this version of the Net2Display Standard must be interpreted by a Client. A Host does not have to generate all of the different Display commands, but must be able to transfer display images with the display commands it uses.

Table 11-2: Display Command Parameter Summary

Display Command	Command Code	Image Width	Image Height	Destination X, Y	Source X,Y	Compression Type	Data Format	PDU Length	Data
RawPixel	0x01	X	X	X	-	X	X	X	Image Data
Copy	0x02	X	X	X	X	-	-	0	-
SolidFill	0x03	X	X	X	-	X	-	1	Fill Color
PatFill	0x04	X	X	X	-	X	-	X	Image Data
BiFill	0x05	X	X	X	-	X	-	X	Image Data

PDU Length: The total size of this PDU. The PDU Length includes the size of all fields in the PDU, including both the Header and the Command Data.

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: The Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channel Codex Capability List provided by the Client. The first item in the Virtual Channel Codex Capability List is numbered 1 and 0 indicates that no codec is used.

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: The x coordinate of the upper left corner of the location where the RawPixel data is to be placed

Destination Y: The y coordinate of the upper left corner of the location where the RawPixel data is to be placed

Source X: The x coordinate of the upper left corner of the location where the data is to be copied from

Source Y: The y coordinate of the upper left corner of the location where the data is to be copied from

Image Data: The image data in the negotiated color key and encoded using the specified encoder

11.8.1 RawPixel Display PDU Request

This command is required and must be supported by all Net2Display Host and Client implementations. The format of the RawPixel PDU is shown in Figure 11-31.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x04				0x01	PDU Length	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		Codec Index	
5	Image Width						
6	Image Height						
7	Destination X						
8	Destination Y						
9	0x00 00 00 00						
10	0x00 00 00 00						
11…	Image Data						

Figure 11-31: RawPixel Display PDU

PDU Length: The total size of this PDU, including PDU Header and Command Data

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: The Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channel Codex Capability List provided by the Client. The first item in the Virtual Channel Codex Capability List is numbered 1 and 0 indicates that no codec is used.

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the RawPixel data is to be placed

Destination Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the RawPixel data is to be placed

Image Data: The image data in the negotiated Colorimetry and encoded using the specified encoder

11.8.2 Copy Display PDU Request

This command is optional to be used by Net2Display Hosts, but it must be supported and processed by all Net2Display Client implementations. The Copy Display PDU format is shown in Figure 11-32.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x04				0x02	0x00 28	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		0x00 00	
5	Image Width						
6	Image Height						
7	Destination X						
8	Destination Y						
9	Source X						
10	Source Y						

Figure 11-32: Copy Display PDU

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: 0 indicates that no codec is used

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the data is to be placed

Destination Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the data is to be placed

Source X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the data is to be copied from

Source Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the data is to be copied from

11.8.3 SolidFill Display PDU

This command is optional to be used by Net2Display Hosts, but it must be supported and processed by all Net2Display Client implementations. The format of the SolidFill Display PDU is given in Figure 11-33.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x04				0x03	0x00 30	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		0x00 00	
5	Image Width						
6	Image Height						
7	Destination X						
8	Destination Y						
9	0x00 00 00 00						
10	0x00 00 00 00						
11 …	Fill Color						

Figure 11-33: SolidFill Display PDU

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: 0 indicates that no codec is used

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the RawPixel data is to be placed

Destination Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the RawPixel data is to be placed

Fill Color: The Fill Color is in the format of the Colorimetry defined for this Net Display remoting.

11.8.4 PatFill Display PDU

This command is optional to be used by Net2Display Hosts, but it must be supported and processed by all Net2Display Client implementations. The format of the PatFill Display PDU is given in Figure 11-34.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x04				0x04	PDU Length	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		Codec Index	
5	Image Width						
6	Image Height						
7	Destination X						
8	Destination Y						
9	0x00 00 00 00						
10	0x00 00 00 00						
11…	Image Data						

Figure 11-34: PatFill Display PDU

PDU Length: The total size of this PDU, including PDU Header and Command Data

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: The Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channl Codex Capability List provided by the Client. The first item in the Virtual Channel Codex Capability List is numbered 1 and 0 indicates that no codec is used

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the RawPixel data is to be placed

Destination Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the RawPixel data is to be placed

Image Data: The image data in the negotiated Colorimetry and encoded using the specified encoder

11.8.5 BiFill Display PDU

This command is optional to be used by Net2Display Hosts, but it must be supported and processed by all Net2Display Client implementations. The Format of the BiFill Display PDU is given in Figure 11-35.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x04				0x05	PDU Length	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		Codec Index	
5	Image Width						
6	Image Height						
7	Destination X						
8	Destination Y						
9	0x00 00 00 00						
10	0x00 00 00 00						
11…	Image Data						

Figure 11-35: BiFill Display PDU

PDU Length: The total size of this PDU, including PDU Header and Command Data

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the update to the Viewport is the last update in a group of updates. The Flip Frame bit is provided to give a hint to the Client to enable keeping updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in a group of updates to a Viewport. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: The Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channel Codex Capability List provided by the Client. The first item in the Virtual Channel Codex Capability List is numbered 1 and 0 indicates that no codec is used

Image Width: The width of the image being passed expressed in pixels

Image Height: The height of the image being passed expressed in pixels

Destination X: A 32 bit signed integer representing the x coordinate of the upper left corner of the location where the RawPixel data is to be placed

Destination Y: A 32 bit signed integer representing the y coordinate of the upper left corner of the location where the RawPixel data is to be placed

Image Data: The image data in the negotiated color key and encoded using the specified encoder

12 Motion Video Remoting

A Motion Video displays a sequence of images (frames) rapidly enough that the eyes see the image as a continuously moving picture. A Motion Video Virtual Channel uses a single encoding format at one time. Motion Video uses one Virtual Channel per Motion Video stream. Typically, the Virtual Channel is opened with the start of the Motion Video and closed when the Motion Video is terminated. A Color Key is available for using with the Motion Video Virtual Channel to specify where the video is shown on the Client so that the Motion Video Virtual Channel capability can be used when overlap causes non rectangular video display areas.

The Motion Video Virtual Channel typically uses UDP for transport, but if UDP transport is not available, TCP may be used.

While motion video data may be sent both from a Client camera to the Host computer and from the Host to the Client display, motion video data remoted from a Client camera is typically carried in a USB Virtual Channel and does not use the Motion Video Virtual Channel. Video cameras are typically connected to USB, which has its own transport requirements and is discussed in the USB section. Thus, the Motion Video Virtual Channel is used only for carrying motion video updates to the Client Display. The quality of displayed video is dependent on the number of PDUs dropped or delivered too late for playback. As with audio, there are different scenarios for video:

- Video Conferencing – this real time video requires low latency and acceptable quality
- Video Playback – several seconds latency acceptable with higher quality

If high quality can be achieved at low latency, then both modes can be supported with the same transport means.

The Motion Video remoting model is shown in Figure 12-1.

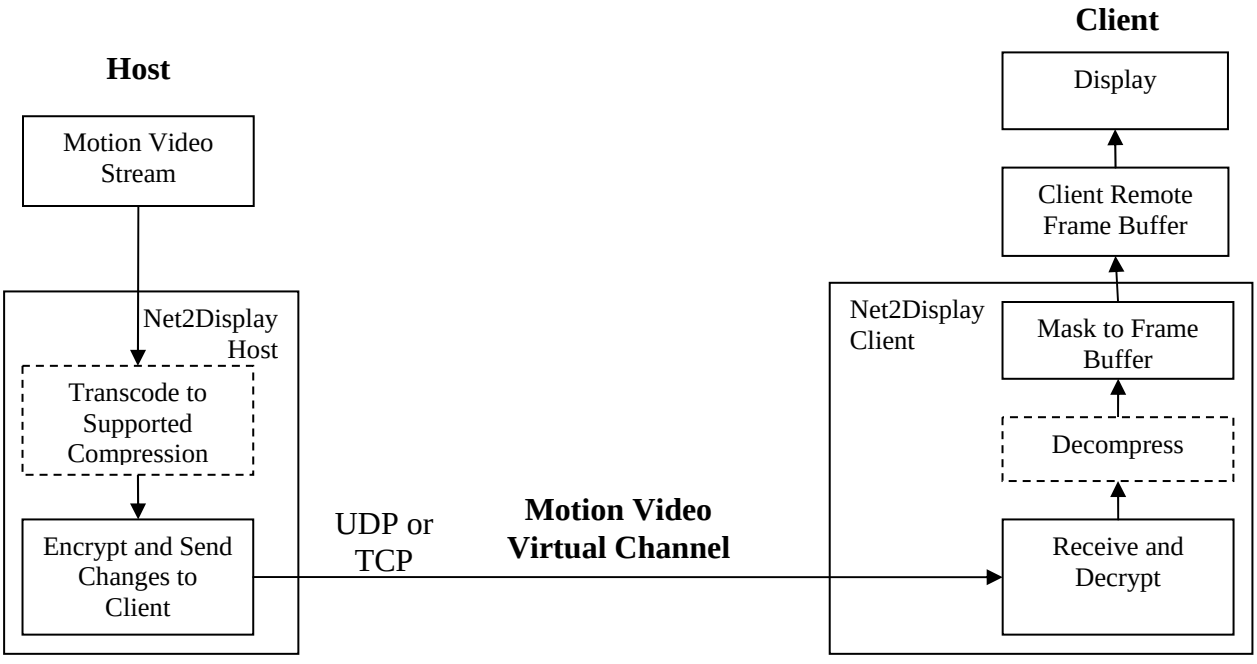


Figure 12-1: Motion Video Remoting Model

12.1 Motion Video Compression

The different codecs that may be used for compression on Motion Video Virtual Channels are specified in Table 12-1. These codecs are communicated in the Virtual Channel Codec Capability List as defined in Section 9.6.1.4, using the format defined in Section 9.6.1.5 to express each codec capability.

Table 12-1: Motion Video Codec Types

Name and XML Attribute Name	Codec_Type	Required/Optional	Reference/ Standard	Lossless/ Lossy	IP Free	Comments
MPEG-2	MPEG2		ISO 13818	Lossy to Visually Lossless	No[1]	Used by DVDs
MPEG-4 ASP	MPEG4 ASP		ISO MPEG-4 Part 2	Lossy to Visually Lossless	No[1]	
H.264/AVC	H.264 AVC		ISO MPEG-4 Part 10	Lossy	No[1]	Better Video quality than WMV9 for same rate[2]
Windows Media Video 9 (VCI)	VCI		SMPTE 421M		???	
Motion – JPEG 2000	MJPEG 2000					No

12.2 Motion Video Virtual Channel Control

12.2.1 Motion Video Initialization

The Motion Video is initialized by opening the Virtual Channel using the Virtual Channel Open PDUs as defined in Section 9.5.1 with the necessary parameters.

12.2.2 Motion Video Virtual Channel Specific Parameters

Motion Video Virtual Channel specific attributes include the video size, the video frame rate and the video encoding.

12.2.2.1 Motion Video Window Placement Coordinates

The Motion Video Placement Coordinates is passed from the Host to the Client to specify the location of Motion Video in the Aggregated Coordinate Space. The location of the upper left corner of the Motion Video is specified.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0001		0x0008	
1	Horizontal_Placement			
2	Vertical_Placement			

Figure 12-2: Virtual Channel Motion Video Placement Coordinates Representation

Parameter_Type: A 16 bit value set to 0x0001.

Parameter_Length: A 16 bit value set to 0x0008

[1] License acquirable from www.mpegla.com

[2] Terhi Mustonen et. al., "Subjective Video Quality of WMV9 and H.264/AVC: Comparison of Codec Performance at Four Target Bit Rates," Nokia Research Center, 3GPP Video Codec Ad-Hoc Meeting, October 28-30, 2003.

Horizontal_Placement: A 32 bit signed integer representing the horizontal location of the upper left corner of the Motion Video in the Aggregated Coordinate Space.

Vertical_Placement: A 32 bit signed integer representing the vertical location of the upper left corner of the Motion Video in the Aggregated Coordinate Space.

12.2.2.2 Source Pixel Count

The Source Pixel Count communicates the size of the image being sent from the source by communicating by the horizontal and vertical sizes. The Source Pixel Count must equal the Pixel Count of the Encoded Source Video. The format of this parameter is given in Figure 12-3. This parameter is the size in pixels of the video that is being transferred. This parameter is most useful for the Motion Video Virtual Channel. The default of the Source Pixel count is that it equal to the Net Display Window Pixel Count values. If this parameter is not recognized by the receiving Client, a response of invalid Parameter must be given.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0002		0x0008	
1	Source_Horizontal_Pixel_Count			
2	Source_Vertical_Pixel_Count			

Figure 12-3: Source Pixel Count Parameter Representation

Parameter_Type: A 16 bit value set to 0x0002

Parameter_Length: A 16 bit value set to 0x0008

Source_Horizontal_Pixel_Count: A 32 bit unsigned integer that specifies the horizontal size of the display in pixels.

Source_Vertical_Pixel_Count: A 32 bit unsigned integer that specifies the vertical size of the display in pixels.

12.2.2.3 Motion Video Pixel Count

The Motion Video Pixel Count communicates the size of the Motion Video by communicating by the horizontal and vertical sizes of the Motion Video in pixels. This is the size in pixels that the remoted display will appear on the Client.

If the Motion Video Pixel Count does not equal the Source Pixel Count, then the Motion Video is scaled to match the Motion Video Pixel Count at the destination. This operation requires the support of Motion Video scaling at the Client. If there is a problem with this scaling, an error will be sent in a Response PDU from the Client.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0003		0x0008	
1	Horizontal_Pixel_Count			
2	Vertical_Pixel_Count			

Figure 12-4: Motion Video Pixel Count Parameter Representation

Parameter_Type: A 16 bit value set to 0x0003

Parameter_Length: A 16 bit value set to 0x0008

Horizontal_Pixel_Count: A 32 bit unsigned integer that specifies the horizontal size of the display in pixels.

Vertical_Pixel_Count: A 32 bit unsigned integer that specifies the vertical size of the display in pixels.

12.2.2.4 Motion Video Region List

The Motion Video Region List is the preferred way to specify the area of the display where the communicated display information will be visible on the Net Display region by expressing a series of rectangular regions. The format of the Motion Video Region list is given in Figure 12-5.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0004		Parameter_Length	
1	Rect 1 X low			
2	Rect 1 Y low			
3	Rect 1 X high			
4	Rect 1 Y high			
…				
N+1	Rect N X low			
N+2	Rect N Y low			
N+3	Rect N X high			
N+4	Rect N Y high			
…				

Figure 12-5: Motion Video Region List Parameter Representation

Parameter_Type: A 16 bit value set to 0x0004.

Parameter_Length: A 16 bit value set to the length of the Motion Video Region List parameter. The parameter length is equal to 16 times the number of rectangles that are included in the Motion Video Region List.

Rect 1 X low: A 32 bit signed integer representing the lower X coordinate for a corner of rectangle 1.

Rect 1 Y low: A 32 bit signed integer representing the lower Y coordinate for a corner of rectangle 1.

Rect 1 X high: A 32 bit signed integer representing the higher X coordinate for a corner of rectangle 1.

Rect 1 Y high: A 32 bit signed integer representing the higher Y coordinate for a corner of rectangle 1

…

Rect N X low: A 32 bit signed integer representing the lower X coordinate for a corner of rectangle N.

Rect N Y low: A 32 bit signed integer representing the lower Y coordinate for a corner of rectangle N.

Rect N X high: A 32 bit signed integer representing the higher X coordinate for a corner of rectangle N.

Rect N Y high: A 32 bit signed integer representing the higher Y coordinate for a corner of rectangle N.

12.2.2.5 Motion Video Color Key

The Motion Video Color Key optionally specifies the area of the display where the communicated display information will be visible.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0005		Parameter_Length	
1…	Color Key			

Figure 12-6: Motion Video Color Key Parameter Representation

Parameter_Type: A 16 bit value set to 0x0005.

Parameter_Length: A 16 bit value set to the length of the Color_Key parameter

Color_Key: The Color_Key is communicated using the negotiated Colorimetry

12.2.2.6 Video Frame Rate

The Video Frame Rate optionally communicates the source video frame rate with the format shown in Figure 12-7.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0006		0x0004	
1	Video_Frame_Rate			

Figure 12-7: Video Frame Rate Parameter Representation

Parameter_Type: A 16 bit value set to 0x0006.

Parameter_Length: A 16 bit value set to 0x0004

Video_Frame_Rate: The Video Frame Rate specifies the frame rate of the Video in microseconds per frame.

12.2.2.7 Colorimetry

The Colorimetry specifies the color encoding used by the Host for this Motion Video. The format of the Colorimetry parameter is as specified in Figure 12-8.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0003		0x0004	
1	Colorimetry			
	Colorimetry Byte 0	Colorimetry Byte 1	Colorimetry Byte 2	Colorimetry Byte 3

Figure 12-8: Colorimetry Parameter Representation

Parameter_Type: A 16 bit value set to 0x0003.

Parameter_Length: A 16 bit value set to 0x0004

Colorimetry: The Colorimetry is described in Section 10.3 and is expressed in the same format as communicated by Display Port.

Byte 0 of the Colorimetry is the DisplayPort Miscellaneous0 Data (MISC0, 8 bits)

- Synchronous Clock (bit 0)
  - 0 = Link clock and stream clock asynchronous
  - 1 = Link clock and stream clock synchronous
  - (When 1, the value M must be constant unless link clock down-spread enabled)
- Component format (bits 2:1)
  - 00 = RGB
  - 01 = YCbCr 4:2:2
  - 10 = YCbCr 4:4:4
  - 11 = Reserved
- Dynamic range (bit 3)
  - 0 = VESA range (from 0 to the maximum)
  - 1 = CEA range
- YCbCr Colorimetry (bit 4)
  - 0 = ITU-R BT601-5
  - 1 = ITU-R BT709-5
- Bit depth per color / component (bits 7:5)
  - 000 = 6 bits
  - 001 = 8 bits

- 010 = 10 bits
- 011 = 12 bits
- 100 = 16 bits
- 101, 110, 111 = Reserved

Byte 1 of the Colorimetry is the DisplayPort Miscellaneous1 Data (MISC1, 8 bits)

- Interlaced vertical total even (bit 0)
  - 0 = Number of lines per interlaced frame (consisting of two fields) is an odd number.
  - 1 = Number of lines per interlaced frame (consisting of two fields) is an even number.
- Stereo video attribute (bits 2:1)
  - 00 = No stereo video transported
  - 01
    - For progressive video, the next frame is RIGHT eye
    - For interlaced video, TOP field is RIGHT eye and BOTTOM field is LEFT eye
  - 10 = RESERVED and must not be used
  - 11
    - For progressive video, the next frame is LEFT eye
    - For interlaced video, TOP field is LEFT eye and BOTTOM field is RIGHT eye
- Bits 7:3 = Reserved
  - Set to 0's.

Byte 2: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

Byte 3: Reserved for future use by Net2Display Remoting. Set to 0x00 for this version.

12.2.3 Motion Video Change

The Virtual Channel Change_Parameter PDUs, defined in Section 9.5.2, is used to change any Motion Video parameters such as the video placement.

12.2.4 Motion Video Ending

A Motion Video is ended by closing the Motion Video Virtual Channel using the Virtual Channel End PDUs as defined in Section 9.5.3. The Motion Video is closed on the sending or receiving of the Close Virtual Channel Response PDU for the Motion Video Virtual Channel.

12.3 Motion Video Virtual Channel Command PDU

Motion Video remoting commands are optional. The support or nonsupport of this primitive must be indicated in the n2dDisplayPrimitivesSupported. The format of the Motion Video PDUs is shown in Figure 12-9.

Word	Byte 0				Byte 1	Byte 2	Byte 3
0	0x00				Virtual Channel ID		
1	0x28				0x01	PDU Length	
2	VC_Timestamp (Optional)						
3	Sequence Number					Received Sequence Number	
4	FF Bit	NF Bit	Rsv Bits	Viewport ID		Codec Index	
5…	Video Data						

Figure 12-9: Video Remoting PDU Format

PDU Length: The total size of this PDU, including PDU Header and Command Data

Command Code: The command code for the Motion Video Command is set to 0x01

FF Bit: The Flip Frame Bit (FF Bit) is set to indicate when the Motion Video PDU is the last in a frame. The Flip Frame bit is provided to give a hint to the Client to enable keeping frame updates within one Vsync to avoid tearing

NF Bit: The New Frame Bit (NF Bit) is set to indicate when the update is the first in new frame. The New Frame bit must be set in every PDU following a PDU where the Flip Frame bit was set.

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Viewport ID: The Viewport ID is a 12 bit field that indicates the Viewport that is the destination of this PDU. The Viewport ID can take on the following values:

- 0 – Update to Net Display space. Coordinates relative to Net Display coordinates
- N – Update relative to Viewport N.

Codec Index: The Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channl Codex Capability List provided by the Client. The first item in the Virtual Channl Codex Capability List is numbered 1 and 0 indicates that no codec is used.

VideoData: The video data associated with the specific CommandCode and encoded using the specified codec

13 Universal Serial Bus (USB) Remoting Architecture

USB Remoting allows devices attached to the Client to be remotely attached to the Host and to operate as if they were attached to the Host. The support of the USB remoting Virtual Channel by the Client is optional, but is required when USB ports are provided on the Client or USB is used to remote other facilities at the Client, such as keyboard, pointer or audio.

Net2Display remoted USB will be remoted at the USB Request Block (URB) level, following the approach used by the USB/IP project (http://usbip.sourceforge.net/). This approach is independent of the USB version number and is capable of providing support for future USB versions.

13.1 Remoted Devices

The USB device types and specific USB devices that are to be remoted from the Net2Display Client to Host are determined by the Client, Host and other management resources in an implementation specific manner. Additionally, some USB devices may be interpreted at the Client for Client usage before they are passed onto the Host, such as the keyboard or pointer. Any processing of these USB devices before they are passed onto the Net2Display Client instance is outside the scope of this standard.

13.2 Architecture and Model

In conventional local USB implementations, a USB Host Controller Driver (HCD) is present in the lowest layer of the device driver stack and abstracts the I/O interface of a host controller into a common API for USB device drivers. A USB PerDevice Driver (PDD) is then responsible for the control of each USB device. The USB Host Controller and its USB HCD provide USB PDDs with abstracted data input/output for I/O buffers. The conventional arrangement of the USB device driver model is shown in Figure 13-1.

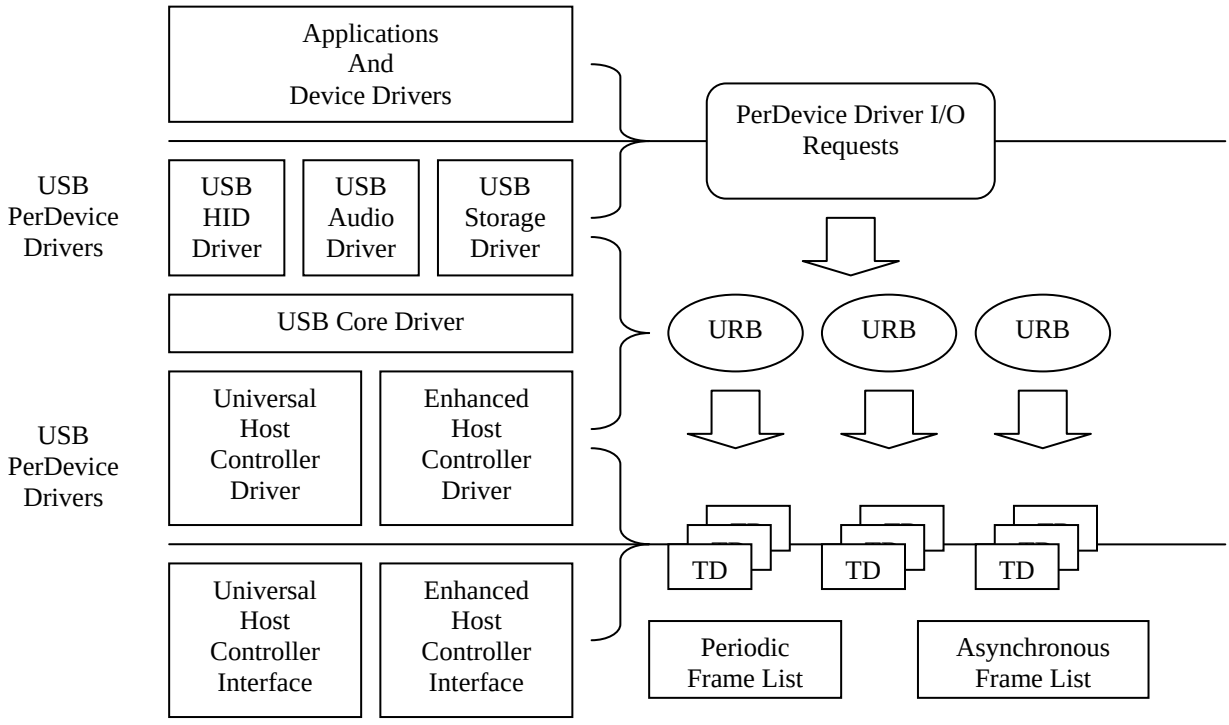


Figure 13-1: Conventional USB Device Driver Model

In USB device drivers, the USB Request Block (URB) presents I/O in a controller independent form, which includes: the I/O Buffer, the I/O direction, the I/O speed, the I/O type (Control/Bulk/Interrupt/Isochronous), I/O destination address and completion handler. USB URBs are formatted as defined in Microsoft Developer

Network: USB Structures and Enumerations (http://msdn.microsoft.com/en-us/library/ms793358.aspx) or the Linux format defined in struct urb of include/linux/usb.h available from http://kernel.org/..

The USB remoting model is shown in Figure 13-2. This model is based on using a Virtual Host Controller Interface (VHCI) driver which acts as a peripheral bus driver. The VHCI is typically built into a low layer of the operating system or virtualization on the Host so that applications and device drivers can access remote devices on the Client through existing software interfaces to these devices. The remoted USB devices are virtually attached to the Host computer via the VHCI driver, allowing the full functionality of the remoted device.

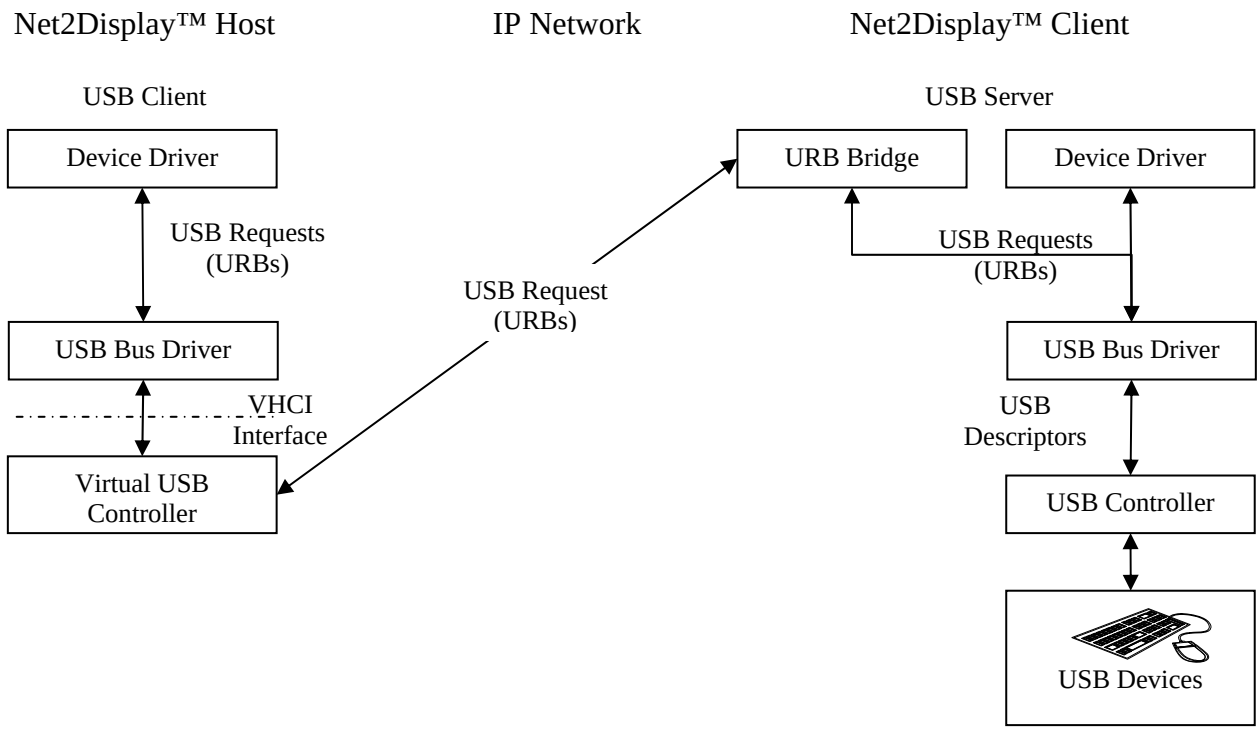


Figure 13-2: USB Remoting Model

13.2.1 USB Data Types

The four USB Data Types are Control, Isochronous, Interrupt and Bulk.

13.2.1.1 USB Control

Control transfers allow access to different parts of a device. Control transfers are intended to support configuration/command/status type communication flows between client software and its function. A control transfer is composed of a Setup bus transaction moving request information from host to function, zero or more Data transactions sending data in the direction indicated by the Setup transaction, and a Status transaction returning status information from function to host. Each USB device is required to implement the Default Control Pipe as a message pipe, which is used by the USB System Software to access the USB device's configuration, status, and control information.

The USB system will make a "best effort" to support delivery of control transfers between the host and devices. A function and its client software cannot request specific bus access frequency or bandwidth for control transfers. The USB System Software may restrict the bus access and bandwidth that a device may desire for control transfers.

13.2.1.2 USB Isochronous

In non-USB environments, isochronous transfers have the general implication of constant-rate, error-tolerant transfers. In the USB environment, requesting an isochronous transfer type provides the requester with the following:

- Guaranteed access to USB bandwidth with bounded latency
- Guaranteed constant data rate through the pipe as long as data is provided to the pipe
- In the case of a delivery failure due to error, no retrying of the attempt to deliver the data

While the USB isochronous transfer type is designed to support isochronous sources and destinations, it is not required that software using this transfer type actually be isochronous in order to use the transfer type.

13.2.1.3 USB Interrupt

The interrupt transfer type is designed to support those devices that need to send or receive data infrequently but with bounded service periods. Requesting a pipe with an interrupt transfer type provides the requester with the following:

- Guaranteed maximum service period for the pipe
- Retry of transfer attempts at the next period, in the case of occasional delivery failure due to error on the bus

13.2.1.4 USB Bulk

The bulk transfer type is designed to support devices that need to communicate relatively large amounts of data at highly variable times where the transfer can use any available bandwidth. The USB bulk transfer type provides the following:

- Access to USB on a bandwidth-available basis
- Retry of transfers, in the case of occasional delivery failure due to errors on the bus
- Guaranteed delivery of data but no guarantee of bandwidth or latency

Bulk transfers occur only on a bandwidth-available basis. For a USB with large amounts of free bandwidth, bulk transfers may happen relatively quickly; for a USB with little bandwidth available, bulk transfers may trickle out over a relatively long period of time.

13.3 Data Compression

Lossless data compression may optionally be used on USB transfers to improve the bandwidth efficiency of the USB transfers. Data compression on the USB channel may be useful for bulk data transfers. Only USB bulk data fields will be compressed if compression is enabled. When enabled, all bulk data will be compressed using a common algorithm.

Note: The specific algorithm has not been selected but is intended to be an IP free open source implementation that creates little overhead for data that cannot be compressed. The specific codecs and data structures to be used for USB bulk data compression will be refined by the reference implementation and defined in the next version of the standard. Examples of different types of compression that are considered for the USB bulk data compression are shown in Table 13-1.

Table 13-1: Lossless USB Virtual Channel Compression Types

Name	Codec_Type Value
LZW	LZW
Huffman- Coding	Huffman

13.4 USB Virtual Channel Commands

13.4.1 USB Transfer Request Block

The USB Transfer Request Block is used to transfer USB data. The format of the USB Transfer Request Block PDU is given in Figure 13-3.

Note: The USBRequestBlock (URB) contains the full contents of the USB Request Block including the data buffers reference by the URB. The URB format will be selected by the reference design and defined in the next version of this standard. The two alternatives are the Microsoft format as defined in http://msdn.microsoft.com/en-us/library/ms793340.aspx or the Linux format defined in struct urb of include/linux/usb.h available from http://kernel.org/.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x1X	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	USBDataType		CompressionType	
5…	USBRequestBlock			

Figure 13-3: USB Transfer Request Block Format

USBDataType: expresses the data type of the USB transfer as shown in Table 13-2. The values are the same as used in the USB Endpoint Type Values

Table 13-2: USBDataType Values

USBDataType	USBDataType Value
Control	0x0000
Isochronous	0x0001
Bulk	0x0002
Interrupt	0x0003

CompressionType: This field is reserved for future use for expressing the compression that is applied to the USB bulk data.

Note: Compression of USB URBs has not been fully defined and this field is to be reserved for definition in the next version of this standard.

USBRequestBlock: contains the full contents of the USB Request Block to be transferred. The URB structure is transmitted in entirety including the necessary data buffers referenced by the URB. Note that read data buffers are empty in the host to client direction and full on return. The addresses of the pointers in the URBs are not modified so replying URBs can reference the correct data buffers. Full data buffers are

concatenated onto the URB structure in the order they appear in the URB. The size of the URB structure and continuing data can be calculated from the URB information.

14 Keyboard Remoting

Keyboard remoting is used to transfer the keystrokes on the Client to the Host. Keyboards may be remoted either by sending across USB or using a dedicated Keyboard Virtual Channel. These different means of remoting keyboards and their requirements and attributes are addressed in this section.

14.1 Keyboard Support

Net2Display Clients may or may not have a keyboard present depending upon their configuration (a keyboard is optional for the Display Client Configuration as shown in Table 6-2 in Section 6.2.1). For specialized and limited functionality Clients, a keyboard is not a required component or accessory. Thus, Client keyboard support and keyboard remoting to the Host via a Keyboard Virtual Channel are both optional features of Net2Display Clients.

When a keyboard is supported on a Client, there are multiple ways of providing the remoting of the keyboard from the Host. The multiple different ways of supporting keyboards on the Client are shown in Table 15-1. A Client may allow only PS/2 keyboards, only USB keyboards or may allow both USB and PS/2 keyboards. Clients that have USB ports and support USB devices must support USB keyboards. If a Client does not support USB keyboards, then it must not have USB ports for other I/O devices. Thus, Keyboard data may be sent from the Client to the Host either via a Keyboard Virtual Channel or may be kept in the USB Virtual Channel for USB keyboards.

Table 14-1: Different Keyboard Approaches

Configuration	Comments	Attributes		
		Keyboard Types Supported	**Multiple Keyboard are Combined on the Client**	**Virtual Channel that carries Keyboard data**
USB and PS/2 Keyboards	Keyboard input is combined on the Client and sent on Keyboard Virtual Channel to the Host	USB and PS/2	Y	Keyboard Virtual Channel
PS/2 Keyboard Only with No USB Ports	Client Terminal that supports only PS/2	PS/2 only	N	Keyboard Virtual Channel
USB Keyboard Only over Keyboard Channel	Only USB over Keyboard Virtual Channel	USB only	Y	Keyboard Virtual Channel
USB Keyboard Only over USB Channel	USB Only Client	USB only	Y	USB Virtual Channel

Note: In all of the configurations shown, the keyboard may be used for local Client configuration or local Client operation.

14.2 Keyboard over USB Virtual Channel

The USB channel may only be used to transfer keyboard data when the Client does not support PS/2 Keyboards. When the USB channel is used to transfer keyboard data, USB specific protocols are used and the data formats and transfer protocols are outside the scope of this standard. With the USB Virtual Channel transporting keyboard data, multiple keyboards may be attached and their input is typically combined at the Host. Remoting of a USB keyboard over the USB Virtual Channel is illustrated in Figure 14-1. The remoting of USB keyboards over the USB Virtual Channel is not further defined in this section and must follow the USB remoting approach specified in Section 13.

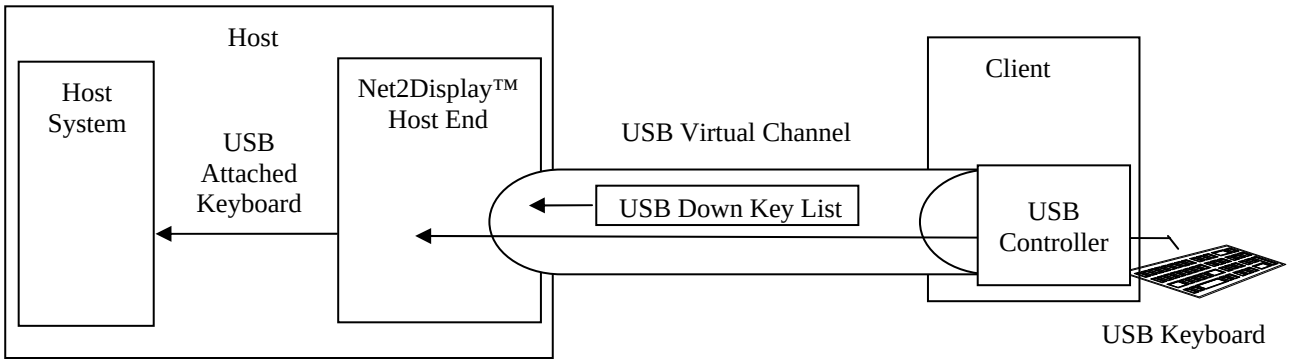


Figure 14-1: USB Keyboard over USB Virtual Channel

14.3 Keyboard Virtual Channel Operation

Keyboard remoting may be performed by setting up one optional Keyboard Virtual Channel on the Association between a Client and a Host. A Keyboard Virtual Channel is only opened up if it is requested by the Client using a Virtual_Channel_Needed_Request. Since the Keyboard Virtual Channel is optional and whether it will be used is determined by the Client, the Client must be the initiator of the formation of a Keyboard Virtual Channel.

If there are multiple keyboards being remoted across the Association, their input must be combined across the Keyboard Virtual Channel. Using a separate Keyboard Virtual Channel provides an advantage in allowing the keyboard data to be processed and transported with a higher priority, separate from other data traffic. However, the Keyboard Virtual Channel is optional and may be omitted if the support of PS/2 keyboards is not required on the Client. The Keyboard Virtual Channel, when present, requires a reliable in-order transport of very small packets of keystrokes from the Client to the Host with low latency for best interactivity.

A general block diagram of the usage of the Keyboard Virtual Channel is shown in Figure 14-2.

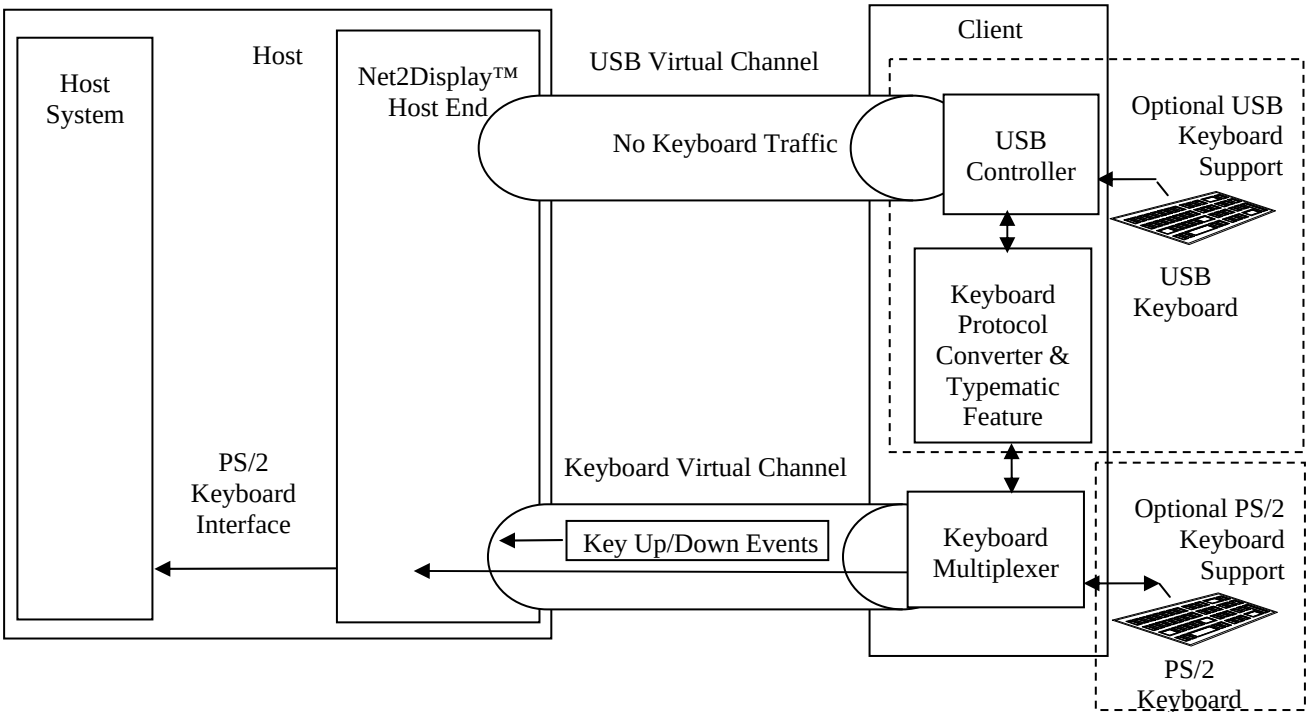


Figure 14-2: Keyboard Virtual Channel: USB and/or PS/2 Keyboards

14.3.1 Keyboard Model

The keyboard is remoted by passing the events to the Host when keys are pressed or released. LED status light settings are passed from the Host to the Client. The keyboard repeat function is performed at the Client End. In general, the overall keyboard remoting model is similar to a PS/2 keyboard interface. A description of the PS2 keyboard interface is found in “The PS/2 Keyboard Interface” by Adam Chapweske.

14.3.2 Keycodes

The Net2Display keyboard remoting model identifies keys using the USB keycodes as defined in the Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID) specification and the Universal Serial Bus (USB) HID Usage Tables specification.

14.3.3 Modifier Keys

Modifier keys (Left Ctrl, Left Shift, Left Alt, Left GUI, Right Ctrl, Right Shift, Right Alt, and Right GUI) must be reported in the Keycode field using their USB HID Code.

14.3.4 Keyboard Typematic Repeat

The keyboard repeat function, also known as typematic, provides the repeat capability of a key when it is held depressed. For PS/2 keyboards, the typematic feature is implemented in the keyboard, while USB keyboards provide the feature through a software driver. When keyboards are remoted across the Keyboard Virtual Channel, it is important that the typematic feature be implemented close to the Keyboard so that the typematic feature will remain responsive and will not trigger inadvertently.

For USB keyboards using the Keyboard Virtual Channel, the Client must provide the typematic function as shown in Figure 14-2.

For both PS/2 and USB keyboards, the Keyboard Virtual Channel provides optional Typematic Parameters for remotely controlling the typematic function from the Host that control the repeat rate and the repeat delay, the time before the repeat starts. For USB keyboards, these parameters control the typematic function in the Client; and for PS/2 keyboards, these parameters are passed on to the attached keyboards to modify the keyboard typematic characteristics. The default values for the keyboard typematic function must be set to the following before the keyboard is Associated to the Host to ensure consistency in initial operation:

- Typematic set to enabled
- Typematic Delay set to 500 milliseconds.
- Typematic Repeat Rate set to 10.9 characters per second

14.3.5 Keyboard LEDs

The Host keeps the state of the Keyboard LEDs and ensures that the Keyboard LEDs are set to the proper value upon formation of a Keyboard Virtual Channel or connection of a USB keyboard. Synchronization between LED states and CAPS LOCK, NUM LOCK, SCROLL LOCK, COMPOSE, and KANA events must be maintained by the Host and not by the Client or keyboard. On reconnection to a Host, the Client needs to send commands to the Host to reflect the current settings of the keyboard indicator LEDs so that they are understood by the Host. The LED Status indicators are set on all keyboards connected to the Client based on the data provided in the Net2Display Keyboard Output Command.

If multiple keyboards are attached through a Net2Display Association, all of the keyboards must share the same settings of the Keyboard LEDs. Thus, when the CAPS LOCK key is pressed on one keyboard, the key press is passed to the Host and a Keyboard Output Command with the setting of the CAPS LOCK LED is sent from the Host that must be delivered to all of the keyboards attached in the Association. This ensures that all of the keyboards reflect the current state of the modifier keys as understood by the Host and its applications.

14.3.6 Local Client Keyboard Usage

The Client is responsible for separating keyboard usage destined for the Host and Client systems in a manner outside the scope of this standard.

14.3.7 USB Keyboards over Keyboard Virtual Channel

A USB keyboard attached to a Client may optionally be used for both operations on the Client and remote Host operations. The sharing of Client attached USB keyboards between the Client and the Host is outside of the scope of this standard.

When USB keyboard input is passed across the Keyboard Virtual Channel, it is the responsibility of the Client to ensure that the keyboard input stream is not passed across the USB channel. The means for the redirection of this USB keyboard input to the Keyboard Virtual Channel is outside the scope of this standard.

The following USB reports are mapped to Net2Display Keyboard Virtual Channel Commands as follows:

- USB Input Report – The Net2Display Keyboard Input Command is used to transfer keyboard input from the Client to the Host
- USB Output Report – The Net2Display Keyboard Output Command is used to provide information to the keyboard for the setting of the LED status indicators.

14.3.8 Local Client Keyboard Echo

Keyboard input may be optionally displayed on the Client using a Local Client Keyboard Echo to lessen the effects of remoting latency. Facilities for implanting a Local Client Keyboard Echo are provided in this standard, although the specifics of an implementation are outside of the current Net2Display Standard

definition. If the keyboard input is generated on the Client, then the input from all of the keyboards attached to the Client must be combined at the Client. The local generation of keyboard input on the Client may be particularly beneficial for latencies greater than 300 milliseconds.

14.3.9 General Keyboard Support

The following are the requirements for remoting using the Net2Display Keyboard Virtual Channel:

- The Client must report a phantom keyboard state Usage(ErrorRollOver) whenever the number of keys pressed exceeds the Report Count. Additionally, a keyboard may report the phantom condition when an invalid or unrecognizable combination of keys is pressed.
- The Keyboard passes USB HID keyboard codes in the keyboard input PDUs

14.4 Keyboard Virtual Channel Initialization

When the Keyboard Virtual Channel is initialized, the Host sends the state of the LEDs to the keyboard using the keyboard output command. The Host may also optionally set the typematic parameters on initialization.

14.4.1 Keyboard Virtual Channel Typematic Parameters

The Typematic Parameters pass the Typematic Repeat Rate and the Typematic Delay. The Typematic Repeat Rate sets the repeat rate for all the keys on the keyboard that are repeatable. The default value of the Typematic Repeat Rate Parameter is given in Section 14.3.4. The format of the Typematic Repeat Rate Parameter is given in Figure 14-3.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0001		0x0004	
1	Typematic Enable	Typematic Repeat Rate	Reserved_Byte	Typematic Delay

Figure 14-3: Typematic Repeat Rate Parameter Representation

Parameter_Type: A 16 bit value set to 0x0004.

Parameter_Length: A 16 bit value set to 0x0008

Typematic Enable: When the Typematic Enable is set to zero, the typematic feature is disabled. When the Typematic Enable is set to 0x01, the typematic feature is enabled.

Typematic Repeat Rate: The Typematic Repeat rate must to set to one of the values shown in Table 14-2.

Table 14-2: Typematic Repeat Rate Parameter Values

Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)	Bits 0-4	Rate(cps)
0x00	30.0	0x08	15.0	0x10	7.5	0x18	3.7
0x01	26.7	0x09	13.3	0x11	6.7	0x19	3.3
0x02	24.0	0x0A	12.0	0x12	6.0	0x1A	3.0
0x03	21.8	0x0B	10.9	0x13	5.5	0x1B	2.7
0x04	20.7	0x0C	10.0	0x14	5.0	0x1C	2.5
0x05	18.5	0x0D	9.2	0x15	4.6	0x1D	2.3
0x06	17.1	0x0E	8.6	0x16	4.3	0x1E	2.1
0x07	16.0	0x0F	8.0	0x17	4.0	0x1F	2.0

Reserved_Byte: The Reserved_Byte is reserved for usage by a future version of Net2Display and must not be interpreted by a recipient of this command.

Typematic Delay: The Typematic Repeat rate must to set to one of the values shown in Table 14-3.

Table 14-3: Typematic Delay Parameter Values

Bits 5-6	Delay (seconds)
0b00	0.25
0b01	0.50
0b10	0.75
0b11	1.00

14.5 Keyboard Virtual Channel Commands

The Keyboard Virtual Channel has two commands, one for transferring keyboard input (Keyboard Input Command) and the other for transferring LED status indicator information (Keyboard Output Command).

14.5.1 Keyboard Input Command

The Keyboard Input Command PDU has a CommandCode value of 0x01. The format of this PDU is given in Figure 14-4. The latest key that was depressed or released is indicated in the Keycode field. Only one key may be indicated for each Keyboard Input Command PDU.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x08	0x01	0x00 14	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved_Byte	Keycode	Reserved_Byte	DownCode

Figure 14-4: Keyboard Input PDU Format

Command_Code: The Command Code is set to 0x01

Keycode: The USB HID code for the key that was pressed. This includes the indication of the pressing of Modifier Keys

DownCode: The DownCode is set to one on the depressing of the key indicate in Keycode. If the DownCode is set to 0x00, a key was released and the released key is indicated in the Keycode field.

Reserved_Bytes: The Reserved_Bytes are reserved for usage by a future version of Net2Display and must not be interpreted by a recipient of this command.

14.5.2 Keyboard Output Command

The Keyboard Output Command PDU has a CommandCode value of 0x02. The format of the Keyboard Virtual Channel Data PDU is given in Figure 14-5.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x08	0x02	0x00 14	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Keyboard_Status_Indicators	Reserved_Byte	Reserved_Byte	Reserved_Byte

Figure 14-5: Keyboard Output PDU Format

Command_Code: The Command Code is set to 0x02

Keyboard_Status Indicators: The Keyboard_Status_Indicators is a bit map of the LED indicators as defined in Table 14-4.

Table 14-4: Keyboard_Status_Indicators

Bit	Name	Description
0	Num Lock	Num Lock
1	Caps Lock	Caps Lock
2	Scroll Lock	Scroll Lock
3	Compose	Compose
4	Kano	Kano
5	Reserved	Reserved
6	Reserved	Reserved
7	Reserved	Reserved

Reserved Bytes: The Reserved_Bytes are reserved for usage by a future version of Net2Display and must not be interpreted by a recipient of this command.

15 Pointer Remoting

Pointer remoting is used for ensuring the proper representation of a cursor icon on a display and ensuring the transfer of pointer movements and the cursor location to the Host. Net2Display Remoting allows different ways of transferring the pointer information and representing the cursor on the display to allow flexibility in the Client design and to support a range of different remoting performances.

Net2Display Remoting may remote both PS/2 and USB pointers. A description of the PS2 mouse interface is found in “The PS/2 Mouse Interface” by Adam Chapweske. The USB pointer protocol is defined in the Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID) specification and the Universal Serial Bus (USB) HID Usage Tables specification.

Net2Display Remoting supports both Local Cursors and Host Cursors. Local Cursors are generated on the Client system and provide the best responsiveness since a network roundtrip is not required for cursor movements. Host Cursors are the simplest to implement and may be used when all pointers are USB and all pointer traffic is simply routed across USB. Dual Host and Local Cursors, where both Host and Local Cursors are present, are used when the responsiveness of the Local Cursor is desired but the Host system is not capable of omitting the drawing of the Host Cursor. Depending upon where pointers are combined, the Client or the Host is responsible for coordinating the operational and initialization of multiple pointer devices.

15.1 Pointer Transport

Pointer movements and locations may be transferred across either the USB Virtual Channel or an optional Pointer Virtual Channel. There may be one optional Virtual Channel set up to a Client for the pointer. When only USB pointers are allowed and there is no local pointer generated on the Client, the Pointer Virtual Channel is not required.

The Pointer Virtual Channel requires reliable in order delivery with low latency. The pointer requires a reliable in-order transport of very small PDUs of Pointer moves from the Client to the Host with low latency for best interactivity. The passing of pointer icons from the Host to the Client requires a reliable in-order transport of very small PDUs with low latency for best interactivity.

Note: When a Pointer Virtual Channel is opened, it is usually opened on the existing Primary Transport.

15.2 Host Pointer Remoting Architecture

While there are different means for attaching pointers at the Client and remoting them to the Host, the basic Host architecture for handling all types of pointers and remoting is a consistent architecture as shown in Figure 15-1. While there is flexibility in the Client configuration for attaching and remoting pointers, the Host must be capable of supporting the different types and remoting approaches for pointers. As shown in Figure 15-1, a Host must be capable of receiving pointer input on the USB or Pointer Virtual Channels. In addition, if the Client is configured to generate a Local Cursor, then the Host also sends Cursor Icon data on the Pointer Virtual Channel.

The Net2Display Host End typically passes the pointer data on to the Host System which typically combines multiple pointers and draws the cursor. The Host System may be capable of omitting drawing the Host Pointer if requested by the Client to enable a sole Local Cursor. The Host pointer architecture is independent of the Client pointer and must be capable of supporting the different Client pointer architectures and pointer remoting approaches

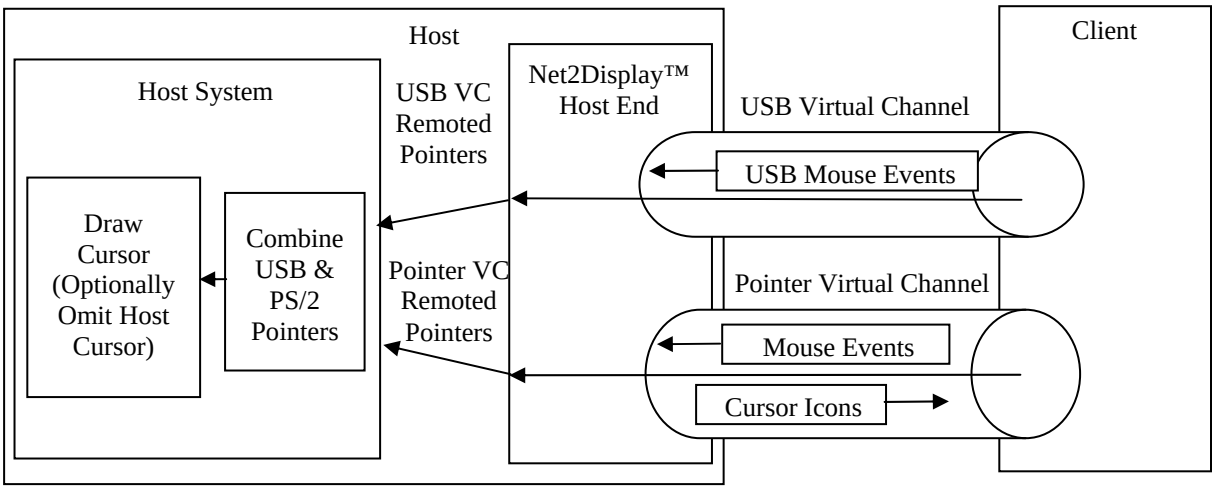


Figure 15-1: Host Pointer Architecture

15.3 Client Pointer Remoting Architecture Alternatives

There are multiple different ways of handling pointers on the Client as shown in Table 15-1. A Client may allow only USB pointer devices or may allow both USB and PS/2 pointer devices. Pointer data may be sent from the Client to the Host either via a Pointer Virtual Channel or may be kept in the USB Virtual Channel for USB pointers. Cursors may be generated either on the Client or at the Host. If the cursor is generated on the Client, then the input from all of the pointing devices attached to the Client must be combined before the Client cursor is displayed.

Table 15-1: Different Client Pointer Approaches

Client Pointer Approach	Pointer Types Supported	Virtual Channel Type	Combine Pointers Location	Draw Pointer Location
All Pointers are USB supported on USB Virtual Channel	USB Only	USB	Host or Client	Host
All Pointers are PS/2, no USB Ports	PS2 Only	Pointer	N/A	At Host and/or Client
USB and PS/2 Pointers Supported, No Combining of Pointers at Client	USB and PS/2	USB for USB Traffic and Pointer for PS/2 Traffic	Host	Host
USB and PS/2 Pointers Supported, Pointers Combined at Client and sent on Pointer VC	USB & PS/2	Pointer	Client	Host
USB and PS/2 Pointers Supported, Pointers Combined and Cursor drawn at Client, and sent on Pointer VC	USB (opt) & PS/2 (opt)	Pointer	Client	Client (and optionally at Host)

Note: Only one PS/2 pointer is typically supported at a Client, combining only PS/2 pointers is moot.

Note: Whether or not the pointer can be used for local Client usage or configuration is outside the scope of this standard. Net2Display Remoting is only concerned with remoting pointing devices that the Client has determined should be remoted and has provided the pointer data for remoting.

Thus, the significant differentiators in these approaches are where the pointers are combined and where the cursor is drawn. Cursors may be generated and displayed either:

- **USB pointers combined and drawn on Host**. This is the simplest configuration, but is not as responsive as the other approaches.
- **On the Client only**. In this configuration, the Host must be capable of not displaying the cursor and prevented from generating the cursor by negotiation.
- **On the Client and the Host**. In this case, the Client cursor will move immediately and the Host Cursor will later converge to the Client cursor position.

15.3.1 Client with Only USB Pointers Using USB Virtual Channel

The USB channel may be used to transfer pointer data when the Client does not support PS/2 pointers and the responsiveness of a network round trip for each pointer move is acceptable. When the USB channel is used to transfer pointer data, the USB specific protocols are used and the data formats and transfer protocols are outside the scope of this standard.

With the USB Virtual Channel transporting pointer data, multiple pointers may be attached and their input is typically combined at the Host. Remoting of a USB pointer over the USB Virtual Channel is illustrated in Figure 14-1. The remoting of USB pointers over the USB Virtual Channel is not further defined in this section and must follow the USB remoting approach specified in Section 13. If only USB pointers are supported and a network round trip is acceptable in the responsiveness, then no pointer specific implementation is required in the Client beyond the general USB remoting support.

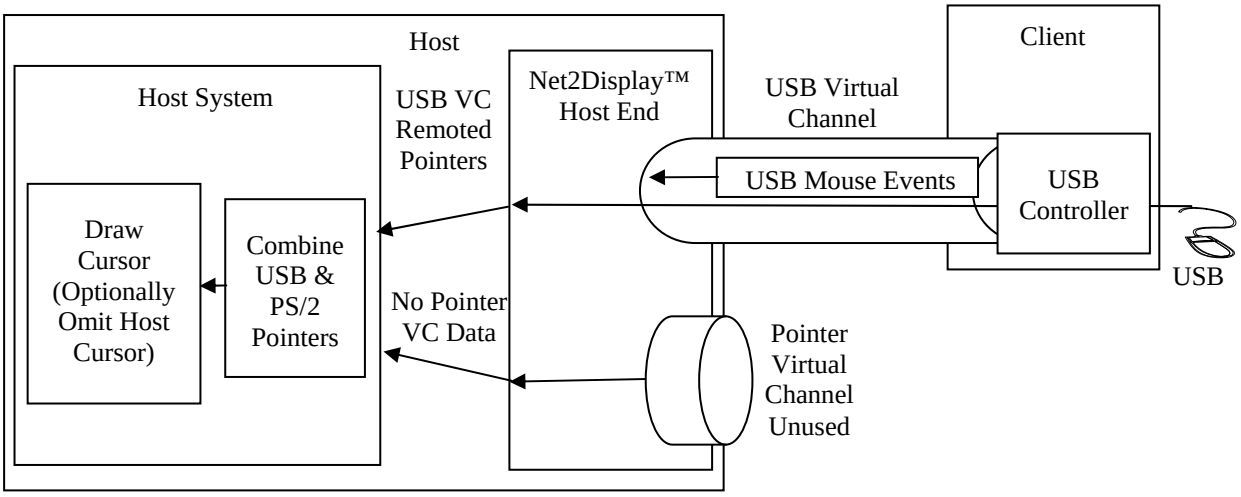


Figure 15-2: USB Pointers over USB Virtual Channel

15.3.2 Client with PS/2 Pointers Only

A Client may be built that only supports PS/2 pointers and does not support USB attached pointers. If a Client does not support USB pointers, then it must not have any USB ports for I/O as a USB pointer could be plugged into those ports. Such a Client may be a Terminal Client, a Secure Client or a Display Client, each of which may omit USB ports (See Section 6.2.1).

A Client that only supports PS/2 pointers is shown in Figure 15-3. The Client may optionally draw the Cursor, in which case it receives cursor icons on the Pointer Virtual Channel from the Host. The operation of drawing of the cursor at the Client, called a Local Cursor, is defined in Section 15.4.

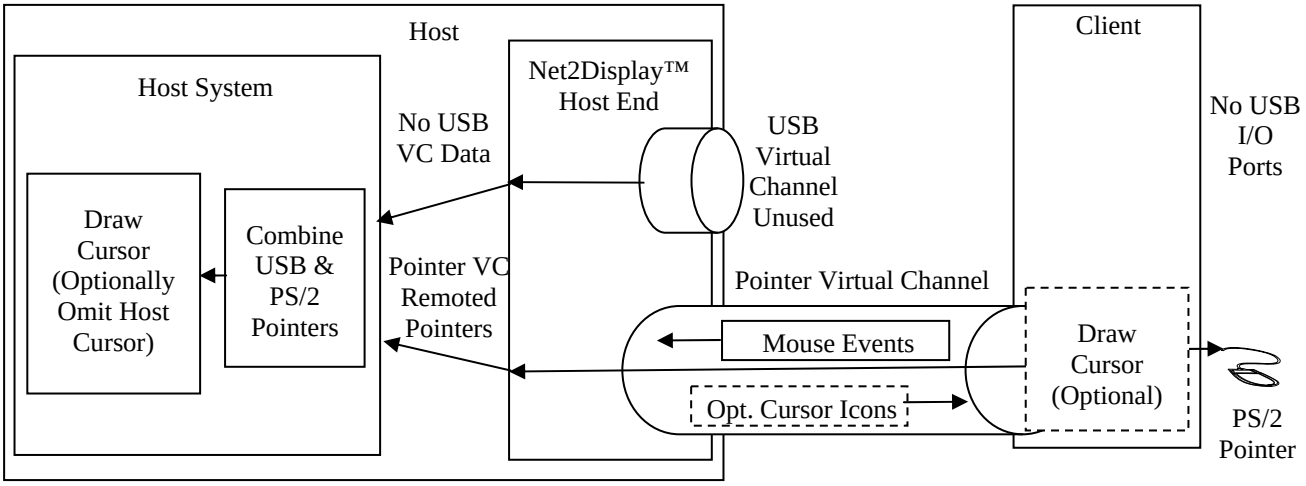


Figure 15-3: PS/2 Pointers Only

15.3.3 Pointers Combined at the Host

In configurations where the network latency is sufficiently low and USB and PS/2 pointers are both supported, both the USB and Pointer Virtual Channels can be used for passing pointer events and the pointers can be combined on the Host. For such an arrangement, the cursor has to be drawn on the Host, since the cursor must be drawn after the pointers are combined. This configuration is shown in Figure 15-4. This is the simplest configuration allowing both types of pointers and adds little complexity to the Client.

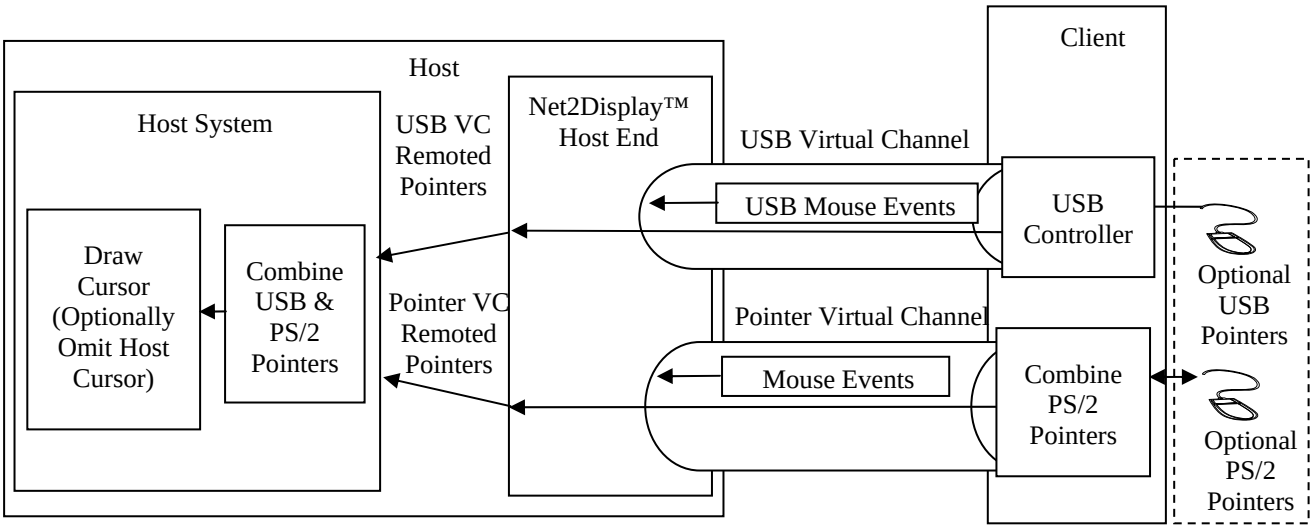


Figure 15-4: Pointers Combined at the Host

15.3.4 Pointers Combined at the Client and Cursor Drawn at Host

All pointers may also be combined at the Client and the cursor drawn on the Host as shown in Figure 15-5. This may be used for supporting both USB and PS/2 pointers when the Client is capable of combining pointers, but Host or Client capabilities prevent the drawing of the cursor on the Client.

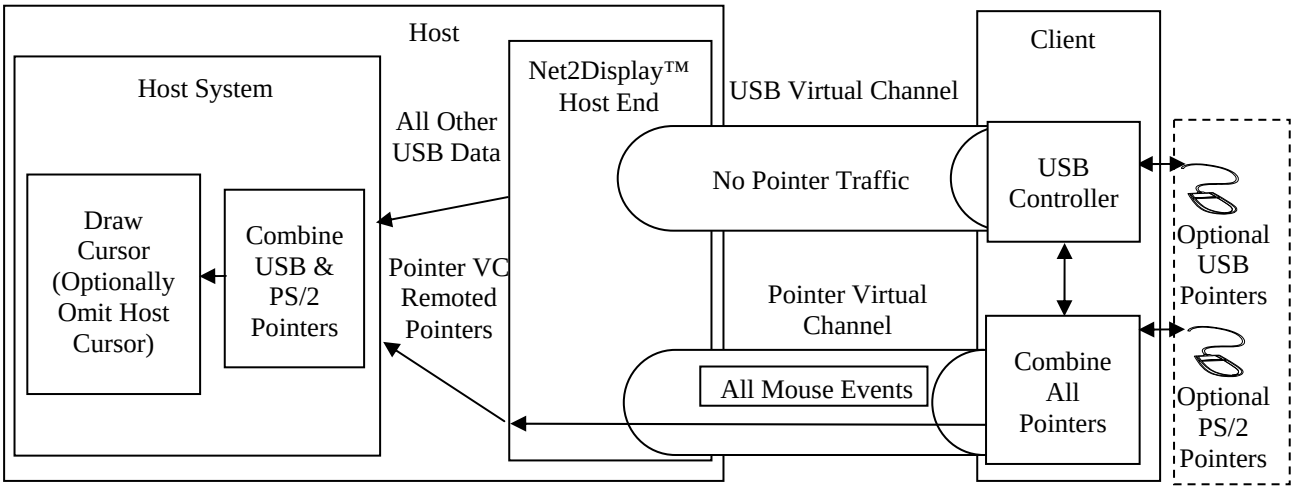


Figure 15-5: Pointers Combined at the Client

15.3.5 Pointers Combined and Drawn at the Client

The most complete and responsive approach is when pointers are combined and drawn on the Client. This configuration can be used when USB and PS/2 pointers are present or when only USB pointers are present. Since the pointer is generated locally at the Client, this approach is very responsive to cursor movements. This configuration may also be useful with the Client instantiation is a full PC and the pointers and cursor are all ready handled completely on the Client.

The configuration of the combining and drawing of the cursor on the Client is shown in Figure 15-6. For this configuration, USB pointer traffic must be separated from other USB traffic and terminated at the Client. The USB pointer traffic must be converted to pointer move commands to update the Client cursor and the Client cursor position passed on to the Host through the Pointer Virtual Channel. The complete operation of drawing of the cursor at the Client, called a Local Cursor, is defined in Section 15.4.

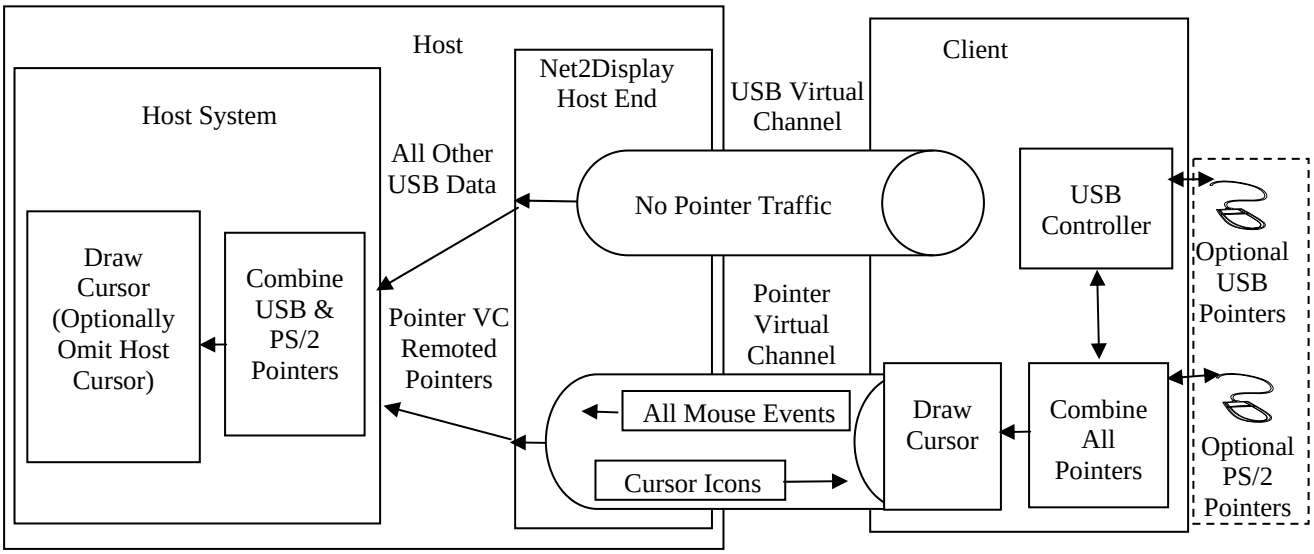


Figure 15-6: Pointers Combined and Cursor Drawn at the Client

15.4 Pointer Operation

A Pointer Virtual Channel is only opened up if it is requested by the Client using a Virtual_Channel_Needed_Request. Since the Pointer Virtual Channel is optional and whether it will be used

is determined by the Client, the Client must be the initiator of the formation of a Pointer Virtual Channel. During the Pointer Virtual Channel initialization, the Pointer Virtual Channel parameters are exchanged to set the characteristics of the pointer and the Pointer Virtual Channel. The Local Cursor mode is negotiated and initiated during the Pointer Virtual Channel initialization.

15.4.1 Local Cursor Operation

A Local Cursor is produced when the Client system draws the cursor icon. This requires several additional features for operation:

- Capability in the Client to support the Local Cursor
- Optionally, capability in the Host to omit the drawing of the Host Cursor
- Exchange of cursor drawing capabilities between the Host and the Client to set the cursor drawing mode.
- Entering of the Local Cursor mode, where cursor icons are passed from Host to the Client and absolute cursor position is passed from the Client to the Host

15.4.1.1.1 Local Cursor Negotiation

The Client and Host capabilities for Local Cursor support are exchanged during the Pointer Virtual Channel initialization using the Client Local Cursor Parameter as defined in Section 15.7.1.1. The following criteria are followed to determine the mode of operation that is used:

- If the Client does not support a Local Cursor, then the Local Cursor Mode must not be used
- If the Client supports Local Cursor and the Host supports Suppressed Host Cursor, then the Local Cursor Only Mode is used
- If the Client supports Local Cursor and the Host does not support Suppressed Host Cursor and both Client and Host allow Dual_Mode, then the Dual Cursor Mode is used
- If the Client supports Local Cursor and the Host does not support Suppressed Cursor Mode and the Client or Host does not allow Dual Mode, then only the Host Cursor must be used and the Local Cursor Mode must not be used.

15.4.1.1.2 Local Cursor Only Mode Operation

Local Cursor Only Mode is when a Local Cursor is drawn at the Client and the drawing of the Host Cursor is suppressed. It provides the highest responsiveness cursor operation.

In Local Cursor Only Mode, when the cursor is on an active area of the display, the Client must pass the absolute coordinates of the pointer location to the Host since the Client controls and has the primary knowledge of the cursor location. To provide for the transfer of the cursor to other displays, when the cursor reaches the edge of the display areas that it controls, it passes relative cursor movements off of the edge of the display to the Host, which may cause the Host to move the cursor to another display of an aggregated Client.

In Local Cursor Only Mode, the Host sends PointerIcon Cache Commands to the Client to cache cursor Icons and PointerIcon Display Commands to provide information to the Cursor on the cursor icon to use.

15.4.1.1.3 Dual Cursor Mode Operation

Dual Cursor Mode is when a Local Cursor is drawn at the Client and a second cursor is drawn at the Host. It provides a similar responsiveness to the Local Cursor Only mode, but could provide confusion in operation when the two cursor icons are both present.

In Dual Cursor Mode, when the cursor is on an active area of the display, the Client must pass the absolute coordinates of the pointer location to the Host so that the locations of the Client and Host cursors are coordinated. To provide for the transfer of the cursor to other displays, when the Client cursor reaches the edge of the display areas that it controls, it passes relative cursor movements off of the edge of the display to the Host, which may cause the Host to move the cursor to another display of an aggregated Client.

In Dual Cursor Mode, when the cursor movement is stopped or slowed, it is optional to remove the Local Cursor to avoid confusion.

In Dual Cursor Mode, the Host sends PointerIcon data commands to the Client to provide information on the Current cursor icon to use.

15.5 Absolute and Relative Pointer Coordinates

Pointer coordinates may be passed across the Pointer Virtual Channel from the Client to the Host or from the Host to the Client.

When the cursor is generated only on the Host, the Client typically passes pointer coordinates to the Host when a pointing device is moved or a button is pressed or released. The Client may pass either relative or absolute pointer coordinates, depending upon the pointing device being used. Relative coordinates are passed for relative input devices such as mice, trackballs, touchpads and joysticks. Absolute coordinates are passed for absolute input devices such as touchscreens and graphic tablets. When the cursor is generated only on the Host, the Host never passes cursor coordinates to the Client.

When the cursor is generated on the Client using Local Cursor Mode, the Client typically passes absolute coordinates to the Host since it must communicate the actual position of the cursor. When a Local Cursor is being generated, the Host may also send absolute pointer coordinates to the Client to update the position of the Client generated cursor.

15.6 Pointer Movement with Aggregated Clients

When multiple independent Clients are grouped together to act as a single Client, it is called an Aggregated Client. An example of an Aggregated Client is shown in Figure 3-6 and Aggregated Clients are described in Section 3.3.3. The mapping of the displays in an Aggregated Client to Net Displays is described in Section 10.1.5. The pointer control in Net2Display Remoting must support the movement of the cursor between the different Client devices and displays of an Aggregated Client. This is done by specifying when the cursor has reached the edge of a display and is ready to be moved to an adjacent display and turning off the cursor when it is no longer on a display. The different cursor handling configurations and their cursor handling for Aggregated Clients is shown in Table 15-2.

Table 15-2: Aggregated Client Cursor Handling

Cursor Mode	Coordinates Passed from Client to Host	Off the Edge Indicator	Comments
Host Cursor Only	Relative	Relative Movement off the edge	Since the cursor is completely handled at the Host, the Host can move the display of the cursor from one Client to another
Host Cursor Only	Absolute	Under Host Control	When cursor is moved with a absolute pointer next to an edge where there is an adjacent display, the Host should move the cursor to the adjacent display
Local Cursor (with Optional Host Cursor)	Absolute	Client needs to Indicate off the Edge to Host	Only the Client is aware of the relative movements of pointers that would move the cursor off an edge

15.7 Pointer Virtual Channel Initialization

15.7.1 Pointer Virtual Channel Specific Parameters

Pointer Virtual Channel specific attributes:

- Client Local Pointer attributes
- Pointer Resolution

15.7.1.1 Client Local Cursor Parameters

The Client Local Cursor Parameters provide the Client capabilities to determine if the cursor will be generated locally on the Client (as defined in Section 15.4). The representation of the Client Local Cursor Parameters is given in Figure 15-7.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0003		0x0004	
1	Client_Local_Cursor_Capable		Dual_Mode_Allowed_by_Client	

Figure 15-7: Client Local Pointer Parameter Representation

Parameter_Type: A 16 bit value set to 0x0003

Parameter_Length: A 16 bit value set to 0x0004

Client_Local_Cursor_Capable: The Client_Local_Cursor_Capable field is set to 0x1 if the Client is capable of supporting a Local Cursor and set to 0x0 if it is not capable.

Dual_Mode_Allowed_by_Client: The Dual_Mode_Allowed_by_Client field is set to 0x1 if the Client allows Dual Mode Cursors, where cursors are generated on both the Host and the Client, and is set to 0x0 if the Client does not allow Dual Mode Cursors to be used.

15.7.1.2 Host Local Cursor Parameters

The Host Local Cursor Parameters indicates the Host capabilities to determine if the cursor will be generated locally on the Client as defined in Section 15.4. The representation of the Host Local Cursor Parameters is given in Figure 15-8.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0004		0x0004	
1	Host_Cursor_Suppress		Dual_Mode_Allowed_by_Host	

Figure 15-8: Host Local Pointer Parameter Representation

Parameter_Type: A 16 bit value set to 0x0004

Parameter_Length: A 16 bit value set to 0x0004

Host_Cursor_Suppress: The Host_Cursor_Suppress field is set to 0x1 if the Host is capable of suppressing the display of its cursor to allow Local Cursor Only Mode operation (see Section 15.4.1.1.2) ands is set to 0x0 if the Host is not capable of suppressing its cursor.

Dual_Mode_Allowed_by_Host: The Dual_Mode_Allowed_by_Host field is set to 0x1 if the Host allows Dual Mode Cursors, where cursors are generated on both the Host and the Client, and is set to 0x0 if the Host does not allow Dual Mode Cursors to be used.

15.7.1.3 Pointer Resolution

The Pointer Resolution specifies the resolution of the pointer in steps per inch. The representation of the Pointer Resolution Parameter is given in Figure 15-9.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0005		0x0004	
1	Pointer Resolution			

Figure 15-9: Pointer Resolution Parameter Representation

Parameter_Type: A 16 bit value set to 0x0005

Parameter_Length: A 16 bit value set to 0x0004

Pointer Resolution: The Pointer Resolution is expressed in steps per inch

15.7.1.4 Pointer Cache Size Parameter

The Pointer Cache Size Parameter specifies the size of cache available for different pointer icons that the Client is capable of caching. The Pointer Cache Size Parameter is only applicable when the Local Cursor is used.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0006		0x0004	
1	Pointer Cache Size			

Figure 15-10: Pointer Cache Size Parameter Representation

Parameter_Type: A 16 bit value set to 0x0006

Parameter_Length: A 16 bit value set to 0x0004

Pointer Cache Size: The Pointer Cache Size is set to the size of the pointer cache in bytes. The number of pointer icons the Client is capable of caching is dependent on the sizes of the pointer icons. If the Client cannot cache any pointer icons then this value must be set to zero.

15.8 Pointer Virtual Channel Client to Host Commands

The following commands are sent from a Net2Display Client to a Net2Display Host on the Pointer Virtual Channel.

15.8.1 PointerButton Data Command

PointerButton Data PDUs flow from the Client to the Host and are sent on the pressing or releasing of a pointer button. The format of a PointerButton Data PDU is given in Figure 15-11.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x01	0x00 20	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	ButtonNumber			
5	ButtonDown			
6	XAbsolute			
7	YAbsoute			

Figure 15-11: PointerButton Data PDU Format

ButtonNumber: The ButtonNumber indicates the specific button that was pressed. ButtonNumber must be 1 for the left pointer button, 2 for the right pointer button and optionally 3 for a middle pointer button. Other pointer buttons may be assigned to other ButtonNumbers as necessary for the navigation devices being supported.

ButtonDown: The ButtonDown field is set to 1 when a pointer button is pressed and 0 when a pointer button is released.

XAbsolute: The x coordinate of the location of the press or release of the pointer button expressed in the Aggregated Coordinate Space

YAbsolute: The y coordinate of the location of the press or release of the pointer button expressed in the Aggregated Coordinate Space

15.8.2 PointerMove Data Command

PointerMove Data PDUs flow from the Client to the Host to indicate movements of an attached pointer device. The PointerMove Data command is used to allow a Host Cursor to follow the pointer movements or to update the movements of a Host Cursor to match those of a Local Cursor. For Associations that use only Local Cursors, the PointerMove Data command may be used to provide updates to the Host of the current cursor position. The format of a PointerMove Data PDU is given in Figure 15-12.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x02	0x00 28	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	PointerType		EdgeIndicators	
5	XAbsolute			
6	YAbsolute			
7	XRelative			
8	YRelative			
9	ZRelative			

Figure 15-12: PointerMove Data PDU Format

PointerType: The PointerType indicates the type of pointer information being passed. When PointerType is set to 1, only relative coordinates are being passed to the Host. When PointerType is set to 3, both absolute and relative data are being sent to the Host to be interpreted by the application. Relative Coordinates are always present in the pointer packet and are set to zero if no relative movement has occurred.

EdgeIndicators: The EdgeIndicators are set by the Client to indicate to the Host that the cursor is against an edge of a display and should be moved to an adjacent display if present. Two EdgeIndicator bits may be set to indicate that the cursor is at a corner of the display. The EdgeIndicators are as follows:

Bit 0: the cursor is against the bottom of the display

Bit 1: the cursor is against the left side of the display

Bit 2: the cursor is against the top of the display

Bit 3: the cursor is against the right of the display

XAbsolute: The x coordinate of the cursor position expressed in the Aggregated Coordinate Space

YAbsolute: The y coordinate of the cursor position expressed in the Aggregated Coordinate Space

XRelative: The x coordinate change being passed to the Host in two's complement notation. Relative Coordinates are set to zero if no relative movement has occurred.

YRelative: The y coordinate change being passed to the Host in two's complement notation. Relative Coordinates are set to zero if no relative movement has occurred.

ZRelative: The z coordinate change being passed, such as from a scroll wheel in two's complement notation. Relative Coordinates are set to zero if no relative movement has occurred.

15.9 Pointer Virtual Channel Host to Client Commands

The Following Commands are sent from a Net2Display Host to a Net2Display Client on the Pointer Virtual Channel. The Pointer Virtual Channel Host to Client commands are only used when a Local Cursor is being used. There are four types of pointer data that are sent from the Host to the Client:

1. Updates to the pointer location
2. Change the pointer icon to:
    a. A different non-cached icon
    b. A cached pointer icon
    c. The default operating system pointer
3. Commands to hide the pointer, reshow a pointer, or change to the operating system default pointer

15.9.1 PointerIcon Cache Command

This PDU is used if the Client is capable of drawing cursors. PointerIcon Cache Command PDUs flow from the Host to the Client to store pointer icons in the Client Pointer Cache. The format of a PointerIcon Cache Command PDU is given in Figure 15-13. The pointer icon is always passed using 24 bits for color.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x03	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Cache Number		Reserved	
5	HotSpotX		HotSpotY	
6	Image Width			
7	Image Height			
8 …	PointerIconData			

Figure 15-13: PointerIcon Cache PDU Format

PDU Length: Variable dependent on the size of the Cursor Icon

Cache Number: Assigns a cache number to this pointer icon for later reference. If the Client had specified that Caches were not supported by setting the Pointer Cache Size Parameter to 0, then the Host must specify the Cache Number as 0.

Reserved: Reserved

HotSpotX: This parameter is the pointer hot spot X coordinate in pixels relative to (top, left) of the pointer icon definition.

HotSpotY: This parameter is the pointer hot spot Y coordinate in pixels relative to (top, left) of the pointer icon definition.

Image Width: The width of the Cursor Icon image being passed expressed in pixels

Image Height: The height of the cursor image being passed expressed in pixels

PointerIconData: The pointer icon is passed using 24 bit color representation.

15.9.2 PointerIcon Display Command

PointerIcon Display Command PDUs flow from the Host to the Client. This PDU is used if the Client is capable of drawing cursors and passes the current cursor location and the current cursor cache location. The format of a PointerIcon Display Command PDU is given in Figure 15-14.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x04	0x00 1C	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Cache Number		Reserved	
5	XAbsolute			
6	YAbsoute			

Figure 15-14: PointerIcon Display Command PDU Format

Cache Number: The Cache Number specifies the index of the pointer icon to use.

Reserved: Reserved

XAbsolute: Absolute coordinate of the upper left of the Cursor Icon

YAbsolute: Absolute coordinate of the upper left of the Cursor Icon

15.9.3 SystemIcon Data Command

SystemIcon Data PDUs flow from the Host to the Client. This PDU is used if the Client is capable of drawing cursors and directs the Client to use the Client default system icon. The format of a SystemIcon Data PDU is given in Figure 15-15.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x05	0x00 1C	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved			
5	XAbsolute			
6	YAbsoute			

Figure 15-15: SystemIcon Data PDU Format

Reserved: Reserved

XAbsolute: Absolute coordinate of the upper left of the Cursor Icon

YAbsolute: Absolute coordinate of the upper left of the Cursor Icon

15.9.4 HideCursor Command

The HideCursor Command Data PDU is sent by a Host to a Client to indicate that the Cursor is to no longer to be shown on the Client. The HideCursor Command must only be sent to a Client that is operating with a Local Cursor. The format of the HideCursor PDU is given in Figure 15-16.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x0C	0x06	0x00 10	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	

Figure 15-16: HideCursor Command PDU Format

16 Audio Remoting Architecture

Audio includes both audio outputs from the Host to the Client for speaker output and microphone inputs on the Client to be transferred to the Host. The quality of audio is dependent on the number of PDUs dropped or delivered too late for playback. There are different scenarios for audio:

- Telephony and Video Conferencing – this real time audio requires low latency and acceptable quality
- Audio Playback – several seconds latency acceptable with higher quality

If high quality can be achieved at low latency, then both modes can be supported with the same transport means.

16.1 Architecture and Model

Attributes of the Audio Remoting Architecture are:

- Audio data is transmitted at a fixed rate based on a transmitter clock and must be rate converted by the receiver.
- For audio data sent over UDP, lost packets are not recovered. This requires the receiver to perform packet loss concealment.
- Rate conversion and packet loss concealment are implementation specific and are not covered by the standard.
- Each audio stream uses a separate Virtual Channel so only one codec stream is sent on one Virtual Channel.

The audio remoting model is shown in Figure 16-1. By default, multi Virtual Channel audio (video, PC speaker and local video) are mixed by the client without attenuation. A vendor specific implementation may provide a local mixer with an application interface."

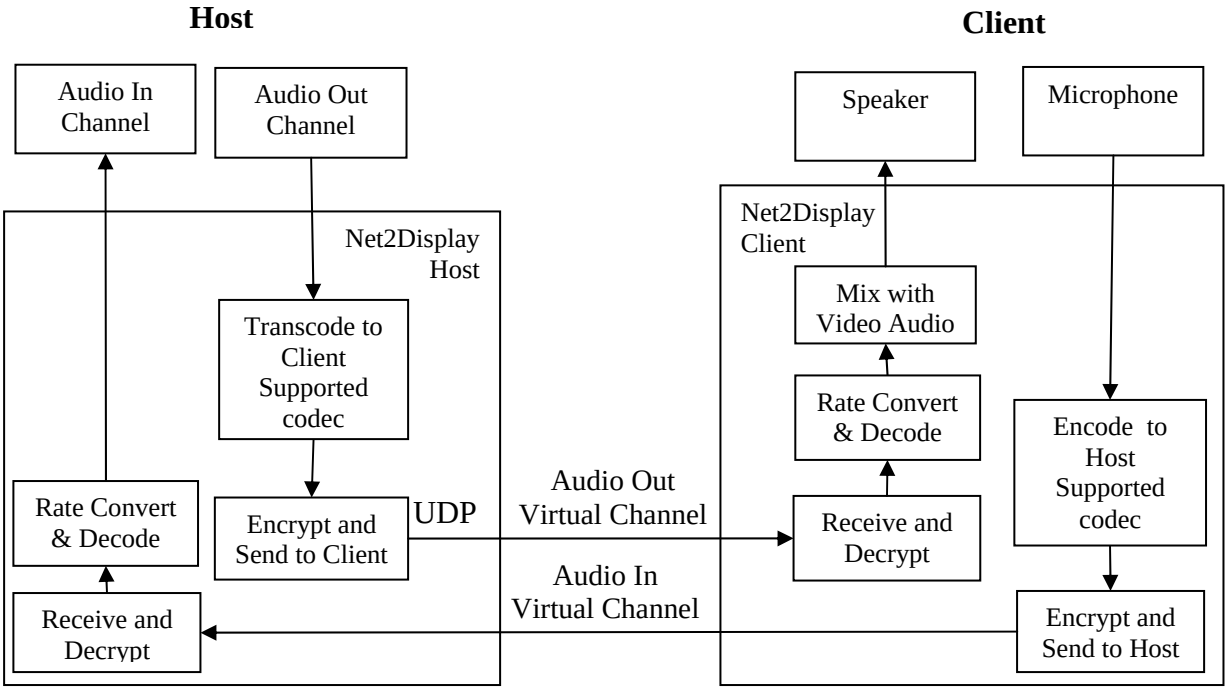


Figure 16-1: Audio Remoting Model

16.2 Facilities

16.2.1 Source Audio Format Requirements

Audio support is optional for Net2Display Host devices. The Net2Display Host devices that support audio must support stereo (two channel) 16 bit per sample LPCM at one or more of 32 kHz, 44.1 kHz or 48 kHz.

It is optional for an audio capable Net2Display Host device to support other sample rates, sample sizes or number of channels within the limits of the audio capability of the Net2Display Client device indicated in its EDID.

The Net2Display Host device must check via EDID or the CEA Timing Extension to EDID which audio formats the Client can support before sending any audio stream data.

The Net2Display Host device is recommended to find out whether the Net2Display Client device is able to receive the audio stream by checking the Audio Codec Types supported by the Net2Display Client.

16.2.2 Sink Audio Format Requirements

A Net2Display Client device that outputs audio must support an audio input stream via a Net2Display Audio or USB Virtual Channel. The audio output may be sound waves (speakers) or electrical analog or digital audio output.

Net2Display Client devices that support audio must support stereo 16 bit LPCM at 32 kHz, 44.1 kHz and 48 kHz. It is optional for all audio capable Net2Display Client devices to support other sample rates, sample sizes or number of channels.

Net2Display Client device must indicate via EDID or the CEA Timing Extension to EDID which audio formats are supported.

Note: As of this writing, only the CEA Timing Extension to EDID provides this information, but it is anticipated that future versions of VESA's E-EDID Standard may also specify audio capabilities of the Net2Display Client device, and these would be acceptable to a Net2Display Host device.

As is the case with sinking video stream, the Net2Display Client device must indicate whether it is able to sink the transmitted audio stream by indicating in its Audio Codec Types.

16.2.3 Audio Compression

The N2D audio system support different audio compression formats. The default compression format is mono DVI4 as described in RFC 3551.

16.2.4 Audio data formatting

- Audio data is communicated following RTP protocol RFC 3550 and RFC 3551.
- The RTP header is replaced with the PDU header supporting Timestamps and Sequence Numbers.
- Sequence numbers allow for detection of missing data.
- Time stamps allow audio gaps due to silence and then to reset timing on the next Sequence Number.
- Timing stamps can be used for rate conversion.
- Mono DVI4 data at 8 kHz sampling rate is the default compressed audio communication rate with 20 millisecond packets.
- Rate conversion at the source and destination is up to the implementation.
- Data payload is formatted as described in RFC 3551.

- Where the RFC does not support multi-channel (stereo), PDUs are formatted with all Left audio followed by all Right audio

16.3 Audio Codec Parameters

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x0001		0x0004	
1	Audio_Codec	Sample Size	Number of Audio Channels	Audio_Device_Number

Figure 16-2: Audio Codec Parameters Representation

Parameter_Type: A 16 bit value set to 0x0001.

Parameter_Length: A 16 bit value set to 0x0004

Audio_Codec: The Audio Codec type expressed in an 8 bit field. The following Audio Codec Types are supported:

- 0 – Uncompressed PCM data
- 1 – DVI4 compressed data

Sample_Size: The size of the audio samples in bits expressed in an 8 bit field. This value is set to 0 for an adaptive sample size. The sample size is set to zero for adaptive compression

Number_of_Audio_Channels – This parameter expresses the number of audio channels in an 8 bit field.

Audio_Device_Number – The number of the audio device expressed in an 8 bit field. The default values are 0 for left and 1 for right. Other values allow more then two speakers or microphones.

16.4 Audio Virtual Channel Commands

16.4.1 AudioOutData

The AudioOutData PDU is used to transfer audio output data from a Host to a Client within an Audio Out Virtual Channel. The format of the AudioDataOut PDU is given in Figure 16-3.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x20	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
5…	AudioData			

Figure 16-3: AudioDataOut PDU Format

AudioData: Audio Stream Data for output

16.4.2 AudioInData

The AudioInData PDU is used to transfer audio input data from a Client to a Host within an Audio In Virtual Channel. The format of the AudioInData PDU is given in Figure 16-4.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x24	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
5…	AudioData			

Figure 16-4: AudioDataIn PDU Format

AudioData: Audio Stream Data for input

17 Extended Virtual Channels

Extended Virtual Channels are used to encapsulate application or vendor specific data that is defined outside of the Net2Display Standard. Extended Virtual Channels are negotiated between the Host and Client. An Extended Virtual Channel is distinguished by the setting of the Extended Bit and the Control Bit not being set. Extended Virtual Channel PDUs must have the first 4 bytes of the Header set as defined in Section 7.2 and all other content of the PDU is Command Data associated with the Extended Virtual Channel. In an Extended Virtual Channel PDU, the following fields are required: Extend Bit set to one, Response Bit, Command Code, and Data Size.

Extended Virtual Channels may be used to pass undecoded image, video, audio or Flash data to Clients that have the capability to decode and process the data directly. This may allow for high performance and more responsive Client devices.

17.1 Extended Virtual Channel Initialization

The Initialization of an Extended Virtual Channel must follow the Virtual Channel Formation protocol defined in Section 9.4. The Extended Virtual Channel also exchanges Virtual Channel Parameters as defined in Section 9.6. Additional Extended Virtual Channel Parameters may be passed, but they must conform to the general Virtual Channel Parameter format defined in Section 9.6.

17.2 Extended Virtual Channel PDU Format

The format of an Extended Virtual Channel PDU is given in Figure 17-1.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x04	Virtual Channel ID		
1	Extended Virtual Channel Data		PDU Length	
2…	Extended Virtual Channel Data			

Figure 17-1: Extended Virtual Channel PDU Format

17.3 Extended Virtual Channel Commands

The Virtual Channel Commands for an Extended Virtual Channel are specific to that type of Extended Virtual Channel and their definition is outside the scope of the Net2Display Standard.

18 Principles of Operation

Note: This section describes example implementations of the Net2Display Standard. These are examples only and are not required implementations to be compliant with the standard.

18.1 Initialization

Net2Display can exist in a range of implementations from a single remote displaying instance to a full remotely hosted infrastructure with Aggregated Clients. Examples of the initialization in a simple environment to a full initialization are both given to show the full range of complexity that Net2Display supports.

18.1.1 Simple Direct Attach with no Input Facilities

Initialization of the example of a simple direct attach Client with no input facilities is shown in Figure 18-1.

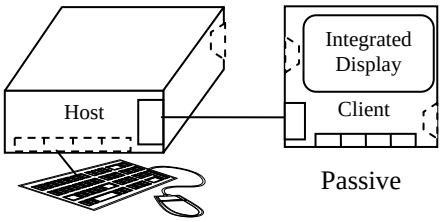


Figure 18-1: Single Integrated Display Direct Attachment with no Input Facilities

1) System Configuration
   a) The user has a new Net2Display enabled computer and connects a Net2Display Integrated Display directly to the Ethernet port on the computer without a keyboard or any I/O connected to the display.
   b) Both computer and Integrated Display are turned on in either order. The order does not affect the final result.
2) Host Initialization
   a) During start up, the Host examines its Display Connector(s) and finds no display attached.
   b) The Host then starts up its Net2Display instance.
   c) The Host Net2Display instance determines that the Host is connected to an isolated network without DHCP or DNS and notifies the Host Server.
   d) The Host offers Hosting Services and simultaneously the Host searches on the network for a Net2Display Client device offering its Client Services. The Host stays in this state until a Client requests the Host Services or Client Services are found.
3) Client Initialization
   a) The Integrated Client powers up and its Net2Display instance initializes.
   b) The Integrated Client detects that it is in a new network configuration and that it has no Input Facilities attached, so it goes into Passive Mode.
   c) Since it is in Passive Mode, the Client advertises its Net2Display Client Services.
4) Association Initiation
   a) The Host locates the Client and forms an Association with the Client.
   b) The Host Server then presents its login screen on the Client.

18.1.2 Simple Direct Attach Initialization

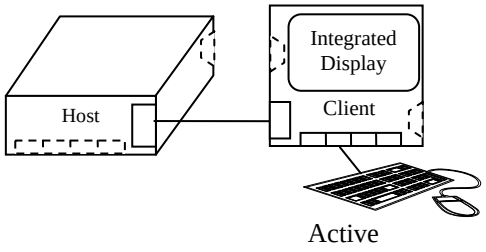


Figure 18-2: Single Integrated Display Direct Attach with Input Facilities

1) System Configuration
   a) The user has a new Net2Display enabled computer and connects a Net2Display Integrated Display directly to the Ethernet port on the computer with keyboard or other Input Facilities connected directly to the Integrated Display.
   b) Both computer and Display are turned on in either order. The order does affect the final result.
2) Host Initialization
   a) During start up, the Host examines its Display Connector(s) and finds no display attached.
   b) The Host then starts up its Net2Display Node instance.
   c) The Host Net2Display instance determines that the Host is connected to an isolated network without DHCP or DNS and notifies the Host Server.
   d) Each Net2Display Node continually checks for the presence of DHCP and DNS so that it can use those services as soon as they show up. The frequency of the checks should be low enough so that the overhead is low and high enough so that it is recognized in one minute so that it configured before the user believes there is a problem. The overhead of checking once a minute is not very high for an isolated network with a smaller number of network attachments.
      i) Check for a DHCP Server:
         (1) If the Node assigned itself an address, then it continuously checks for a DHCP server
         (2) If the user entered a static address into the Node, then the Node does not need to check for a DHCP server
   e) Check for DNS
   f) The Host offers Hosting Services and simultaneously the Host searches on the network for a Net2Display Display device offering its Client Services. The Host stays in this state until a Client requests the Host Services or Client Services are found.
3) Client Initialization
   a) The Integrated Client powers up and its Net2Display instance initializes.
   b) The Integrated Client detects that it is in a new network configuration and that it has Input Facilities attached, so it goes into Active Mode.
   c) Since it is in Active Mode, the Client searches for Net2Display Hosting Services.
   d) The Client locates the Host and forms an Association with the Host.
4) Client Association Initiation
   a) The Host Server then presents its login screen on the Client.

18.1.3 Infrastructure Environment Initialization

An example of an initialization in an infrastructure environment is:

1) Network Configuration – General network setup, independent of Net2Display Remoting. The recommendations for setting up a network amenable to Net2Display Remoting are given in Appendix G.
    a. Setup Network
    b. Setup DNS Services registration server
    c. Distribute Security Keys and Certificates – the recommended Security Authorities and Certificates are given in Appendix I.
2) Connection Management Server (CMS). The CMS is described in Section 2.4.
    a. Connection Management Server is initialized
    b. Connection Management Server registers with the DNS Service registry
3) Host Initialization
    a. Host system locates the Connection Management Server through the DNS Service registry
    b. Host system registers its services with the Connection Management Server
4) Client Initialization
    a. Client configured with connection procedure to use
    b. Client initializes and goes to the DNS Service registry to get the CMS Server address
    c. Client requests Host Service from the CMS Server
    d. CMS Server gives Host address to Client
5) Client Association Initiation
    a. Client requests a connection with the Host
    b. Client forms a Secure TLS connection over HTTP to the Host
    c. Client and Host Exchange Configuration information over SOAP
6) Primary Client to Host Association
    a. Client initiates IPsec channel with Host
    b. Client forms TCP connections to Host equal to number of connections and priorities negotiated
    c. Host opens Virtual Channels to Client to meet capabilities expressed by Client
        i. One Virtual Channel is opened for each separate communications path that is needed between the Host and Client.
7) Secondary Client Joining
    a. Client or Host directs the Secondary Client to connect to the Host (Security information could be passed on at this point)
        i. Using the Client to do this would get around NAT issues. How can the Client remember the full list of
        ii. Using the Host would be more consistent and scalable, but has NAT issues
8) Secondary Client

a. Secondary Client requests connection to the Host with the provided credentials
b. Secondary Client forms a Secure TLS connection over HTTP to the Host
c. Secondary Client and Host Exchange Configuration information over SOAP

9) Secondary Client to Host Association
    a. Secondary Client initiates IPsec channel with Host
    b. Secondary Client forms TCP connections to Host equal to number of connections and priorities negotiated
    c. Secondary Client opens Virtual Channels to Host for Display, Audio, USB, Keyboard, Mouse and other data types as necessary
        i. Is there one Virtual Channel per Video stream? Yes, virtual channel needs to specify position on display

18.2 Reconnection

When a Client becomes disconnected from a Host due to network issues, there are three different situations that may result:

1. There is no network path between the Client and the Host and reconnection is not possible.
2. The previous network path between the Host and Client is restored for a connection.
3. A new network path is formed between the Host and the Client, potentially using different network interfaces, so that the Client MAC address and IP address may be different.

For the last two cases, it is beneficial that the original Association exchanges a cookie or token that identifies the session, so that Association can be reestablished quickly after being broken. To reestablish the Association, the session identifier can be passed allowing it to be quickly reformed.

The Association_Cookie (as described in Section 8) passed during Association formation remains a secure secret shared between the host and client that allows a secure reconnection. The Association_Cookie takes precedence over the IP address and the MAC address.

19 Node Association Management (NAM)

Note: Net2Display state machines define the detailed behavior of Net2Display Nodes which will be more completely characterized in the next version of this standard after a reference design has been implemented.

19.1 Scope and Purpose

Net2Display Node Association Management (NAM) manages Net2Display Nodes and manages the formation, maintenance and Associations between Clients and Hosts. NAM performs the following functions:

- Node Initialization
- Address Assignment
- Host End Management
- Client End Management
- CMS Discovery
- Host Discovery
- Client Discovery
- Client Service Registration
- Host Service Registration
- Answer Requests for Client Service
- Answer Requests for Host Service
- Form Associations
- Exchange Configuration Information
- Manage Associations

19.2 Organization

NAM is composed of multiple cooperating state machines to support the flexibility that is needed in Net2Display Remoting. A general block diagram of a Net2Display NAM is given in Figure 19-1.

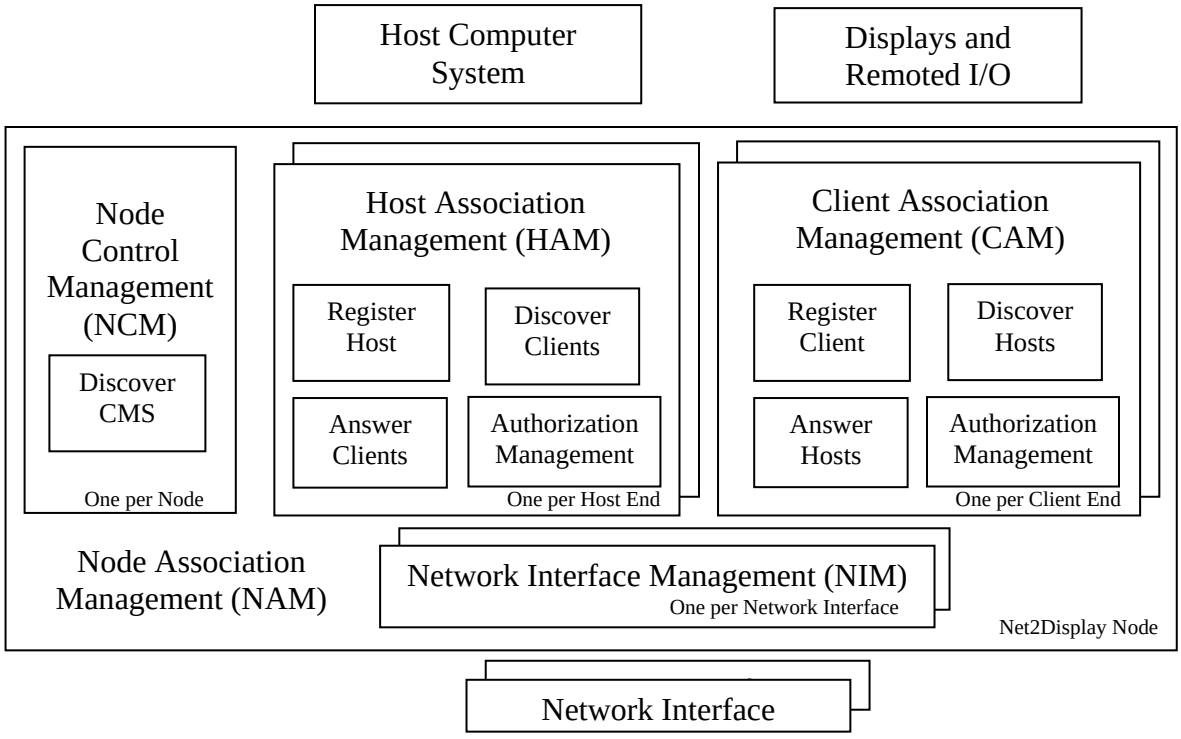


Figure 19-1: Node Management Block Diagram

The functions of the different NAM state machines are as follows:

- **Node Control Management (NCM)** coordinates the operation of a single Net2Display Node and maintains the Node-wide status and state.
    - **Node Initialization** initializes Net2Display Node state
    - **Discover CMS** uses the Discovery Processes defined by Net2Display to search for and locate Connection Management Servers
- **Host Association Management (HAM)** controls the operation of a Host End, the Host instance serving remote data to one Client. The HAM sets up the Host end of the Association. The HAM has a number of subprocesses that it uses to perform specific operations:
    - **Discover Clients** uses the Discovery Processes defined by Net2Display to search for available Client Services on the network.
    - **Register Host** registers the Host as a Net2Display Host Service provider with the Service Discovery Processes. The Host may register itself with a Connection Management Server or with another Discovery Process server.
    - **Answer Clients** responds to Client requests for Host Service
    - **Authorization Management** manages the authorization of the Host to the CMS, the authorization information provided by the Client and the state of the authorization for the Association. Authorization Management maintains Authorization status that can be used for the Automatic Reconnection of a dropped Association either over the original Network Interface or over another Network Interface in the Node.
- **Client Association Management (CAM)** controls the operation of a Client End, the Client instance displaying remote content from one Host. The CAM sets up the Client end of the Association. The CAM has a number of subprocesses that it uses to perform specific operations:
    - **Discover Host** uses the Discovery Processes defined by Net2Display to search for available Host Services on the network.
    - **Register Client** registers the Client as a Net2Display Client Service provider with the Service Discovery Processes. The Client may register its Client Service with a Connection Management Server or with another Discovery Process server.
    - **Answer Hosts** responds to Host requests for Client Service
    - **Authorization Management** manages the authorization of the Client to the CMS, the authorization information provided by the Client and the state of the authorization for the Association. Authorization Management maintains Authorization status that can be used for the Automatic Reconnection of a dropped Association either over the original Network Interface or over another Network Interface in the Node.
- **Network Interface Management (NIM)** works with each Network Interface to ensure that a unique IP address is assigned either through DHCP or using Autoconfiguration and verification of the uniqueness of the address.

The operation of Host and Client Ends are described in the following sections to show the full operation of all state machines involved in Host and Client instances. These sections are descriptive in nature and do not fully specify the state machine operation. The state machine behavior of the Host is specified in Section **Error! Reference source not found.**, while the state machine behavior of the Client is specified in Section 21.2.2.

19.3 NCM Services

This section describes the services provided by NAM and other entities within a Net2Display Node. This interface is used by a Net2Display Node to monitor and control the operation of the NAM entity. Additional information is provided in other sections of the Net2Display document concerning the conditions that invoke these primitives and the Net2Display Node actions upon the receipt of the generated primitives. The Node Control Management Interface Primitives are:

- Reset_Node - This primitive is presented to NAM to initialize the Net2Display Node or to reset the state machines to a known state. The receipt of this primitive causes NAM to take the following actions:

  1) Initialize all state machines, timers and other initialization parameters

  2) Issue a NCM_Reset signal to NCM

- Start_Host_End_Instance
- Start_Client_End_Instance

19.4 Facilities

The following facilities are used within NAM for maintaining state within a Net2Display Node. Facilities are variables, flags, signals and timers. Variables may take on more than two different values, while flags are a special type of variable constrained to take on either a one or a zero value. Flags are designated in this standard by a suffix of "_Flag." A signal is an instantaneous communication that causes an immediate transition in a cooperating state machine.

19.4.1 Variables

The following variables are used within NAM to communicate between state machines:

- **Input_Facilities**: Input_Facilities indicates when a keyboard, mouse, input tablet, keypad or other input facilities are attached to a Client that can allow it to provide user input and function in Active Mode. One instance of Input_Facilities exists in a Client System.
- **Host_Address_List(n):** One instance of the Host_Address_List(n) exists for each instance of the Discover Host State Machine.

19.4.1.1 NCM Variables

There is one instance of each of these NCM variables. The NCM variables are as follows:

- **CMS_Address_List(n):** One instance of the CMS_Address_List(n) exists for the Discover CMS state machine.
- **CMS_Address_Flag:** This flag indicates that the NCM has a CMS address. The CMS address was received either through user or administrator assignment or through the CMS Discovery process. The CMS address may be valid, not reachable or no longer valid.
- **No_CMS_Flag:** This flag indicates when the CMS Discovery process has completed and no CMS address is known.
- **CMS_Response_Flag:** This flag indicates when a response has been received from a CMS to indicate that a CMS is present

19.4.1.2 NIM Variables

There is one instance of each of these NIM variables for each NIM instance and associated Network Interface. The NIM variables provide state from the NIM state machine and direct the operation of the state machine. The NIM Variables are as follows:

- **Node_Address:** This flag indicates the IP Address acquired for this Network Interface. Each different Network Interface will have its own Node_Address.
- **DHCP_Address:** An IP address acquired from a DHCP server
- **Auto_Address:** An IP address attempted by the zero configuration process
- **Media_Flag:** This flag indicates that the network media is active and available for address assignment.
- **Valid_IP_Address_Flag:** This variable indicates whether a network interface that will be used by Net2Display for communication has a valid IP address. This variable is set by the NIM state machine and monitored by the Client Association State Machine and Host Association State Machine. One instance exists for each Network Attachment State Machine.

19.4.2 Signals

A signal is an instantaneous communication that causes an immediate transition in a cooperating state machine. The following signals are used to communicate between the different cooperating state machines:

- **NCM_Reset**: Signal to NCM to initially start up the Net2Display instance or to reset it to the initial state
- **NIM_Reset**: Signal from the NCM state machine to startup or reset the NIM state machine
- **HAM_Reset**: Signal from the NCM state machine to startup or reset a HAM state machine
- **CAM_Reset**: Signal from the NCM state machine to startup or reset a CAM state machine

19.4.3 Timers

19.4.3.1 Timer Functions

19.4.3.2 Timer Expiration Values

The following timer expiration values ranges are specified to provide interoperability.

- **ARP_Timeout:** The time to wait after sending an ARP packet to provide a high confidence that no response will be received.
- **DHCP_Timeout:** The total time to wait for a DHCP response from when the DHCP_Discover request is sent

19.5 Node Control Management (NCM)

There is one instance of the Node Control State Machine that runs for each Net2Display Node. The Node Control Management State Machine is shown in Figure 19-2.

19.5.1 NCM Detailed Description

General design is:

1) NCM state machine starts up on NCM_Reset signal to startup with argument specifying to start up Host or Client instance

2) NCM state machine starts up the NIM state machines with the NIM_Reset signal and waits until Network_Interface_Ready is asserted.

3) Network Interface State Machine looks at the status of the networking interface and starts it up or assists in acquiring an address or name if necessary.

    a. If DHCP is enabled, then use DHCP

    b. If a IP address was assigned, then use that address

    c. Otherwise, self assign an IP address

4) Network Interface State Machine signals the Node Control State Machine with Network_Interface_Ready when the network interface is ready to use.

5) Node Control State Machine then starts up Host or Client State Machine as requested

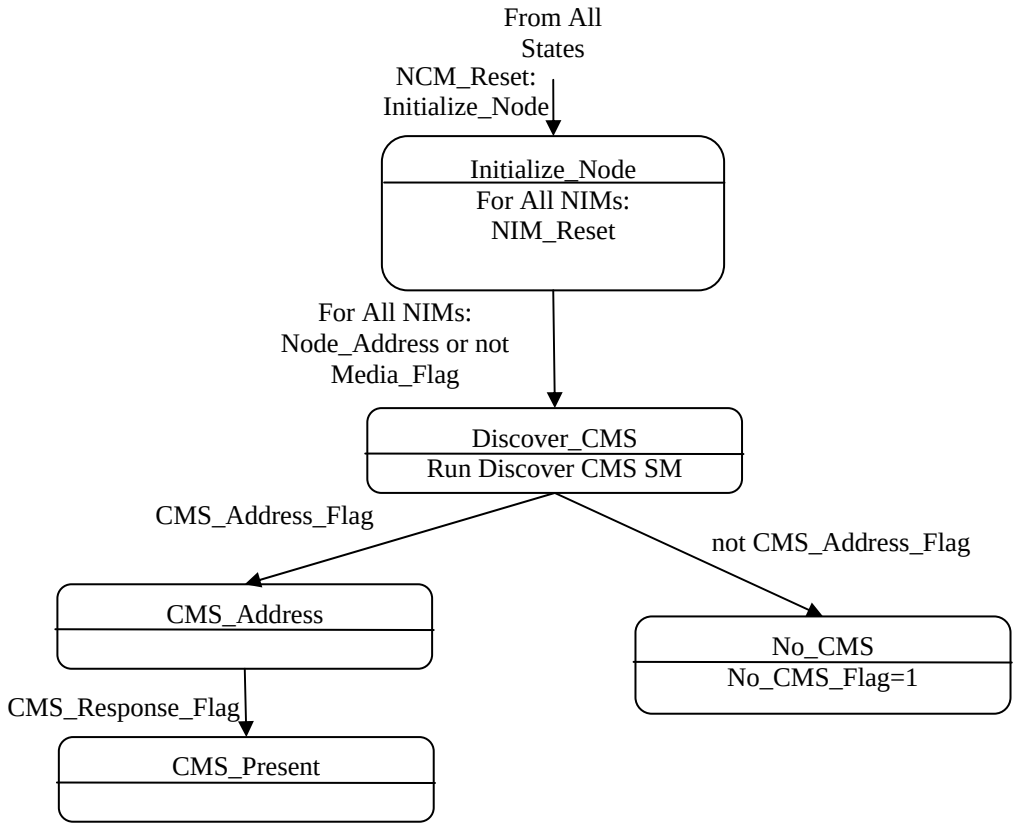


Figure 19-2: Node Control Management (NCM) State Machine

19.5.1.1 General Transitions

The following transitions may be made from any state:

- **NCM_Reset:** A transition is made to the Initialize_Node state whenever the NCM_Reset signal is asserted. The Initialize_Node actions are taken on this transition.

19.5.2 Discover CMS

Discover_CMS uses available Discovery mechanisms on available Network Interfaces to provide a list of CMS Servers. The Discover_CMS starts when requested by a Host or Client and completes when all the

Discovery processes have received responses, have timed out or a Reset is issued. The different approaches used in Discover_CMS are shown in Figure 19-3.

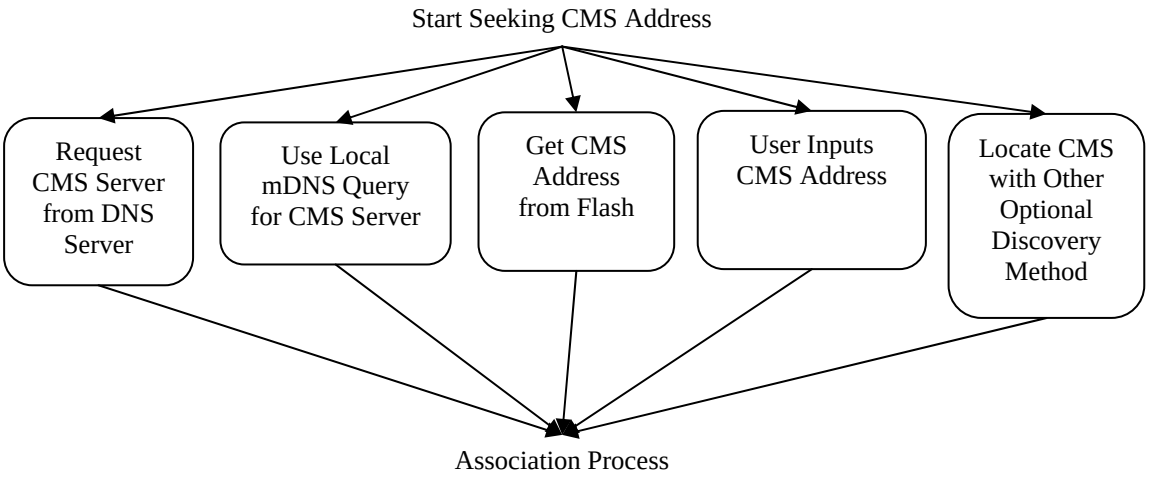


Figure 19-3: High Level CMS Discovery Process

The different CMS Discovery methods can be grouped into three different approaches: DNS-based, implementation dependent and optional. The DNS-based approaches are using DNS Services and using mDNS. The implementation dependent approaches include getting the CMS Address from flash and the user input of the CMS Address. The optional CMS discovery methods include using SLP V2, LDAP, DNS Lookup on Last_CMS or DNS Lookup on cms.mysubnet.

When DNS Services are requested, the approach described in Section 4.3.5.2 is used. When Multicast DNS (mDNS) is used for locating the CMS Server, the approach described in Section 4.3.5.1 is used. The implementation dependent and optional approaches are not further defined in this standard as they are specific to implemenations or are defined in other standards.

19.6 Network Interface Management (NIM)

The NIM state machine performs the service of acquiring an address for a Net2Display Node, either a Client or a Host. When a Net2Display Client or Host joins a network, it needs to acquire an IP address to participate in the network. The general procedure is that DHCP is first tried and used available, but if not the Node will assign itself an address.

Use DHCP or Autoconfiguration to obtain the address. This is done for each Network adapter separately at initialization. This may be a completely separate state machine that is done at initialization, although Net2Display should wait for an address before using a network interface.

There is one instance of the NIM state machine for each Network Interface being used by Net2Display Remoting. The Network Interface state machine must support IETF RFC 3927 Dynamic Configuration of IPv4 Link-Local Addresses and IETF RFC 2464 IPv6 Stateless Address Autoconfiguration. The Network Interface State Machine must work with network interfaces that are being used for other communication as well. The Network Interface state machine communicates with the NIC and provides a single interface to the networking.

- The Network Interface State Machine is started on receipt of the Network_Interface_Initialize signal
- The Network Interface State Machine queries the IP address and network host name and if they are both present returns the Network_Interface_Ready signal to the Node Control State Machine

- If there is not an IP address, the Network Interface State Machine checks if there is a DHCP server, if so, it requests and address from the DHCP server.
- If there is no DHCP server, the NISM assigns a random IP address and performs an ARP to see if the address is already being used. If the address is already being used, NISM will select another IP address and use ARP to see if that address is already in use. This is continued until an IP address is assigned and the IP_Address flag is then set.

19.6.1 Manual Address Assignment

A Net2Display Node may allow its IP address to be manually assigned. The manual assignment of an IP address is outside the scope of this standard.

19.6.2 Dynamic Host Configuration Protocol (DHCP)

DHCP provides address assignment when a DHCP server is present within the network infrastructure. The DHCP protocol is specified in IETF RFC 2131 and should be implemented according to that specification.

19.6.3 No DHCP Present

Both IPv4 and IPv6 have standard ways of choosing IP addresses without help from DHCP. By RFC 3927, IPv4 uses the 169.254.* (link-local) set of addresses. The technique for IPv4 is called *IPv4 Link-Local* (IPV4LL) in the RFC, however Microsoft refers to this as *Automatic Private IP Addressing* (APIPA) or *Internet Protocol Automatic Configuration* (IPAC). For IPv6, see RFC 2462.

19.6.4 NIM Detailed Description

The Node Interface Management (NIM) State Machine is shown in Figure 19-4.

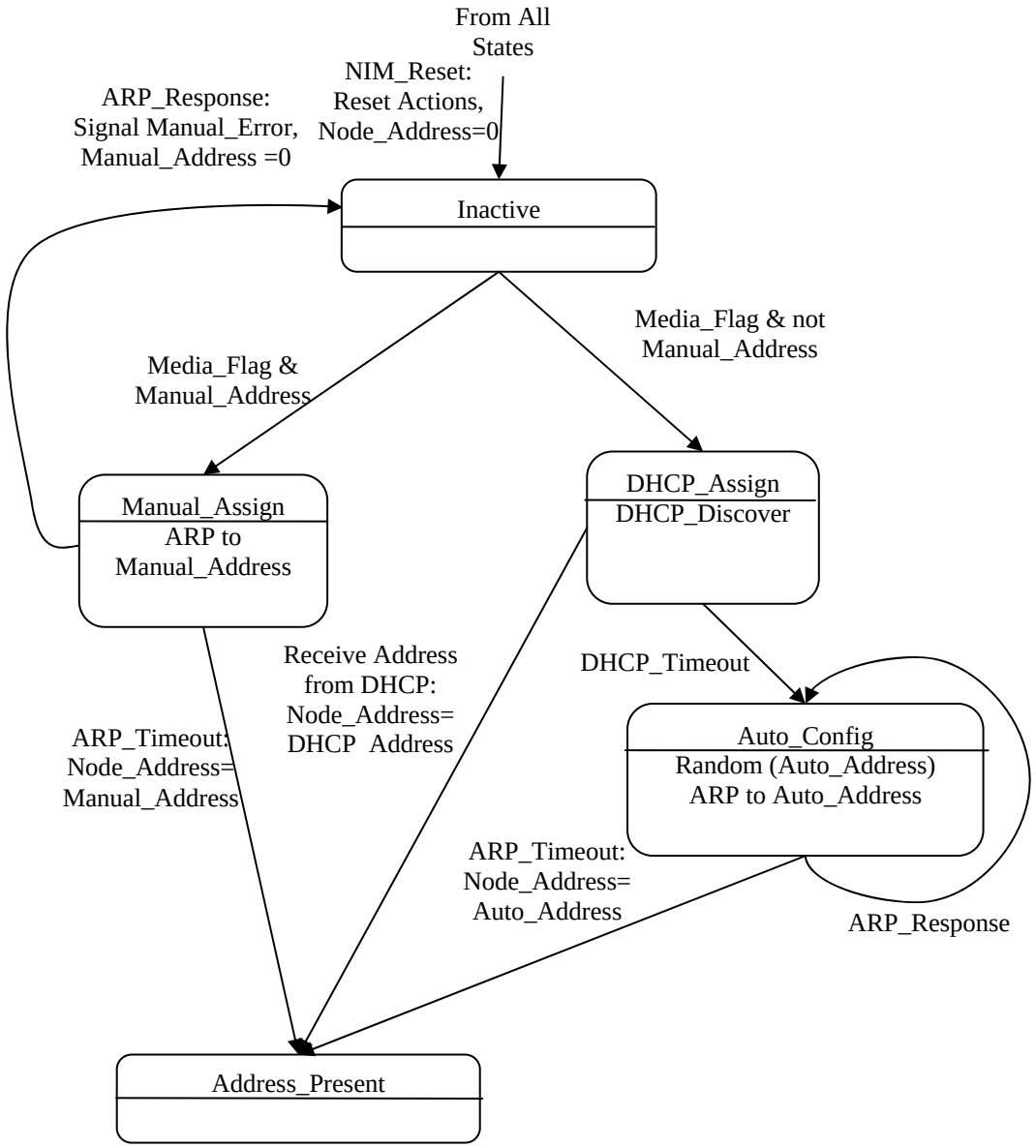


Figure 19-4: Network Interface Management (NIM) State Machine

20 Host Operation

Note: Net2Display state machines define the detailed behavior of Net2Display Nodes which will be more completely characterized in the next version of this standard after a reference design has been implemented.

The high level operational behavior of the Host is described in this section to give a general overview of the relationships of the different NAM state machines in operation. The initialization of the Associations between Clients and Hosts can be broken into major stages that happen at the Host as an Association is formed. The following stages occur for the first Host initializing in a Node.

- **Node Initialization by NCM** – The initialization of the Net2Display state machines occurs first if they have not already been initialized.
- **Network Initialization by NIM –** The Network Interfaces within the Node are initialized if they were not already started. If the Network Interfaces are initialized, DHCP or Autoconfiguration are used to assign and IP Address.
- **Discover CMS by NCM** – The Net2Display Node attempts to locate a CMS address using all available Network Interfaces if the Node does not already have a current CMS address. This stage is exited when a CMS address is present or timeouts occur in the discovery of the CMS.
- **Host Association Management by HAM** –
    - **Wait for Client** – The Net2Display Host waits for a Client to request a connection or for an internal request for it to inform a Client to request a connection
    - **Locate Client** – Starts when a Net2Display Node begins seeking an address and ends when the Net2Display Node has one or more Client Addresses for prospective Associations. The Host makes itself available for Association with a Client using one or more of the following approaches in parallel:
        - Registering its Host Service with a CMS Server
        - Registering its Host Service with a DNS Server
        - Responding to Host Service Requests sent using nDNS
        - Having a Host Address that is known by a User, Client or CMS
    - **Association Process** – The Association Process begins when the Net2Display Client has a Host address. If the Association Process fails in its Association attempt with the Host, either the Association Process is tried on another Host address passed from the Discovery Process or the Discovery Process is reentered to acquire another Host address. In the Association Process the Client requests and Association with a Host, authenticates with the Host (either directly or via CMS) and then exchanges Configuration Information with the Host. The Association Process ends when an Association with a Host is formed.
    - **Association Management** – Once an association is formed, Association Management maintains the Association.

A high level view of the Host initialization process is shown in Figure 20-1.

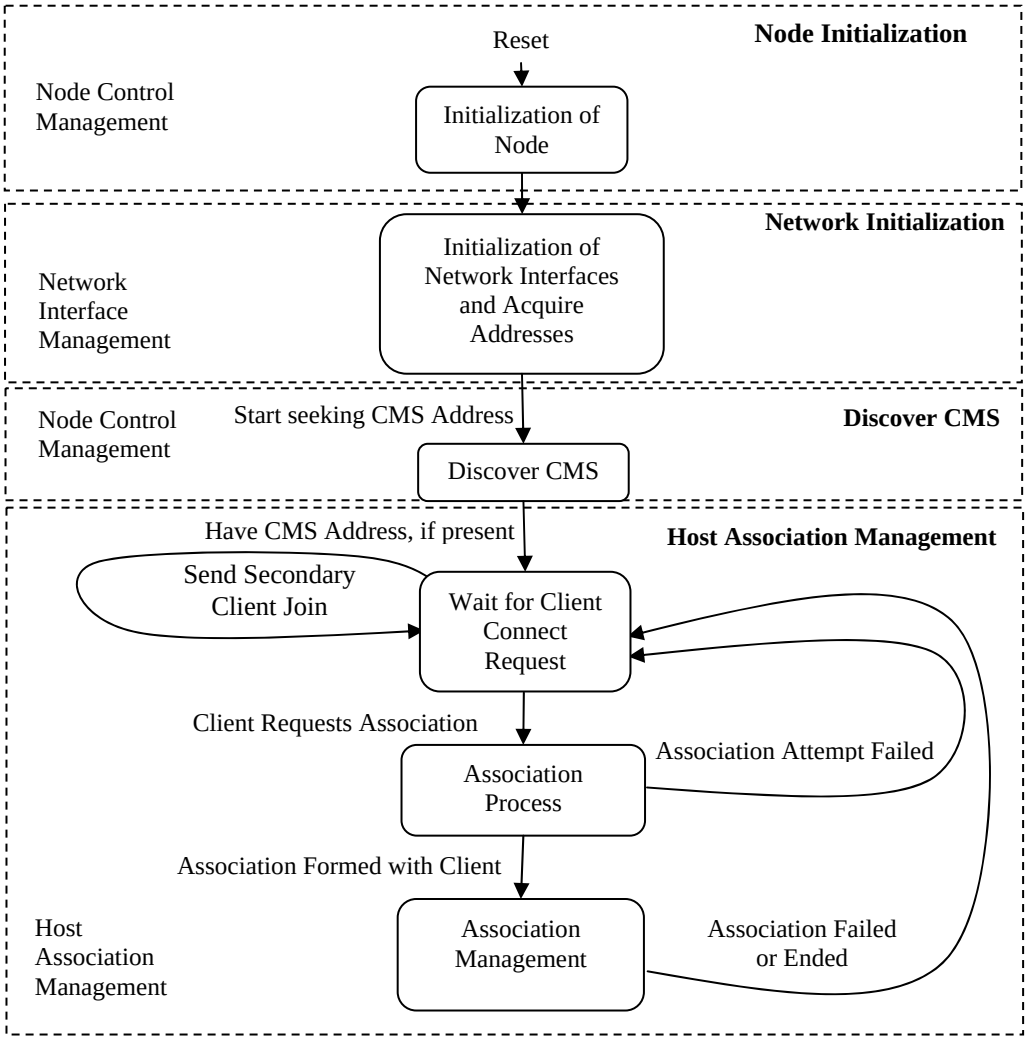


Figure 20-1: Overall Host Operation

20.1 HAM Interface

The Host Interface describes interactions that normally take place between the Host Server computer and the Net2Display Host Node. This interface is conceptual and does not have to be explicitly implemented for compliance. The HAM Interface Primitives are:

- Client_Services_Query - A Host Server may quesry to locate Net2Display Client Services that are available on the network. The Client_Services_Query returns a list of the Net2Display Clients in Passive Mode that are offering their Client Services and the IP addresses for each. A Host Server may seek Client Services for when the Host Server detects that no display is attached directly to a Display Connector or when the user wishes to add an additional display to his display configuration
- Client_Services_Request - A Host Server may request access to a networked Net2Display Client device. The Host may request access by using a Client Address in the form of an IP Address, an IP host name or a IP subnet identifier
- Host_Status_Indication – The Host Server may request the status of the Net2Display Host instance. The status may include information such as whether an Assoication is present and the Client to whom the Host is Associated.

20.2 Host Association Management (HAM) State Machine

Within a Net2Display Host, there is one HAM state machine for each Association that the Host forms with a Client. Thus, if a Host is only capable of associating with a single Client at one time, then there will only be one instance of the Host Association State Machine. A Host capable of associating with two different Clients simultaneously will have two instances of the Host Association State Machine.

The Host Association State Machine state diagram is shown in Figure 20-2, a high level description of the state machine is given in this section and detailed descriptions of each state are given in the following sections. If there are any discrepancies or differences between these different descriptions, the Host Association State Machine state diagram, shown in Figure 20-2, takes precedence.

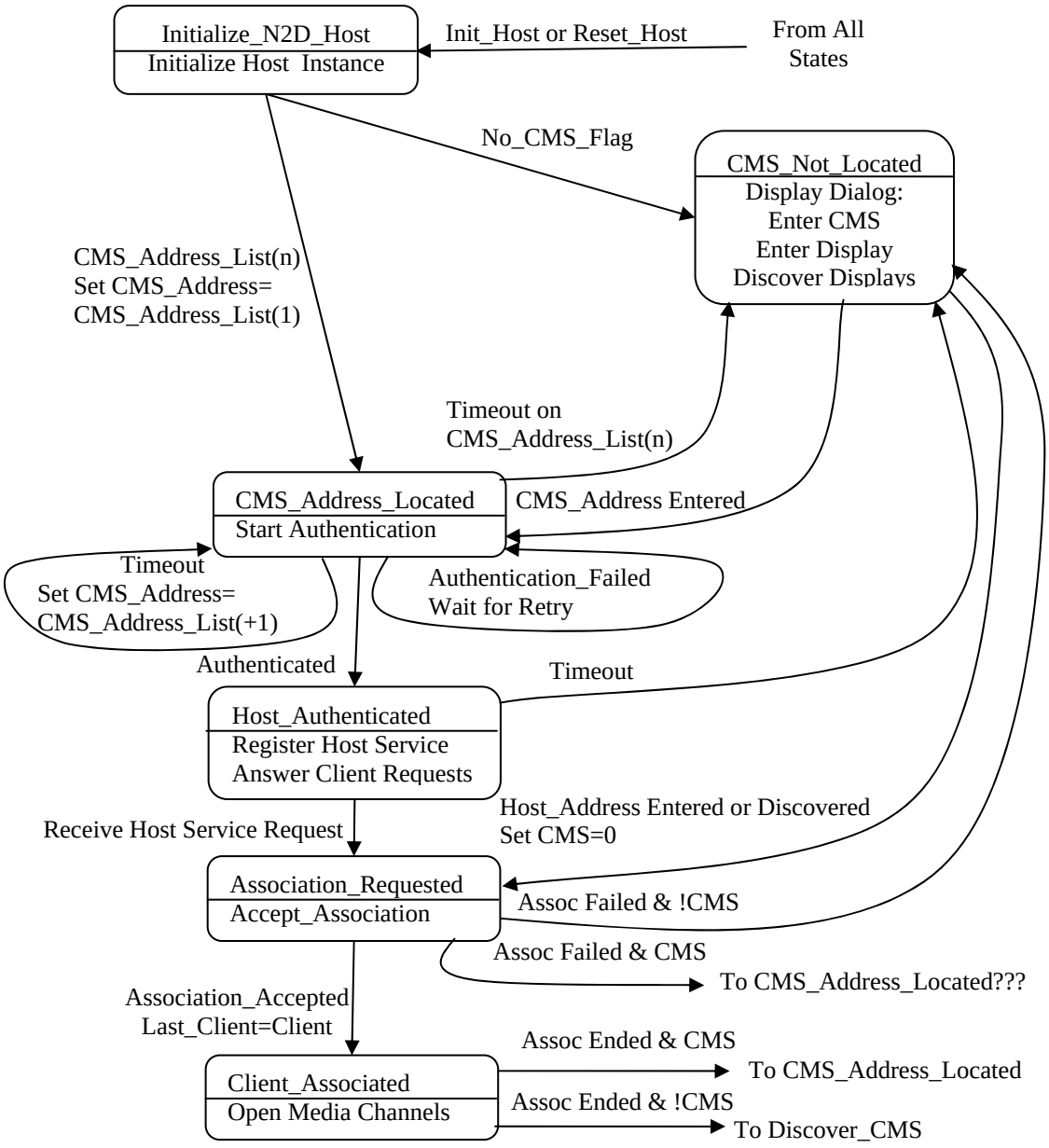


Figure 20-2: Host Association State Machine

A high level description of the Host Association State Machine is:

1) Initialize_N2D_Host: Host Initializes
2) Acquire_Address: Host acquires a numeric IP address
3) Discover_CMS: Host searches to discover Connection Management Server(CMS)
4) Authenticate Host with CMS
    a) What does this require
5) Registration of Host Service with:
    a) DNS SD
    b) CMS
6) Wait for Client to Associate
7) Associate with Client
8) Request Passive Displays to be Associated

20.2.1 HAM Functional Description

20.2.1.1 Initialization Sequence

Host Server determines if Net2Display Host Node will offer Host Services or seek Client Services. Notification of the network configuration is given to the Host Server to aid in decisions of services to seek or offer. Recommendations and Examples of policies for offering Host Services or seeking Client Services are given in an Appendix.

The Host specific steps are given below.

1) Optionally register the Host Services with a Connection Management Server in an implementation specific manner.
2) Wait for a Client requesting an Association or request Client Services if no Display is attached to the Display Connector
3) If a Connection Request is received from a Client, authenticate the Client Device and User
4) Exchange Configuration Information with Client
5) Associate with the Client
6) Optionally, at the initiation of the Host Server, request and Associate with any additional Passive Displays

20.3 Register Host

Register_Host is run by a Host to register the Host capability with CMS and/or other Service Discovery means on the network. The Register_Host process starts when requested by a Host and completes when all the registration methods have competed or timed out or a Reset is issued.

20.3.1 Discover Clients

Discover_Display uses available Discovery mechanisms on available Network Interfaces to provide a list of Displays. Discover_Clients starts when requested by a Host and completes when all the Discovery processes have received responses, have timed out or a Reset is issued.

20.3.1.1 Discover Clients Functional Description

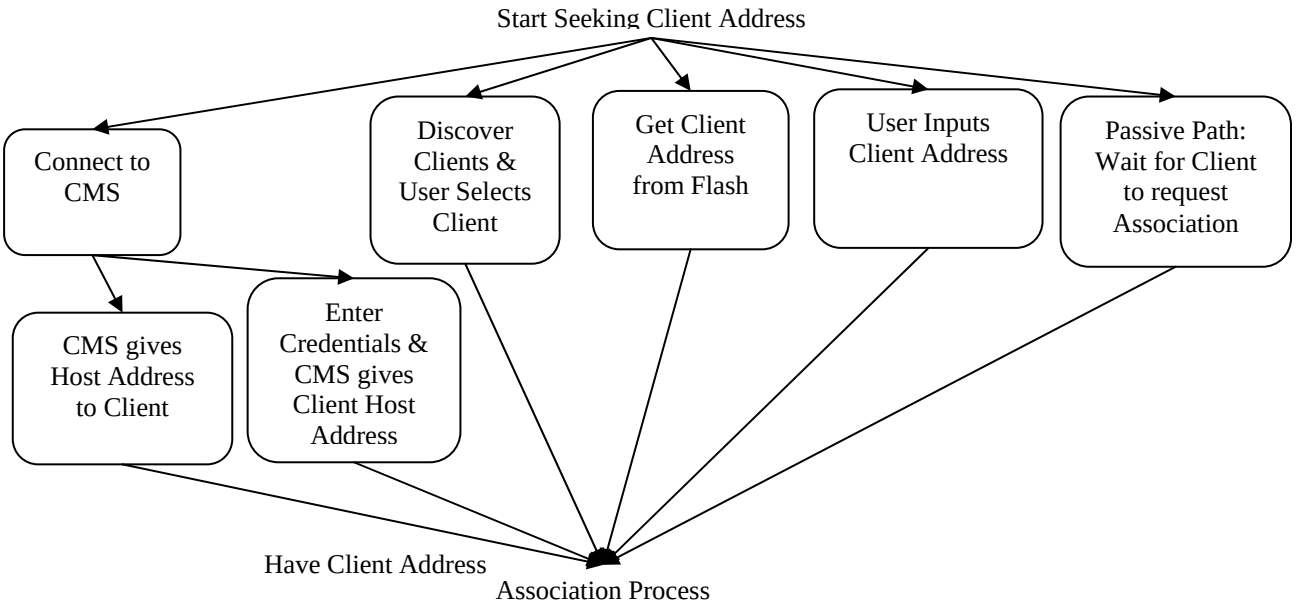


Figure 20-3: High Level Discover Clients Process

20.3.1.2 Discover Clients Detailed Description

20.3.2 Answer Clients

Answer_Clients is a process where a Host will respond to queries for service from Clients. These queries may come via mDNS or other local Discovery means. The Answer_Clients state is entered when requested by a Host and is exited when a Client requests an Association or the Host is reset.

20.4 Association Process

20.4.1 Host Association Process Functional Description

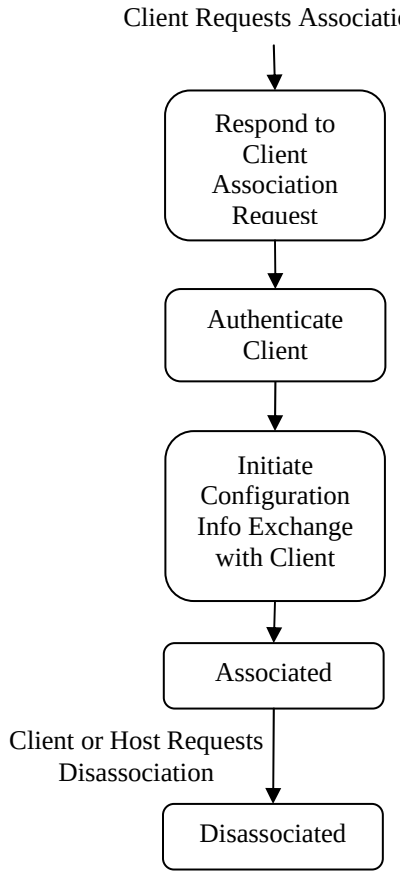


Figure 20-4: Host Association Process

20.5 Authorization Management

20.5.1 Automatic Reconnection

Authorization Management maintains Authorization status that can be used for the Automatic Reconnection of a dropped Association either over the original Network Interface or over another Network Interface in the Node.

20.6 Association Processing and Management

Association Processing

21 Client Operation

Note: Net2Display state machines define the detailed behavior of Net2Display Nodes which will be more completely characterized in the next version of this standard after a reference design has been implemented.

The operation of the Associations between Clients and Hosts can be broken into stages that happen at the Client as an Association is formed. The following stages occur for the Client:

- **Node Initialization by NCM** – The initialization of the Net2Display state machines first if they have not already been initialized.
- **Network Initialization by NIM –** The Network Interfaces within the Node are initialized if they were not already started. If the Network Interfaces are initialized, DHCP or Autoconfiguration are used to assign and IP Address.
- **Discover CMS by NCM** – The Net2Display Node attempts to locate a CMS address using all available Network Interfaces if the Node does not already have a current CMS address. This stage is exited when a CMS address is present or timeouts occur in the discovery of the CMS.
- **Client Association Management by CAM**
    - **Wait for Client Connect Request –** The Client waits until it is directed to seek a connection to a Host
    - **Discover Hosts** – Starts when a Net2Display Node begins seeking an address and ends when the Net2Display Node has one or more Host Addresses for prospective Associations. Discover Hosts uses one or more of the following approaches in parallel:
        - Client requests and receives a Host Address from CMS
        - User enters Credentials and CMS gives Client Host Address
        - Client Runs Host Discovery and User selects a Host
        - Get the Host Address from local Flash Storage
        - User inputs Host Address on Client
        - Passive Path: Client Registers Service with CMS and responds to Client Service Requests and receives a Host Address across the network
    - **Association Process** – The Association Process begins when the Net2Display Client has a Host address. If the Association Process fails in its Association attempt with the Host, either the Association Process is tried on another Host address passed from the Discovery Process or the Discovery Process is reentered to acquire another Host address. In the Association Process the Client requests and Association with a Host, authenticates with the Host (either directly or via CMS) and then exchanges Configuration Information with the Host. The Association Process ends when an Association with a Host is formed.
    - **Association Management** – Once an association is formed, Association Management maintains the Association.

A high level view of the Client initialization process is shown in Figure 21-1.

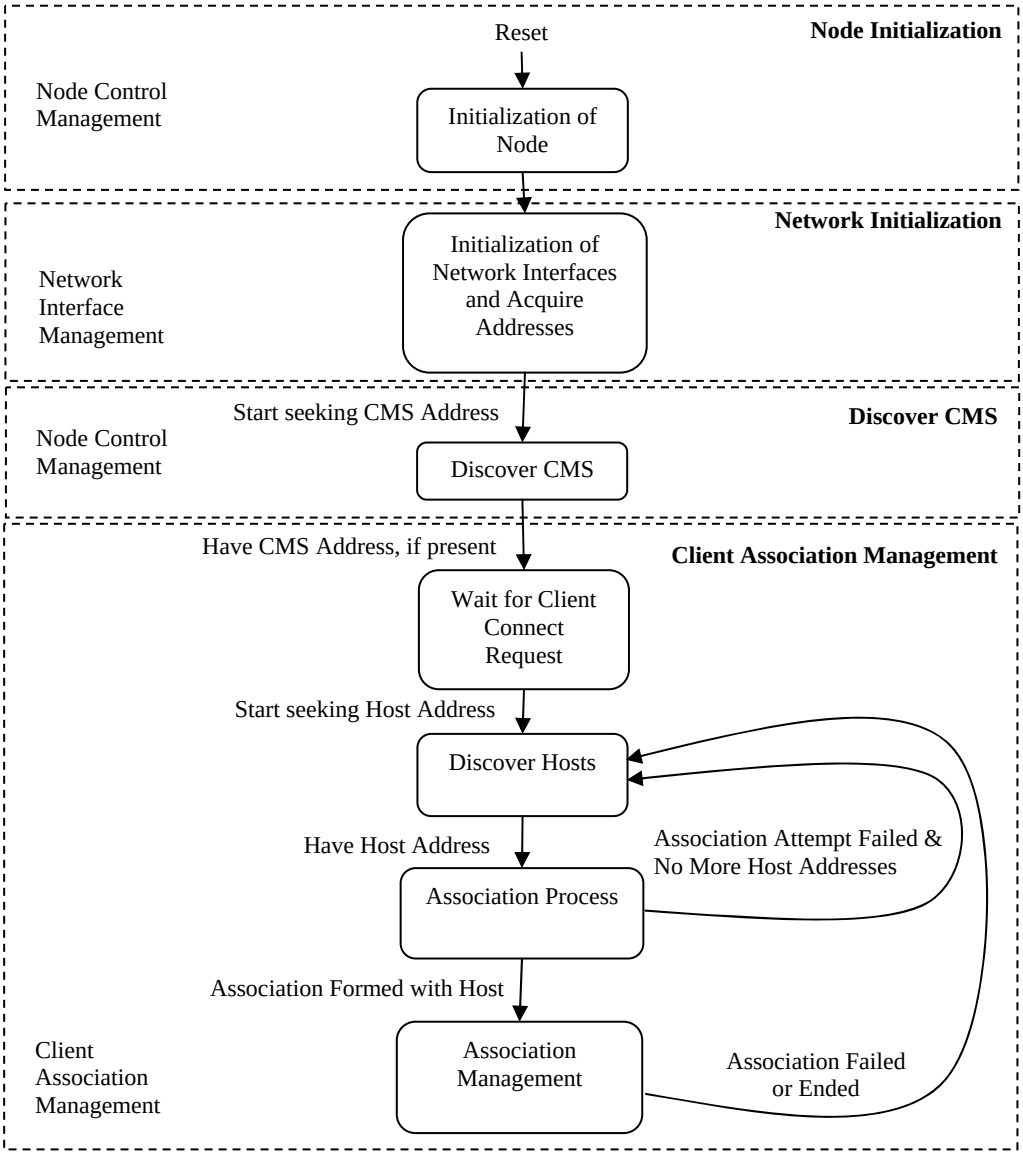


Figure 21-1: Overall Client Operation

21.1 Client Interface

The Client Interface describes interactions that normally take place between the user and the Net2Display Client. This interface is conceptual and does not have to be explicitly implemented for compliance. The CAM Interface Primitives are:

- Host_Services_Request – The Host_Services_Request is used to request Host Services for a Client. The Host_Services_Request contains an address of the remote Host that should be Associated with this Client instance. The Host_Services_Request may include authentication input as a parameter of the request for the authentication of the user or Client System
- Client_Status_Indication – The Client System may request the status of the Net2Display Client instance. The status may include information such as whether an Assoication is present and the Host to whom the Client is Associated.

21.2 Client Association Management (CAM) State Machine

Within a Net2Display Client, there is one Client Association State Machine for each Association that a Client forms with a Host. Thus, if a Client is only capable of associating with a single Host at one time, then there will only be one instance of the Client Association State Machine. A Client capable of associating with two different Hosts simultaneously will have two instances of the Client Association State Machine.

The Client State Machine must work for:

- Passive and Active Clients
- Presence and absence of DHCP and DNS
- Software Client Implementation

The Client Connection State Machine is made up of several cooperating state machines that perform specific operations. These include:

- Client State Machine – receives the startup signal and controls the overall state of the Client
- Network Interface State Machine – sets up network interface, acquiring address and name, identifies if DHCP and DNS are present.
- Active Mode State Machine – tracks whether Client is in Active or Passive Mode, (Does this include responding to Client Services Requests)
- Discovery State Machine – Generates and receives requests for discovery
- Client Services State Machine – Is this in the Active Mode State Machine or separate

21.2.1.1 Initialization Sequence

The Client Association State Machine state diagram is shown in Figure 21-2, a high level description of the state machine is given in this section and detailed descriptions of each state are given in the following sections. If there are any discrepancies or differences between these different descriptions, the Client Association State Machine state diagram, shown in Figure 21-2, takes precedence.

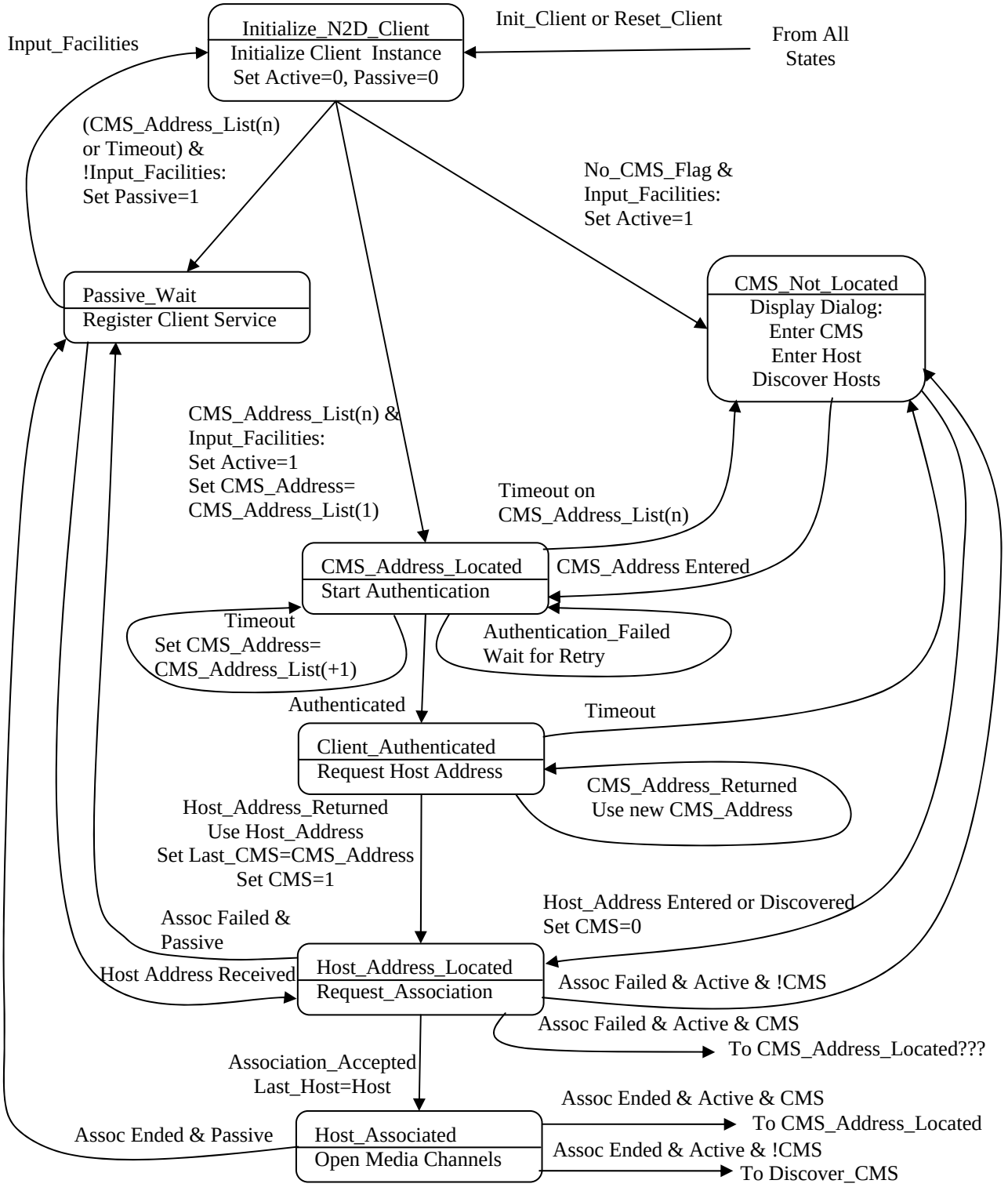


Figure 21-2: Client Association State Machine

A high level description of the Client Association State Machine is:

1) On startup, Client Determines if it is Active (Input Facilities attached) or Passive (Display Only)

2) CMS_Address_Located
    a) Client waits in this state until authentication is started
    b) If Authentication fails, wait for a retry
    c) If a timeout occurs when communicating to the CMS, try CMS_Address_List(+1)
    d) If a timeout occurs on CMS_Address_List(n), the last address, go to CMS_Not_Located
    e) If Authentication is successful, go to Client_Authenticated
3) Client_Authenticated: User and optionally, Client device, have been authenticated
    a) Request Host Address from CMS
    b) When CMS gives response, Set CMS_Flag=1 and save Last_CMS=CMS address
    c) If CMS returns name of another CMS, Reenter Authenticated state with the new CMS address
    d) If CMS returns a Host address, go to Host_Address_Located
    e) If timeout, go to CMS_Not_Located
4) CMS_Not_Located: If Client is Active and CMS not found, Give option to enter CMS address, enter Host address or to discover a Host
    a) If CMS is entered, go to Discover_CMS
    b) If Host address is entered, go to Host_Address_Located
    c) If discover a Host is selected, Attempt to locate available Hosts using the following Service Discovery approaches in parallel
        i) SLP V2 (RFC 2608)
        ii) DNS-SD (RFC 2782)
        iii) mDNS (multicast DNS)
    d) If only one Host is located go to Host_Address_Located
    e) If multiple Hosts are located, present a selection list of the first 10
        i) After User selects a Host, go to Host_Address_Located
5) Passive_Wait: If Client is Passive, wait until Host Address is received for connection, then go to Host_Address_Located
6) Host_Address_Located: Request to Associate to located Host
    a) If Association fails, Client is Active and CMS is present, notify CMS of connection error and request other Host_Address
    b) If Association fails, Client is Active and no CMS is present, go to CMS_Not_Located
    c) If Association fails and Client is Passive, display error and return to Passive_Wait
    d) If Association succeeds, Complete association formation and setup Media Session
7) Host Associated: Association Complete

21.2.2 CAM Detailed Description

21.2.2.1 Initialize_N2D_Client

The first state that a Net2Disply Client instance enters on initialization is the Initialize_N2D_Client state. Upon entry into this state, all of the internal state machine variables are initialized to zero. The transitions from this state are:

- **Init_Client or Reset_Client:** If the Net2Display Client Node or End is startup or reset, this transition causes the state machine to restart from all states
- **CMS_Address_List(n) & Input_Facilities:** A transition to the CMS_Address_Located state is taken if CMS Addresses are returned from the CMS Discovery process. The Active flag is set to one on this transition since Input_Facilities are present
- **(CMS_Address_List(n) or Timeout) & !Input_Facilities:** A transition is made to the Passive_Wait state if no input facilities are connected to the Client and either CMS Addresses are returned or a Timeout occurs. The Passive flag is set on this transition.
- **Timeout & Input_Facilities:** If Input Facilities are present and a timeout occurs, the Active flag is set to indicate Active Mode and a transition is mad eot the CMS_Not_Located state.

21.2.2.2 CMS_Address_Located

On transition to this state, user authentication is started with the Connection Management Server. Optionally, Client device authentication is started with the Connection Management Server as well.

Transitions from this state are:

- **Authenticated:** the state machines transitions to the Client_Authenticated state when the Authenticated signal is received, indicating that the user and optionally the Client Device have been authenticated
- **Authentication_Failed:** When the user or Client device authentication fails, the Authentication_Failed signal is asserted and the state machine renters the CMS_Address_Located State to wait for the reentry of proper authentication information.
- **Timeout on CMS_Address_List(n):** When a timeout occurs on attempting authentication with the last Connection Management Server address given in the CMS_Address_List, a transition is made to the CMS_Not_Located state.
- **Timeout**: When a timeout occurs in the CMS_Address_Located state and all of the addresses in the CMS_Address_List have not been attempted, the CMS_Address_Located State is reentered and an authentication is attempted with the next address in the CMS_Address_List list.

21.2.2.3 Client_Authenticated

In this state, the user has been authenticated to the Connection Management Server and the Client has requested a Host address from the Connection Management Server and is awaiting the Host address. Transitions from this state are:

- **CMS_Address_Returned:** If the Host address is requested and the Connection Management Server returns the address of another Connection Management Server, the state machine returns to the Client Authenticated state and places a request for a Host address to the newly received CMS Address.
- **Host_Address_Returned**: If a Host address is returned, the state machine trasistions to the Host_Address_Located state, sets the CMS flag to indicate that a Connection Management Server is

being used and sets Last_CMS to CMS_Address to use in CMS discovery in future Discovery processes.

- **Timeout**: If a Client is authenticated and times out waiting for a valid Host address, it transitions to the CMS_Not_Located state to await the entry of a functioning Connection Management Server or a Host address.

21.2.2.4 CMS_Not_Located

The CMS_Not_Located state is entered when no Connection Management Server is found during the CMS discovery process or attempts to locate a Connection Management Server fail.

Transitions from this state are:

- **CMS_Address_Entered:** When a CMS address is entered while in the CMS_Not_Located state, a transition is made to the CMS_Address_Located state
- **Host_Address Entered or Discovered:** When a Host Address is entered by a user or discovered in the CMS_Not_Located state, the sate machine transitions to the Host_Address_Located state and sets the CMS flag to zero to indicate that a Connection Management Server was not used in the process

21.2.2.5 Passive_Wait

When a Client enters the Passive Wait state, it registers the Client Service with Discovery Servers and Answers Host requests for its Client Services.

Transitions from this state are:

- **Input_Facilities**: If a Client is waiting in the Passive_Wait state and Input Facilities are connected to the Client, the Client transitions to the Discover CMS state to restart in Active Mode.
- **Host_Address_Received**: When a Client waiting in the Passive_Wait state receives a Host address for connection, it transitions to the Host_Address_Located state where it requests an Association.

21.2.2.6 Host_Address_Located

Transitions from this state are:

- **Association_Accepted:** This transition occurs when the Host and Client complete the Association process and the Host address is saved in the variable Last_Host for future connection attempts.
- **Assoc Failed & Passive:** If the Association attempt fails and the Client is operating in Passive Mode, the Client returns to the Passive_Wait state to await another Host address.
- **Assoc Failed & Active & !CMS:** If the Association attempt fails and the Client is and active Mode and a Connection Management Server was not located, a transition is made to the CMS_Not_Located state.
- **Assoc Failed & Active & CMS:** If the Association attempt fails and the Client is and active Mode and a Connection Management Server was located, a transition is made to the CMS_Address_Located state.

21.2.2.7 Host_Associated

This state is entered on the formation of an Association between the Client and a Host.

Transitions from this state are:

- **Assoc Ended & Passive:** If the Association ends and the Client is operating in Passive Mode, the Client returns to the Passive_Wait state to await another Host address.

- **Assoc Ended & Active & !CMS:** If the Association ends and the Client is and active Mode and a Connection Management Server was not located, a transition is made to the Discover_CMS state to recheck that a Connection Management Server has not been added to the network since the beginning of the last Association.
- **Assoc Ended & Active & CMS:** If the Association ends and the Client is and active Mode and a Connection Management Server was located, a transition is made to the CMS_Address_Located state.

21.2.3 Register Client

Register_Client is run by a Client to register the Client capability with CMS and/or other Service Discovery means on the network. The Register_Client process starts when requested by a Client and completes when all the registration methods have competed or timed out or a Reset is issued.

21.2.4 Discover Hosts

Discover_Host uses available Discovery mechanisms on available Network Interfaces to provide a list of Hosts. Discover_Host starts when requested by a Client and completes when all the Discovery processes have received responses, have timed out or a Reset is issued.

21.2.4.1 Discover Hosts Functional Description

The different Discovery approaches are shown in Figure 21-3.

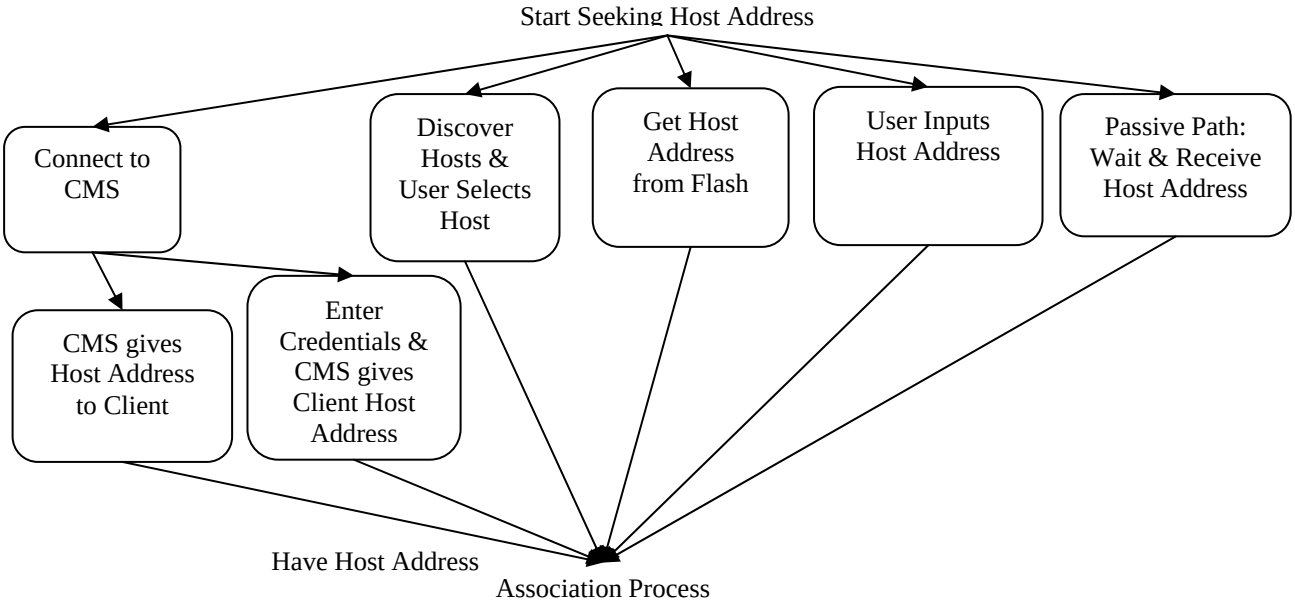


Figure 21-3: High Level Client Discovery Process to get Host Address

21.2.5 Answer Hosts

Answer_Hosts is a process where a Client will respond to queries for service from Hosts. These queries may come via mDNS or other local Discovery means. The Answer_Hosts state is entered when requested by a Client and is exited when a Host address is received for making an Association or a Reset is issued.

21.3 Association Process

21.3.1 Client Association Process Functional Description

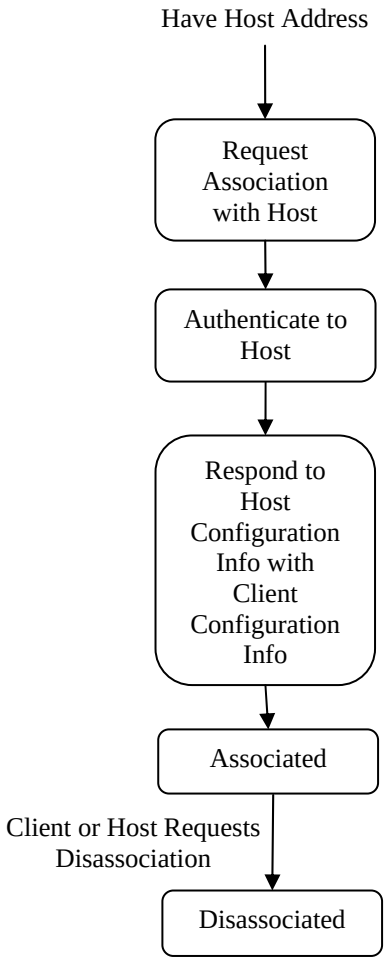


Figure 21-4: Client Association Process

22 Configuration Management

Note: Host and Client configurations and XML descriptions are best developed with a real implementation and will be defined as part of the reference design and specified in the next version of this standard.

22.1 Different Types of Configuration Information

There are several different types of configuration information that need to be exchanged for the formation of a Net2Display Association. They may need to be exchanged at different stages in the Association formation. They are:

- Node (Host or Client) Information – pertains to the overall Net2Display Client or Host Node Instance. Exchanged on formation of the Association. Includes
    - Net2Display Version number
    - Vendor Identification
    - Network Interfaces
- Association Information – Information pertaining to the particular Host-Client Association relationship. Exchanged on the formation of the Association.
    - Number and Types of Virtual Channels
- Virtual Channel General – Setup using Control PDUs. Exchanged on the Formation of the Virtual Channel
    - Bandwidth Allocated for Virtual Channel
    - Transport for Virtual Channel
- Virtual Channel Specific Information - Channel Specific information, such as Keyboard type, display type. Exchanged on the formation of the Virtual Channel
    - Keyboard Type

22.1.1 Exchange of Configuration Information

Two Exchanges of Information:

- Exchange on Association Formation
- Exchange on Virtual Channel Formation

22.2 Association Configuration Management

Configuration Information is the information contained within Net2Display Hosts and Clients that expresses the Node's capabilities. The exchange of this information between Clients and Hosts is necessary for the highest performance and use of optional capabilities. This information is exchanged at the formation of an Association between a Net2Display Host and a Net2Display Client.

Net2Display Configuration information formats and exchange are expressed using the Web Services Description Language (WSDL), which uses XML Schema to express the data formats. This approach allows the information to be expressed in a precise extensible format that is readily transferred to an implementation. For a description of WSDL, see Web Services Description Language (WSDL) Version 2.0 Part 0: Primer, and for a description of XML Schema, see XML Schema Part 0: Primer.

The XML Schema defines the rules that the XML data must follow. Example XML descriptions for the following variety of Net2Display implementations and configurations are in Appendix D:

- Integrated Display Client with Net2Display interface

- Net2Display Hub Client supporting one or multiple displays
- Net2Display Client Application running on a conventional Client PC
- Net2Display Host supported by hardware remoting
- Net2Display Host supported by software remoting
- Net2Display Host running on a virtualized system sharing network adaptors with other instances

Note: Host and Client configuration representation and communication are part of an implementation and will be defined as part of the reference design and specified in the next version of this standard

22.2.1 Net2Display Descriptors

Net2Display Node Descriptor:

- Net2Display Type: Node and/or Host
- Net2Display Version
- Net2Display Oldest Version Supported
- Manufacturer
- Model Number
- Serial Number
- Hardware Version
- Hardware Date
- Firmware Version
- Firmware Date
- Software Version
- Software Date

Processing Capabilities Descriptor

- List of Compression Algorithms Supported
- List of Preferred Compression Algorithms (high performance support)
- List of Decompression Algorithms Supported
- List of Preferred Decompression Algorithms (high performance support)
- List of Display Remoting Primitives Supported
- List of Preferred Display Remoting Primitives (high performance support)
- Client Render Cursor Support

Network Interface Descriptor (One per Network Interface)

- Network Type
- Network Speed
- MAC Address
- DHCP

- Network Mask
- IP Address
- IP Name
- Default Gateway
- DNS Name Server

Network Protocols Descriptor

- List of Protocols Supported (TCP/IP, UDP, HTTP,XML)
- List of Encryptions Supported

Attachment Capabilities Descriptor

- USB Version
- USB Bandwidth
- USB Ports Available
- Number of Displays Connectors
- Audio Capability

Attachment Descriptors

- Number of Displays Attached
- Display Descriptors (one per display)
    - Display Resolution
    - Display Refresh Rate
    - Display EDID Descriptor
- Number of USB Devices Connected
- USB Descriptor (one per USB device)
    - Port No.
    - Vendor ID
    - Product ID
    - Device class
    - Sub class
    - Protocol
    - Product Name
    - Product Serial No.

22.2.2 WSDL 2.0 Document for Identification and Configuration Web Service

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/2006/01/wsdl"
    targetNamespace= "http://net2display.example.com/2004/wsdl/IDSvc"
    xmlns:tns= "http://net2display.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://net2display.example.com/2004/schemas/IdSvc"
    xmlns:wsoap= "http://www.w3.org/2006/01/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx= "http://www.w3.org/2006/01/wsdl-extensions">

  <documentation>
    This document describes the Net2Display Web service.  Additional
    application-level requirements for use of this service --
    beyond what WSDL 2.0 is able to describe -- are available
    at http://net2display.example.com/2006/configuration-documentation.html
  </documentation>

  <types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://net2display.example.com/2006/schemas/IdSvc"
        xmlns="http://net2display.example.com/2004/schemas/IdSvc">

      <xs:annotation>
        <xs:documentation xml:lang="en">
            N2D Node schema for VESA.org.
        </xs:documentation>
      </xs:annotation>

      <xs:element name="n2dNodeConfiguration">
        <xs:complexType>
          <xs:sequence>

            <xs:element name ="n2dNodeElements">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="n2dHost" type="xs:string"/>
                  <xs:element name="n2dVersion" type="xs:string"/>
                  <xs:element name="n2dOldestVersionSupported"
                                    type="xs:string"/>
                  <xs:element name="n2dManufacturer" type="xs:string"/>
                  <xs:element name="n2dModelNumber" type="xs:string"/>
                  <xs:element name="n2dSerialNumber" type="xs:string"/>
                  <xs:element name="n2dHardwareVersion" type="xs:string"/>
                  <xs:element name="n2dHardwareDate" type="xs:date"/>
                  <xs:element name="nd2FirmwareVersion" type="xs:string"/>
                  <xs:element name="n2dFirmwareDate" type="xs:date"/>
                  <xs:element name="n2dSoftwareVersion" type="xs:string"/>
                  <xs:element name="n2dSoftwareDate" type="xs:date"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
```



```
<xs:element name ="n2dProcessingCapabilities">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="compressionAlgorithmsSupported">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="compressionType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="compressionAlgorithmsPreferred">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="compressionType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="decompressionAlgorithmsSupported">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="decompressionType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="decompressionAlgorithmsPreferred">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="decompressionType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="n2dDisplayPrimitivesSupported">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="primitiveType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="n2dDisplayPrimitivesPreferred">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="primitiveType" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="clientRenderCursorSupport"
                  type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
<xs:element name="networkInterface"
                minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="networkType" type="xs:string"/>
      <xs:element name="networkSpeed" type="xs:string"/>
      <xs:element name="macAddress" type="xs:string"/>
      <xs:element name="dhcp" type="xs:string"/>
      <xs:element name="networkMask" type="xs:string"/>
      <xs:element name="ipAddress" type="xs:string"/>
      <xs:element name="ipName" type="xs:string"/>
      <xs:element name="defaultGateway" type="xs:string"/>
      <xs:element name="dnsNameServer" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name ="networkProtocols">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="supportedProtocol" type="xs:string"
                      minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="supportedEncryption" type="xs:string"
                      minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name ="attachmentCapabilities">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="usbVersion" type="xs:string"/>
      <xs:element name="usbBandwidth" type="xs:string"/>
      <xs:element name="usbPortsAvailable" type="xs:string"/>
      <xs:element name="displayConnectorNumber" type="xs:string"/>
      <xs:element name="audioCapability" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
            <xs:element name ="attachedDevices">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="numberOfDisplaysAttached" type="xs:string"/>
                  <xs:element name="displayInformation"
                                  minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="displayResolution" type="xs:string"/>
                        <xs:element name="displayRefreshRate" type="xs:string"/>
                        <xs:element name="displayEDID" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                  <xs:element name="numberOfUSBDevicesConnected"
                                  type="xs:string"/>
                  <xs:element name="usbDescriptor"
                                  minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="portNumber" type="xs:string"/>
                        <xs:element name="vendorID" type="xs:string"/>
                        <xs:element name="productID" type="xs:string"/>
                        <xs:element name="deviceClass" type="xs:string"/>
                        <xs:element name="subClass" type="xs:string"/>
                        <xs:element name="protocol" type="xs:string"/>
                        <xs:element name="productName" type="xs:string"/>
                        <xs:element name="productSerialNumber"
                                        type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </types>
```



```
  <xs:annotation>
    <xs:documentation>
      Need to Complete Definition of Interface, Binding and Service
    </xs:documentation>
  </xs:annotation>

  <interface  name = "configurationInterface" >

    <fault name = "invalidDataFault"
            element = "ghns:invalidDataError"/>

    <operation name="opGetConfiguration"
            pattern="http://www.w3.org/2006/01/wsdl/in-out"
            style="http://www.w3.org/2006/01/wsdl/style/iri"
            wsdlx:safe = "true">
        <input messageLabel="In"
              element="ghns:GetConfiguration" />
        <output messageLabel="Out"
              element="ghns:GetConfigurationResponse" />
        <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
  </interface>

  <binding name="configurationSOAPBinding"
          interface="tns:configurationInterface"
          type="http://www.w3.org/2006/01/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>

    <operation ref="tns:opGetConfiguration"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
  </binding>

  <service name="configurationService"
       interface="tns:configurationInterface">

     <endpoint name="configurationEndpoint"
               binding="tns:configurationSOAPBinding"
               address ="http://net2display.example.com/2004/configuration"/>
  </service>
</description>
```


A Appendix: Net2Display Remoting Requirements (Informative)

R: Items labeled R (Required) must be included in the first version of the Net2Display Remoting Standard as a required feature, supported scenario or configuration.

O: Items labeled O (Optional) must be included in the first version of the Net2Display Remoting Standard as an optional feature, optionally supported scenario or configuration.

F: Items labeled F (Future) must be included in a future version of the Net2Display Remoting Standard as required or optional.

NS: Items labeled NS (Not Supported) are not currently planned to be included in any version of the standard, but their exclusion is noted for completeness.

A.1 Overview

A.2 Principles of Operation

(O) Net2Display Remoting must enable multiple Clients connected to one Net2Display Host.

A.2.1 Client

(R) Net2Display must not require any permanent storage other than configuration data in Client.

(R) Ethernet connected Net2Display Clients and Hosts must support 10M/100M Ethernet for future capability.

(O) Net2Display must be sufficiently simple that a low power implementation is possible without requiring a fan in the Client, lessening desktop noise.

(O) Net2Display must provide enablement for optional mouse movements to be provided locally on the Client through hardware cursor support.

(O) Net2Display must enable high performance 2D graphics.

(O) A software version of the Net2Display Client will be available that runs on a standard PC.

(O) A software version of the Net2Display may be capable of simultaneously displaying two or more different host sessions

A.2.1.1 Hub Client

(O) Net2Display Remoting must enable a Net2Display Hub for a single user that has adapter ports for connecting one or more displays to DVI or VGA outputs and USB ports for connection the keyboard, mouse and I/O devices.

(O) Net2Display Remoting must enable an inexpensive power-adapter sized Hub that allows the connection of Legacy Displays and I/O.

A.2.1.2 Integrated Display Client

(O) Net2Display Remoting must enable a Net2Display Client that is configured as a network connected Integrated Display with USB ports for connecting the keyboard, mouse and I/O devices.

(O) Net2Display Remoting must enable a Kiosk Net2Display Client that is configured as a network connected display with integrated USB keyboard and pointing device.

(O) Net2Display Remoting must enable a Net2Display Client that is configured as a network connected Integrated Display without any external I/O ports.

(O) Net2Display Remoting must enable the association of multiple displays together.

A.2.1.3 Client Stability

(O) Net2Display Client hardware must be stable for at least five years.

(O) Net2Display Remoting protocol must enable a Client resolution and I/O bandwidth that will be sufficient for 10 years.

(O) Net2Display must not require firmware updates to Client and any optional updates will not be needed more than once every 3 years.

(R/F) A base subset of Net2Display protocols must always be supported by future implementations and versions of the standard.

A.3 Network

A.3.1 Network Characteristics

(R) The IP networking protocol must be used by Net2Display Remoting

(O) A Net2Display Client must be capable of supporting a typical professional user when connected on a dedicated 10 Mbps Ethernet link.

(O) A Net2Display Wireless Client should be implemented with only slightly degraded performance (11 Mbps link).

(O) Net2Display Remoting must be capable of operating over DSL and Cable modems with only moderately degraded performance.

(O) Net2Display Remoting must be capable of operating over a dial-up modem connection with significantly degraded performance.

(R) Net2Display USB-2 data traffic must be transported by a reliable transport mechanism, like TCP/IP.

(O) Since a reliable transport, such as TCP/IP already may be present for handling the USB-2 traffic, this reliable transport may be used for the slowly updating display traffic, providing reliable delivery and managing out of order delivery.

(O) Video and Audio may use an unreliable Transport mechanism.

(O) Net2Display Remoting must provide facilities for flow control to limit bandwidth in congested networks.

(O) Net2Display Remoting may tradeoff of latency and bandwidth to efficiently deliver data with acceptable latency and minimum bandwidth.

(O) Net2Display Remoting must be capable of less that 100 milliseconds latency for simple updates on cross country (2000 miles) internet communication, given 50 milliseconds network latency with a 10 Mbps bandwidth from end to end.

A.4 Data Transport

(R) Use of multiple different protocols must be justified compared to the increase in complexity

(R) TCP service is required to be supported.

(O) UDP service is optional and may be used only on data that can tolerate loss, such as realtime audio.

(R) Any data type that can use UDP for transport, must be capable of operation on TCP

(O) All data of one type must be kept in the same stream or connection or provide a mechanism for preventing reordering problems.

A.4.1 Priorities of Data

(R) If different priorities are not used for different types of data, then the DiffServ bits must be set to 0x0

(O) It may be desirable to use different priorities for different types of data, such as high for interactive, medium for display content and low for USB bulk transfers. This would prevent interactive updates from being stuck behind large bulk transfers.

(O) Prioritization may also be performed in reordering the transmit queue on the sender to transmit interactive updates immediately at a higher priority over bulk transfers. This does not provide the advantages that network prioritization provides in a heterogeneous network environment. Any reordering of the transmit queue must account for the interactions between the reordered data and operations.

(R) If different priorities are used for different types of data, that priority must be indicated both in the contents of the message and in the IP header. This construct allows the PDU contents to be constructed and handled in the sender with the priority and then the IP header to be added later to reflect that priority.

(R) All packets in a TCP connection must have the same setting of the DiffServ bits. Any reordering due to prioritization would only be reversed at the receiver where all data is presented in the order that it was transmitted by the sender.

(O) If Net2Display desires to use multiple different priorities for data sent across TCP, then it must setup a separate TCP connection for each different DiffServ priority. It may be desirable to have multiple TCP connections for different priorities of data and different DiffServ settings.

A.4.2 Reordering

(R) Operations may be combined or reordered at the sending end as long as there is no noticeable effect on the display other than latency of displayed material.

A.4.3 Video

(R) There are three different types of video that need to be supported, but they are indistinguishable at the Host, so they must all use the same means of transport. They are:

- Real time video conference – needs low latency and minimal buffering to preserve phone call like responsiveness
- Real time broadcasts – can have added latency to prevent running out of data
- Stored Video – Can add latency to prevent running out of data

A.4.4 UDP

(O) If UDP is used, different settings on the DiffServ bits may be used on different packets.

(O) UDP may optionally be used for transporting Audio and Video and USB Audio and Video.

(O) UDP may optionally be used for transporting coordinates of a moving pointer

(O) UDP may optionally be used for sending intermediate display updates of a changing display

A.4.5 USB Data

(O) USB data may be transferred at all the same default priority

(R) USB traffic must use TCP to provide data integrity for user data transport. Could TCP be used for the bulk data, mouse and keyboard, and use UDP for video and audio?

(O) If USB desires to use different priority levels for traffic using TCP, then it may only use different priorities between different endpoints and it must use a separate TCP connection for each different priority level expressed in DiffServ.

A.5 Security

(R) Confidentiality and authentication must be provided for Net2Display control and media.Authentication can be either Device Authentication or User Authentication.

(O) Net2Display Remoting security systems are intended to provide security suitable for business office use, financial services and medical services.

(O) Net2Display Remoting may optionally support user defined security services for applications such as the military.

(O) Device Authentication – It must be possible to authenticate that an endpoint (client, host, admin server, etc) is the device it claims to be.

(O) User Authentication – While user authentication is not part of Net2Display Remoting, secure connection of user authentication devices (biometric devices) must be supported.

(O) Authorization – Net2Display Remoting must support a method to determine if an authenticated device is allowed to connect.

(O) It must be possible for an administrator to remotely manage the security configuration of a Net2Display Remoting device.

(O) It must be possible to lock the association between a specific host and a specific client.

(O) Confidentiality - The Net2Display Remoting must be capable of encrypting network traffic.

(O) The encryption algorithms must be negotiated by both ends from a list of available algorithms.

(O) At least one encryption algorithm must be suitable for low-cost hardware implementation.

(O) Integrity – The Net2Display Remoting security system must ensure the integrity of its communication preventing unauthorized insertion, deletion, substitution or playback of communications.

A.6 Content Protection

(F) Premium content protection may be included in a future version of the standard.

(O) Hooks for content protection may be provided in the initial version of the standard.

A.7 PDU Header

A.8 Net2Display PDU Header Fields

A.9 Identification and Configuration Facilities

A.9.1 Node Association Management (NAM)

(O) Net2Display must enable the association of a Client and a Host in less than one second

A.9.2 Client Operation Objectives

- **Consistency**
  - Clients behave predictably when moved to a different network
  - Clients do not have excessive hidden state that causes unpredictable behavior
  - Clients from different vendors behave consistently
  - Clients behave according to a users expectations
- **Simplicity**
  - Client requires minimum configuration when connected directly to a Host
  - Client will go directly to a specified Host, if allowed
- **Security**
  - Services are not advertised to a larger network region than necessary
  - Clients can be locked down to use a Connection Management Server
  - Clients display distinguishable differences when they are being used in Active and Passive modes

A.9.3 Discovery Process

(R) Discovery must specify one primary Discovery Process that works for infrastructure, simple environments and point-to-point links. Other secondary processes may be specified. Examples:

- Infrastructure – DHCP and DNS servers present
- Simple Environment - Local home network, may lack DHCP or DNS server access
- Point-to-point – Net2Display Client is connected to Net2Display enabled computer and display is directed to the Net2Display Client.

(R) The Discovery Process must work in the absence of DHCP and DNS, assigning an address while avoiding duplicate addresses, assigning a name and searching for Net2Display services (Hosts, Connection Management Servers, and Passive Displays)

(R) The Discovery process will use existing standards if those standards meet our discovery requirements.

(R) The Discovery process must work with existing infrastructure.

(R) A Simpler Discovery Process is preferred for ease and correctness of implementation. Available source code would be preferable.

(R) The Discovery Process should make it simple to install new devices, with little or no configuration required to add a new Net2Display Host or Client.

(R) The Discovery Process must be scalable from one Net2Display Client and Host pair to thousands of Net2Display Clients and Hosts.

(R) The Discovery Process must be OS agnostic and support heterogeneous environments of Windows and Linux.

(R) The Discovery Process should have no known IP or licensing restrictions.

(R) Default configuration must be present, so that with it a client can locate a host and connect by default.

(R) One or more common sets of protocols must be supported by all implementations so that all Net2Display systems are interoperable.

(R) Use of multiple different protocols must be justified compared to the increase in complexity.

(R) The Net2Display Client must be responsible for identifying the Net2Display Host to which it intends to connect, to form an association between Net2Display Client and Host.

(O) A variety of means, which may be optionally specified by this standard, may be employed for identifying Net2Display Clients to Hosts, including keyboard entry, USB keys, smartcards, USB cabling to an already associated Client or selection from a list of available Net2Display Hosts.

(O) One option is to have either all Hosts or all Clients implement all Discovery protocols

(R) All Clients and Hosts should be discoverable.

(R) Discovery and Connection can be initiated by either Hosts or Clients.

A.9.4 Connection Initialization

(R) When a Net2Display Client is connected to a Net2Display Host, the first step in the process must be to exchange and negotiate options and settings between the Client and the Host.

(R) During connection initialization, the Client must communicate to the Host the display resolution of the integrated or connected display.

(R) During connection initialization, the Client and the Host must exchange sets of supported compression algorithms and must only use compression algorithms that are supported by the other end.

(R)) During connection initialization, the Client and the Host must exchange sets of supported encryption algorithms and must only use encryption algorithms that are supported by the other end.

A.10 Remoting Commands

(R) The Net2Display Remoting Protocol must provide Client remoting by remoting display, USB-2 I/O and optionally full audio (microphone, MIDI, and wave audio).

(O) The Net2Display Remoting Protocol may be asymmetric, as there are different requirements for the video in each direction.

(O) The USB-2 data transfer requirements are more symmetric and may use a symmetric remoting protocol.

(O) Net2Display Compression must be an optional negotiated feature. Different applications requiring compression are: CAD - wire frame, Full motion video with synchronized audio and text as image.

(O) Keyboard encryption must be supported. Encryption is an optional negotiated feature for other data streams.

A.11 Display Remoting

A.11.1 Graphics

(O) Net2Display must define a Net2Display Graphics Interface that allows graphics chips to directly pass display changes and full motion video streams to the Net2Display Remoting subsystem.

(O) Net2Display must enable Net2Display Hosts that capture the Legacy Display outputs, such as VGA, DVI, DPVL or DisplayPort, and pass it to the Net2Display Remoting subsystem.

(O) The Net2Display graphics interface must enable the support of multiple remoted displays to multiple users.

(O) Net2Display may use the Host graphics 2- and 3-D graphic chip capabilities.

A.11.2 Display Remoting

(R) The Net2Display architecture must be capable of supporting Microsoft's Windows Vista operating system's Aero® desktop experience user interface.

(O) Net2Display must support Scaled Video streams.

(O) Net2Display must efficiently deliver full motion video with high quality and low bandwidth.

(R) Rate and phase of Video and Audio must be synchronized.

(O) Net2Display will optionally provide buffering for video and audio to stay ahead of a congested network.

(O) Net2Display Clients, when suitably configured, may display raw OS user boot screens and blue screens to allow remote problem diagnostics.

A.12 USB I/O Remoting

(O) Net2Display must support USB-2 Ports for remote I/O connections.

(O) Net2Display must use USB keyboards and mice or may use an integrated USB keyboard and pointing device.

(O) Net2Display must optionally allow remote booting via remotely attached USB-2 storage devices.

(R) Net2Display will meet the remoted USB required latency times.

(O) Net2Display will allow USB-2 devices to be optionally disabled by an administrator based on the USB device type, e.g., mass storage devices.

(O) Net2Display must enable USB-2 data to pass through encryption for security of remote data.

(O) Net2Display USB-2 data may be optionally compressed so that bandwidth may be minimized for large data transfers.

A.13 Keyboard and Pointer Remoting

(O) Net2Display must enable responsiveness below 100 milliseconds from mouse click to simple data update.

A.14 Audio Remoting

(O) The merging of multiple audio streams may be supported in a Net2Display Client that supports audio.

(O) Net2Display must support the remoting of audio either directly or over USB.

A.15 Compression Facilities

(R) When compression is used, Net2Display Remoting must use lossless compression for USB data.

(O) Net2Display Remoting compression for display and audio may be lossless or lossy depending upon the application.

(O) Optionally, Net2Display Remoting may use lossy compression on display updates when the network is congested.

(O) When Net2Display Remoting provides lossy compression, an indicator must be given on the Client to show when the compression is lossy.

(O) The Net2Display Remoting compression algorithm must be simple enough for low-cost hardware implementation.

B Appendix: Design Decisions (Informative)

B.1 Objectives

The initial standard write-up will focus on simplicity of design. These selections may be expanded as the standard develops further. This includes:

1. IP must be used as the network protocol for all Net2Display traffic (Vote 10/11/06).
2. A single TCP connection will be used for all display traffic. Multiple TCP connections may be allowed when a scheme is developed on how to manage the multiple connections.

B.2 Overview

B.2.1 Topologies Supported in Version 1

What are the criteria for determining supported topologies? Topologies will be supported if either:

1. The topology does not require any special discovery, signaling or configuration to support
2. The topology is sufficiently compelling that it is expected to be a pervasive topology for Net2Display configurations. This applies for Display Wall Elements without I/O.

What are more complicated Net2Display configurations that may require different Net2Display topologies?

1. Multiple Displays will be a pervasive Net2Display configuration that may be realized by Associating multiple Clients to a single Net2Display Host. Thus, whether this is a significant topological consideration depends upon how multiple Displays are associated with a single Host as examined in Section B.3.2.

The following considerations must be made in supporting the topologies shown in Figure 2-3: Net2Display Example Topologies:

1. Remote Display - Discovery process either using default configuration or reverts to offering SLP service to Hosts
2. Thin Client – typical Net2Display Client
3. Multi-Headed Client – requires additional Configuration information, already specified in standard
4. Multi-Headed DPVL Based Client – Requires Host to support DPVL
5. Display Wall – Same Discovery issues as Remote Display. Also need to Associate multiple displays together
6. Trader with one set User I/O – Requires Association of Multiple Displays
7. Virtualized PCs – Requires Host Network adapter to handle traffic from Multiple Clients
8. Multi-Host Views on One Display – Supported if no additional difficult configuration is necessary

Unanimous Agreement: Net2Display will support configurations that are based on one to one Associations between Clients and Hosts. Where there are multiple Clients communicating with the same server, each will form a separate one to one Association with the Net2Display Host.
Unanimous Agreement: Net2Display will support the discovery and configuration of Net2Display Clients with attached Display and no I/O. This is the case of the Display Wall Element which was viewed as sufficiently compelling to merit support. The means to Discover these Display Wall elements is still to be determined
Unanimous Agreement: While all setup, discovery and configuration will be done one to one between Client and Host, multicast UDP may be used for distributing video or audio to Clients receiving the same material from a Host.

B.2.2 Hosts

Within the Net2Display Standard, different designations are needed for: 1) the computer system or operating system instance that produces the display output to be remoted, 2) the Net2Display instance that includes all of the hosted Associations from the system and manages them and 3) the Host end of the Net2Display connection that goes to a single Client. These can be defined as:

Host Server: A Host Server is a computer or operating system instance that is capable of sourcing Net2Display Remoting traffic for display on Net2Display Displays. In general, this standard will use the term "Host Server" to refer to the image source (may be a PC, workstation, set-top-box, etc).

Host Node: The Host Node is the full Net2Display Host instance that resides at a Host Server and includes the Host Ends of all of the Net2Display connections to remote Clients. There is one Host Node for each Host Server which may contain multiple Host Ends.

Host End: A Host End is the single instance of the Net2Display Remoting protocol that implements the sourcing end of the display and communicates with the Client on the other end of the Connection. There may be multiples instances of Host Ends in a Host Node.

The alternative naming conventions include:

- Host Server/Host Node/Host End
- Host Server/Host/Host End
- Host Server/Host Node/Host

Decision: Use Host Server/Host Node/Host End designations.

B.3 Client Configurations

Clients can take on different modes during the Discovery Process. The Client can either initiate the Discovery Process or can be located by a Host during the Discovery Process. In one case, the Client initiates the Discovery Process and Association and in the other case, the Client is located during the Discovery Process and receives an Association request from a Host. These different Client Modes need different names. Some options on the names that may be used and examples of similar usage are:

- Active Mode/Passive Mode – used by FTP for different transfer modes
- Initiator Mode/Target Mode – used by PCI, SCSI, iSCSI for each transaction
- Primary Mode/Secondary Mode
- Terminal Mode/Display Mode
- Initiating Mode/Receiving Mode
- Principal Mode/Minor Mode

Decision: Use Active Mode/Passive Mode to designate Client modes.

B.3.1 Net2Display Classes

B.3.1.1 Net2Display Standard Version 1 Will Only Specify One Class of Host and Clients

Advantages of Specifying Multiple Classes in Standard:

- There will be distinct Net2Display systems that do not need all the capabilities that are defined within Net2Display

- Could potentially save money on usages that do not require the full Net2Diplay functionality

- Could distinguish higher performance Net2Display capable systems, such as 3-D graphics

Disadvantages of Specifying Multiple Classes in Standard:

- Could fractionate Net2Display into different distinct uses

- Could cause confusion in the marketplace

Whether different classes of Net2Display are defined depends upon:

1. The size of the market for special purpose devices that would use the simpler classes
2. The size of the market for higher performance devices that should be differentiated
3. The ability to define differentiating characteristics for the different classes of Net2Display devices

Special Purpose systems that may benefit from having Net2Display classes defined include:

1. Special Purpose Display Walls
2. Kiosks
3. ATM machines
4. Thermostats
5. Refrigerator displays – TV capable
6. 3-D graphics workstation
7. TV Video Clients – Full Motion Video Capable designation
8. USB Hubs – Display-less

Potential differentiating characteristics that may be used for distinguishing the different Net2Display classes include:

1. Motion Video Support and Compression
2. USB I/O
3. Presence of Display
4. Audio – may not be needed for video walls, ATMs and thermostats, but would be low cost to include

Unanimous Agreement: Net2Display will have only one Class for the first version of the standard, but the Class will be communicated in the Configuration information. Future versions of the Net2Display standard may define additional Classes.

B.3.1.2 Characteristics that Could Differentiate Classes in a Future Net2Display Version

As noted in the previous section, there will be only one Net2Display class supported in Version 1 of the standard. For a future version of the standard, when different classes may be present, the best distinguishers for different classes would be the most expensive options to implement, such as options requiring additional processing, memory, hardware or I/O bandwidth. Some features that may be used for distinguishing different classes of Net2Display are shown in Table B-1.

Table B-1: Feature Usage by Scenario

Feature Usage by Scenario Usage Key: 0 = Not Needed 1 = Optional 2 = Required	Static & Extended Setup	Pointer/Touchscreen	Keyboard	Other USB I/O	TCP/IP	Encryption	Compression	Motion Video	Local Cursor	Extreme Low Latency	Audio	IP Phone	Multiple Displays	Total
Airport Display Wall	2	0	0	0	0	0	1	0	0	0	0	0	1	4
Store Catalog/Map/Sizing Kiosk	2	2	1	1	0	0	1	0	0	0	1	0	0	8
Conference Room Projector: Local or Remote	2	1	1	1	0	2	1	1	0	0	1	1	0	11
Store Purchase Kiosk with Credit Card Reader	2	2	1	2	2	2	1	0	0	0	1	0	0	13
ATM Machine	2	2	1	2	2	2	1	0	0	0	1	0	0	13
Cash Register	2	2	2	2	2	2	1	0	0	0	1	0	0	14
Patient Room Medical Instruments Monitor	2	2	2	2	2	2	1	0	0	0	2	1	0	16
Corporate Broadcast and Advertising Display	2	0	0	0	0	0	2	2	0	0	2	0	0	8
Video Display Wall or Advertising Wall	2	0	0	0	0	0	2	2	0	0	1	0	2	9
Nurses Station Terminal	2	2	2	2	2	2	2	1	1	0	2	1	1	20
Library Internet Browsing PC	2	2	2	1	2	2	2	2	1	0	2	0	0	18
Remote Help Desk PC	2	2	2	1	2	2	2	1	1	0	2	1	1	19
Remote Local Insurance Office PC	2	2	2	2	2	2	2	1	1	0	2	1	1	20
Administrative Assistant PC	2	2	2	2	2	2	2	2	1	0	2	1	1	21
Basic Home User with Local Server	2	2	2	2	2	2	2	2	1	0	2	1	1	21
Network TV, Projection TV or Projected Home Computer Display	2	2	2	2	2	2	2	2	2	0	2	1	0	21
Basic Home User optionally using remote Hosted PC Service	2	2	2	2	2	2	2	2	2	0	2	1	1	22
Professional PC	2	2	2	2	2	2	2	2	2	1	2	1	2	24
High End Medical or Satellite Imaging Display	2	2	2	2	2	2	2	2	2	2	2	1	2	25
Gaming Home PC	2	2	2	2	2	2	2	2	2	2	2	1	2	25
Financial Trader	2	2	2	2	2	2	2	2	2	2	2	1	2	25
High End CAD Graphics	2	2	2	2	2	2	2	2	2	2	2	1	2	25

B.3.1.3 Initial Net2Display Version 1 Defaults

The default configuration is not of major significance if all the configuration information is exchanged between the Host and Client during the Association process. A default configuration would only have to be defined if the exchange of configuration information was optional, as discussed in Section B.9.7.10, which would have severely limited the interoperability of future versions of the standard.

Unanimous Agreement: Net2Display needs to define default parameters for the setup for Net2Display Clients and Hosts so that systems can find default Connection Management Servers and Hosts with an out of the box configuration.

B.3.1.4 Options Allowed in Net2Display Version 1 will be Gathered as the Standard Progresses

Options allowed in Net2Display Version 1 will be:

1. Optional compression approaches
2. Multiple Display support

Unanimous Agreement: The Net2Display options will not be defined at this time, but will be gathered as the standard progresses and noted as options.

B.3.1.5 Minimal Requirements for a Client are Those for an I/O-less Display Client

Unanimous Agreement: The minimal requirements for a Net2Display Client are the minimal characteristics of an I/O-less Display Client.

B.3.2 Multiple Display Support:

B.3.2.1 What are the Different Means to Support Multiple Displays?

Multiple Displays may be Associated together to display information from a single Host for a single User using the following different means:

1. The Client has multiple display connections that multiple Displays are connected to. This has the advantage that it is simple to implement. The disadvantages of this approach are:
    a. It is not scalable and does not scale beyond a small number of displays.
    b. It also does not take advantage of Net2Display Integrated Displays
2. Multiple Net2Display Clients, likely Integrated Displays, associated with the same Host and user. These multiple Net2Display Clients can be Associated together either by:
    a. Requiring the user to sign onto each Client separately to form the Associations. This is simple to implement, but is a much more complex procedure than what is necessary for a user to use multiple Displays with a PC.
    b. Requiring the user to sign onto a Primary Client and the Associated Clients are added in through other means. These means include:
        i. A physical connection between the additional displays and the Primary Client that provides the Host information to the Secondary Clients. This could be accomplished either through a USB connection or through another specialized connection
        ii. After the Primary Client is Associated with the Host, the Host locates the other Clients and joins them in with the Primary Client. This requires the Host to be able to Discover and Associate to Clients which is an issue examined in Section B.9.7.9. The first time this is done requires user setup, but on subsequent Primary Client Associations with the same Host, the Secondary Clients could also be brought along.
        iii. After the Primary Client is Associated with the Host, the user provides the networking addresses of the Secondary Clients to the Host. The first time this is done requires user setup, but on subsequent Primary Client Associations, the Secondary Clients could also be brought along.

Recommendation: Support Multiple Displays with TBD Association Mechanism.

B.3.3 Multiple Host Support

Can a Net2Display Client show material from multiple Hosts? Can it be shown simultaneously? Can it swap between showing different Hosts? Does this need to be specified in the standard? Are Hooks added to the standard to allow this functionality?

Unanimous Agreement: Net2Display will allow the support of connections to multiple Hosts by one Client, where the Client can switch between viewing the display contents from each of the connected Hosts
Unanimous Agreement: Additional commands will need to be added to Net2Display to support the switching between viewing different Hosts including: a channel for selecting the Host for display, a refresh or update screen command, a disconnect command, and a command from the Client to Host to indicate that the Host is not currently being viewed and the Host should not send updates.
Unanimous Agreement: As for composing the Display on the Client from different Host Sources, the Net2Display standard will not explicitly define or prohibit composing on the Client by an Application.

B.3.4 Net2Display Clients Must Have a Minimal Display Size of One Pixel

Displayless Clients are not allowed in Version 1 of the Net2Display Standard. Displayless Clients would only be possible if either Hosts are allowed to locate Clients (See Section B.9.7.9), if those devices are configured to find the default Association Server, or if displays can be connected temporarily to the Clients to configure the client for Discovery.

The issues that prevent a Display-Less Client are very similar to those that make a display wall element with no attached I/O difficult, it is difficult to initiate a Discovery process from a Client with a keyboard and display. Two approaches are:

The Client uses the default Discovery process to locate the Target Locator and the default Host.

If the Client is unable to locate a Host, then the Client will register itself with a SLP Server as a SLP Service. This would then allow a local Host to locate the Client using SLP.

A possible rule to add in the Discovery Process is that if a Display-less Client or an I/O-less Display fail in the Discovery process, then they register with SLP and can then be discovered from the Host.

Unanimous Agreements: Net2Display will only require a Display with a Client, with the Display required to have a minimal size of one pixel. Thus, a single LED will meet the minimal display size requirement.

B.4 Principles of Operation

B.4.1 Required Net2Display Features

Is this a set of required Net2Display Features?

- TCP/IP or UDP Networking
- USB I/O or no USB I/O
- Remoted USB Keyboard and Pointer or Client Controlled Keyboard and Mouse
- HTTP support
- Full Discovery Process support
- Display Commands Supported: RawPixel, Copy, SolidFill, PatFill, BiFill
- Support for Video Streams
- Compression Types Supported
- Client Aggregation or no Aggregation

- Client and User Authentication
- SSL/TLS or IPsec Security

B.4.2 Optional Net2Display features may include:

1. Audio

B.5 Networking

Unanimous Agreement: Decision on 2/13/08 was to adopt TCP to be required and all other Transports to be optional, with UDP and Reliable Transport over UDP to be specified with the Net2Display Standard

Unanimous Agreement: A keep-alive or ping function will be added to maintain the congestion window and the connection. This "heartbeat" will also be timestamped to provide Net2Display with a means for computing the network latency.

B.5.1 Communications Protocol Options

Table B-2: Comparison of Transport Protocols

	UDP	TCP	DCCP	SCTP	Net2Display Specific Transport
Header size	8 Bytes	20 Bytes	Varies	12 Bytes + Variable Chunk Header	8 to 12 Bytes
Transport layer unit	Datagram	Segment	Datagram	Datagram	Datagram
Port numbering	Yes	Yes	Yes	Yes	Yes
Error detection	Optional	Yes	Yes	Yes	Yes
Reliability: Error recovery by automatic repeat request (ARQ)	No	Yes	No	Yes	Yes
Virtual circuits: Sequence numbering and reordering	No	Yes	Yes	Optional	Optional
Flow control	No	Yes	Yes	Yes	Yes
Congestion avoidance: Variable congestion window, slow start, time outs	No	Yes	Yes	Yes	Yes
Multiple streams	No	No	No	Yes	Yes
Windows Implementation Available	Yes	Yes	**No**	**No**	**No**
Linux Implementation Available	Yes	Yes	Yes	Yes	No
Wide Use and Experience	Yes	Yes	**No**	**No**	**No**
Suited for Hardware Implementation	Yes	**No**	No	No	Yes

A standard transport protocol must be used. It would take too long to develop and implement a Net2Display specific transport protocol. Both SCTP and DCCP do not have the maturity of UDP and TCP.

Recommendation: Use TCP and UDP for all data transport. Define a generic interface to the transport level so that a alternative transport could be defined and used.

B.5.2 Network Transport

Typically, TCP is used to provide a reliable transport for the remoting of Net2Display traffic. However, for Motion Video and Audio traffic, the unreliable UDP protocol may be used, allowing lost packets to be skipped rather than introducing the additional latency that a reliable protocol such as TCP would incur when recovering from lost network packets. Net2Display communication typically takes place using TCP connections and UDP datagrams for transport. This does not preclude Net2Display from being implemented

on top of any other protocol on the Internet, or on other networks. The required features of a transport protocol to be used by Net2Display Remoting include:

- Reliable – retransmissions of undelivered PDUs
- In-Order – PDUs are presented to the recipient in the order they were originally generated
- Datagram Service – PDUs transferred as a series of independent messages
- Stream Service – PDUs transferred as one continuous data stream
- VC_Timestamp – a Timestamp is provided in PDUs to allow media to be presented in a time synchronized manner
- Ack Feedback – acknowledgments are sent in response to Net2Display PDUs to provide flow control, congestion control or reliability

The generic transport types that need to be supported by Net2Display are reliable in-order datagram service, reliable in-order stream service and unreliable datagram with time stamp and ack feedback. These different generic transport types can be supported by Net2Display by using the standard transport protocols as shown in Table 22-1.

Table 22-1: Preferred Transport Protocols for Different Generic Service Types

Data Type	Default Transport, For Longer Distance	When UDP traffic is blocked	**Optional** Higher Performance	**Optional** Transport
Reliable In-Order Datagram Service	Datagram Service over TCP	Datagram Service over TCP	Vendor Specific Reliable In-Order UDP	SCTP Ordered
Reliable In-Order Stream Service	TCP Stream	TCP Stream	TCP Stream	SCTP Ordered
Unreliable Datagram with TimeStamp and Ack Feedback	UDP w/ TimeStamp & Ack Feedback	TCP Stream	UDP w/ TimeStamp & Ack Feedback	Unordered SCTP

Thus, the different types of transport that are needed according to Table 22-1 are:

- Datagram Service over TCP - No N2D Ack needed, May or may not have TimeStamp
- TCP Stream Service - No N2D Ack needed, May or may not have TimeStamp
- Optional Reliable In-Order UDP – Uses N2D Ack Feedback. This Transport may or may not have or use TimeStamp
- Unreliable UDP with TimeStamp and N2D Ack Feedback

Net2Display Remoting allows for flexibility and optionally allows the following transports if they are supported by both the Host and the Client:

- Unreliable UDP without TimeStamp or Ack Feedback
- Optional SCTP – Ordered or Unordered

B.5.2.1 Transmission Control Protocol (TCP)

TCP is one of the core protocols of the Internet protocol suite. Using TCP, applications on networked hosts can create *connections* to one another, over which they can exchange data or packets. The protocol guarantees reliable and in-order delivery of sender to receiver data. TCP also distinguishes data for multiple, concurrent applications (e.g. Web server and email server) running on the same host.

B.5.2.2 User Datagram Protocol (UDP)

UDP is one of the core protocols of the Internet protocol suite. Using UDP, programs on networked computers can send short messages known as *datagrams* to one another. UDP does not provide the reliability and ordering guarantees that TCP does; datagrams may arrive out of order or go missing without notice. However, as a result, UDP is faster and more efficient for many lightweight or time-sensitive purposes. Also its stateless nature is useful for servers that answer small queries from huge numbers of clients. Common network applications that use UDP include the Domain Name System (DNS), streaming media applications, Voice over IP, Trivial File Transfer Protocol (TFTP), and online games.

Unanimous Agreement: UDP will be allowed by the first version of the Net2Display standard as a Transport protocol.

B.5.2.3 Stream Control Transmission Protocol (SCTP)

Not setting the U bit in SCTP means reliable delivery. When the U bit is not set, the recipient waits for the next packet before and only presents all packets in order. So SCTP either presents all packets in complete order (like TCP) or only received packets in the order received (like UDP). So SCTP either gives Transport Services like TCP or UDP.

General Agreement: SCTP would not be specified in the first version, but TCP and UDP would be used. Reasons:

- No Windows implementation of SCTP currently available.
- SCTP supports multiple streams, but chunks from the different streams can be mixed in the same packet not under the control of the application, but by SCTP, making setting priorities on packets with mixed content difficult.
- Corresponded with Randy Stewart, editor of SCTP and he agreed that using DiffServ priorities with SCTP is difficult.
- One IANA registered Port Number for TCP and one for UDP can be used for supporting multiple priorities, because the registered Port Number is used on the Host end and different Port Numbers may be used on the Client for different priorities.

B.5.2.4 Real-time Transport Protocol (RTP)

General Agreement at Nov. 15, 2007 Face to Face: RTP is not needed as it introduces additional unnecessary complexity

Real-time Transport Protocol (RTP) defines a standardized packet format for delivering audio and video over the Internet. RTP is built on top of UDP. RTP communications over UDP are done on an even port and the next higher odd port is used for RTP Control Protocol (RTCP) communications. RTP is generally configured to use ports 16384-32767. RTP only carries voice/video data. Call setup and tear-down is usually performed by the SIP protocol. The fact that RTP uses a dynamic port range makes it difficult for it to traverse firewalls. In order to get around this problem, it is often necessary to set up a STUN (Simple Traversal of UDP over NATs) server.

Table B-3: RTP Packet Format

Byte 0				Byte 1		Byte 2	Byte 3
V=2	P	X	CSRC Cnt	M	Payload Type	Sequence Number	
Timestamp							
Synchronization Source (SSRC) Identifier							
Contributing Source (CSRC) Identifiers							

B.5.2.5 Real Time Streaming Protocol (RTSP)

RTSP is a protocol for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as "play" and "pause", and allowing time-based access to files on a server. Some RTSP servers use RTP as the transport protocol for the actual audio/video data. Many RTSP servers use RealNetworks's proprietary RDT as the transport protocol.

B.5.2.6 Session Initiation Protocol (SIP)

General Agreement at Nov. 15, 2007 Face to Face: SIP is not needed as it introduces additional unnecessary complexity

B.5.2.7 Voice over Internet Protocol (VoIP)

Voice over Internet Protocol (also called VoIP, IP Telephony, Internet telephony, and Broadband Phone) is the routing of voice conversations over the Internet or through any other IP-based network. The voice data flows over a general-purpose packet-switched network, instead of dedicated legacy circuit-switched telephony transmission lines. Protocols used to carry voice signals over the IP network are commonly referred to as Voice over IP or VoIP protocols.

Because IP does not provide a mechanism to ensure that data packets are delivered in sequential order, or provide Quality of Service guarantees, VoIP implementations face problems dealing with latency and jitter. This is especially true when satellite circuits are involved. The receiving node must restructure IP packets that may be out of order, delayed or missing, while ensuring that the audio stream maintains a proper time consistency. This functionality is usually accomplished by means of a jitter buffer.

Another challenge is routing VoIP traffic through firewalls and address translators. Private Session Border Controllers are used along with firewalls to enable VoIP calls to and from a protected enterprise network. Some VoIP traverse firewalls using protocols such as STUN or ICE.

B.5.2.8 STUN (Simple Traversal of UDP over NATs)

STUN (**Simple Traversal of UDP over NATs**) is a network protocol allowing clients behind NAT (or multiple NATs) to find out its public address, the type of NAT it is behind and the internet side port associated by the NAT with a particular local port. This information is used to set up UDP communication between two hosts that are both behind NAT routers. It will work with three of four main types of NAT: full cone NAT, restricted cone NAT, and port restricted cone NAT. It will not work with symmetric NAT (also known as bi-directional NAT) which is often found in the networks of large companies.

B.5.3 Reliable Datagram Protocol

B.5.3.1 What are the Desirable Transport Options for Net2Display?

1) A Standard reliable transport for setting up Net2Display Associations and configuring Associations – TCP

2) A Standard reliable transport for the static portion of the display and for USB data traffic – TCP

3) A Standard low overhead unreliable transport for video and audio – UDP or DCCP

4) A negotiated low overhead reliable transport for the static portion of the display and USB data traffic. This should be high performance in an inexpensive implementation. It should have flow control and congestion avoidance capabilities – No standard transport offers these capabilities.

Note: On 3/26/2008 the Net2Display task group voted to delay specifying a reliable transport protocol and to add a note that a reserved bit in the Net2Display Header can be used for indicating when an optional transport protocol is included in Net2Display

B.5.3.2 What are the Options for Providing a Low Overhead Reliable Transport, Since no Standard Currently Exists?

1) Define a new Transport protocol that goes below Net2Display. This would likely need to go through IETF approval and would take at least a year to solidify and approve.

2) Define a reliable transport within the regular Net2Display header. This work would be extensive. At this time, the following would need to added:

   a) Definition of all the transport fields in Net2Display

   b) Specification of all the values in all the fields for all the types of packets. This includes specification of the synchronization of the sequence numbers.

   c) Algorithms for flow control and congestion avoidance.

3) Separate out and make optional the fields required to provide a reliable datagram service in Net2Display. A bit could be defined that designates when the optional header is present, so the full optional header would not need to be defined at this time. The size of the optional header could be defined at this time.

4) Delay on the solution and solve at a later time. This low overhead reliable transport could be included later through the specification of the protocol in the IP field or through an extension of the Net2Display header using a reserved bit in the header.

Note: On 3/26/2008 the Net2Display task group voted to delay specifying a reliable transport protocol and to add a note that a reserved bit in the Net2Display Header can be used for indicating when an optional transport protocol is included in Net2Display

B.5.3.3 Does Net2Display Remoting Create its own Reliable Datagram Protocol or Does it Use an Existing Reliable Datagram Protocol?

Existing defined reliable Datagram protocols are given below. Some are defined IP protocol types (See http://www.iana.org/assignments/protocol-numbers). Reliable Datagram protocols can be specified and used by just specifying the IP protocol type in the IP header. If an existing reliable datagram service was used, the header for it would be placed before the Net2Display header. Here are some existing reliable datagram protocols:

- RDP (IP Protocol #27) (rfc 908, rfc 1151)
- Xpress Transport Protocol (IP Protocol # 36) (http://www.cs.columbia.edu/~hgs/internet/xtp.html, http://www.cs.virginia.edu/~acw/netx/xtp_long.html)
- RUDP (See http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-sigtran-reliable-udp-00.txt)
- T/TCP (RFC 1644) TCP Extensions for Transactions

Or Net2Display Remoting could define its own reliable datagram transport. This would require much research and would produce a similar result to what is already present.

B.5.3.4 What are the Existing Reliable Datagram Protocols?

	Description	Document	Date Developed	IP Protocol #	Widely Used	Implementations Available	Complexity	Congestion Avoidance	Windowing	Flow Control	Selective Acks	NACK	Comments
Reliable Data Protocol (RDP)	Early rfcs for Reliable Datagrams	rfc 908 & rfc 1151	1984 - 1990	27	**No**	**No**	**Low**	**No**		No			Lack of flow control and congestion avoidance
Xpress Transport Protocol (XTP)	Designed for hardware Implemen-tation.	XTP Specifi-cation Version 4	Up to 1995	36	**No**	**No**	**Low**	**No**	Yes	Yes			No work on this since 1995
Reliable UDP Protocol (RUDP)	For transport of telephony signaling over IP		1999		**No**	**No**	**Low**	**No**					Replaced by SCTP? Likely dropped to no congestion avoidance
TCP Exten-sions for Trans-actions (T/TCP)	TCP extension for transaction oriented request/ response service	rfc 1644	1994		No	No	Low						Major spoofing problem, not widely used
Microsoft R-UDP	Used in MediaRoom IPTV multicast delivery	**Proprie-tary**			Micro-Soft Only	No	?	?	?	?	?	?	
Apple Reliable UDP	Used with Quicktime Streaming	**Proprie-tary**			Apple Only	No	?	?	?	?	?	?	
IBM Rapid Transport Protocol	IBM SNA Protocol	**Proprie-tary**	1995		IBM Only	No	Low						
SCTP			2000		No	Yes	High						Formerly MultiNetwork Datagram Transmission Protocol (until July 99)
Net2-Display specific Transport	Not yet fully defined		TBD		Future	TBD	TBD						

B.5.3.5 What Characteristics are Needed in a Reliable Datagram Protocol used with Net2Display Remoting?

1) Reliability
2) Congestion Control, similar to TCP/IP – adapts transmissions to network bandwidth available
3) Ability to process packets received out of order within some window size
4) Flow Control – prevents source from providing more data than receiver can process
5) Other?

B.5.3.6 If Net2Display Remoting was to Define its own reliable datagram service, how would it be included in Net2Display Remoting?

Reliable UDP (RUDP) is only partially specified in Net2Display Remoting. RUDP will need to be more fully specified for its inclusion in Net2Display Remoting. Options for including Reliable UDP in Net2Display Remoting are:

1) Fully specify RUDP in Net2Display now
    a) Advantages
        i) Would allow us to ensure that the RUDP fields are fully specified
        ii) Would allow the fields to be placed at multiple different locations in the header
2) Specify separate header for RUDP that is always present
    a) Advantages
        i) Allows RUDP fields to be defined later
3) Specify Optional separate header for Reliable UDP. This header would be present when indicated by a bit in the regular header indicating RUDP as the transport.
4) Separate out header bits for reliable UDP and let Dave Hobbs define in separate section later
    a) Disadvantages
        i) Hard to separate out Sequence Number since it may be used by both reliable and unreliable UDP

B.5.3.7 If a Reliable Datagram Service is Included in Net2Display Remoting, Where are the Fields it uses Included in the Net2Display Header?

Options:

1) Front – Since the other transport headers are located in front of the Net2Display header. Also placing it all at one location would make it easier for it to be an optional header.
2) After Virtual Channel Identification – Since the Reliable UDP is provided on a per Virtual Channel basis, it only makes sense to identify the type of transport for a Virtual Channel after that channel has been identified. Also placing it all at one location would make it easier for it to be an optional header.
3) Mixed – It may be beneficial to provide the identification of transport type very early in the header, since it is know whether it is TCP or UDP very early in the packet. Other RUDP fields may be better placed after the Virtual Channel is identified since the Virtual Channel number serves as a connection identifier. Mixed may be the best approach since the Sequence Number will be used by both the Virtual Channel and RUDP

B.5.3.8 Fields Used by Reliable UDP

The following fields are used by RUDP.

- RUDP Bit – A bit that is present that indicates that this PDU is part of a Virtual Channel that is using RUDP. This bit is set in all RUDP PDUs including data and Ack PDUs. This bit is analogous to the packet type in the IP header.
- Data Bit – called Sequential Packet bit in Datagram and Transport Acknowledgements Systems document
- Is a SYN bit needed to initialize and synchronize the Sequence Number like TCP? Especially if the number is used for providing reliability?

B.5.3.9 Sequential Data Bit

Field indicates that this is a reliable UDP stream, that there is data in this PDU (Not just an ACK), and that there is a Sequence Number present that applies to this data.

Note: It may be preferable to have a bit that indicates when data is present, which helps for the Sequence Number field

B.5.3.10 ACK Bit

Ack Bit indicates that PDUs up to the Sequence Number in the Ack Sequence Number field have been successfully received.

B.5.3.11 NAK Bit

Need to add a NAK Bit. When the NAK bit is set, the Ack Sequence number contains the number of the last packet that was successfully received.

B.5.3.12 Sequence Number

When is Sequence Number incremented? Is it only incremented when data is present?

Sequence number when present must always be incremented by one for each PDU sent. Sequence numbers are useful for reliable delivery and could also be used for congestion detection or flow control. Sequence number is not useful for TCP packets and is really redundant for that case.

Options for handling when this field is valid are:

1) Sequence Number is required in all packets that contain Data. It does not appear in Ack packets. There is no one bit that indicates that Data is present in packet. This could be derived from the Sequential Data Bit not being set for a reliable UDP Virtual Channel.
2) Sequence Number is required to be valid for all UDP data Packets.

B.5.3.13 Ack Sequence Number

The Ack Sequence Number field is valid if the ACK or NAK bits are set.

B.5.3.14 Command Code

What CommandCode is used in an Ack Packet? Options:

1) There is some way that the processing knows that there is no Data. This could be derived from the Sequential Data Bit not being set for a reliable UDP Virtual Channel.
2) A Null Command Code could be useful for sending in Ack Packets and for sending PDUs for keeping connections open

B.5.3.15 VC_Timestamp

The VC_Timestamp is really mainly needed for the video and audio channels traveling over variable latency paths. Five options for presence of VC_Timestamp:

1) VC_Timestamp appears in every PDU
2) VC_Timestamp appears in every data PDU, but not in the Ack PDUs
3) VC_Timestamp appears only in the data PDUs where it was indicated to appear in the setup of the Virtual Channel
4) There is a bit in every PDU that indicates the presence of a valid VC_Timestamp
5) The VC_Timestamp is valid when it takes on a non-zero value.

B.6 Security

B.6.1 Mandatory Security

Due to the difficulty of managing security when it is optional and the presence of security as an integrated part of RDP, a decision was made to make security mandatory.

Unanimous Motion Approval (7/19/07): To require confidentiality and authentication for Net2Display control and media.

B.6.2 Client Configuration

Unanimous Agreement: Net2Display Clients need to have a capability defined for locking down Clients in a standards defined manner so that they can be managed by an IT organization and the user cannot change the configuration.

- A Net2Display system will initially be configured as trusting and then after configuration security will be strengthened.
- A way for a Management Console to access the basic Client Configuration information must be defined for Net2Display Remoting.

B.6.3 TPM

Unanimous Agreement: TPM will be an option in Net2Display that is communicated in the Configuration information.

B.6.4 Certificate Authorities and Certificates will be Recommended in an Appendix

Unanimous Agreement: Net2Display will have an Appendix that will recommend the certificate authorities and certificates to be in the Client.

B.6.5 IPSec

IPsec (**IP security**) is a standard for securing Internet Protocol (IP) communications by encrypting and/or authenticating all IP packets. IPsec protocols operate at the network layer. Other Internet security protocols in widespread use, such as SSL and TLS, operate from the transport layer up. This makes IPsec more flexible, as it can be used for protecting both TCP and UDP-based protocols, but increases its complexity and processing overhead, as it cannot rely on TCP to manage reliability and fragmentation.

IPsec is a set of cryptographic protocols for (1) securing packet flows and (2) key exchange. Of the former, there are two: Encapsulating Security Payload (ESP) provides authentication, data confidentiality and message integrity; Authentication Header (AH) provides authentication and message integrity, but does not offer confidentiality. Originally AH was only used for integrity and ESP was used only for encryption;

authentication functionality was added subsequently to ESP. Currently only one key exchange protocol is defined, the IKE (Internet Key Exchange) protocol.

B.6.6 TLS or SSL

Secure Sockets Layer (**SSL**) and **Transport Layer Security** (**TLS**), its successor, are cryptographic protocols which provide secure communications on the Internet. There are slight differences between SSL 3.0 and TLS 1.0, but the protocol remains substantially the same.

In typical use, only the server is authenticated (i.e. its identity is ensured) while the client remains unauthenticated; mutual authentication requires public key infrastructure (PKI) deployment to clients. The protocols allow client/server applications to communicate in a way designed to prevent eavesdropping, tampering, and message forgery.

B.6.7 Security Options

The different combinations of types of communication and security are shown in Table B-4. It is expected that one approach will be required for a secure connection-based service and one approach.

Table B-4: Security Option Characteristics

	Insecure HTTP	HTTP using TLS	HTTP using IPSec	Insecure TCP/IP	TCP/IP using TLS	TCP/IP using IPSec	Insecure UDP	UDP using TLS	UDP using IPSec
Secure Connection Based		X	X		X	X			
Insecure Connection Based	X			X					
Works across most restricting Firewalls	X	X							
Works across most Firewalls	X	X		X	X	X			
Secure Datagrams								X	X
Insecure Datagrams							X		
Allowed by all Residential broadband services	X	X		X	X		X	X	
Works with all NAT routers	X	X		X	X		X	X	

B.7 Association Identification

Decision: Use the secure transport to group communication within one Association

The different options evalued were:

1) There is the potential for multiple Associations between a Host/Client pair. Each Association may use multiple different transports, including TCP and UDP. Typically, all of the Transports in one Association are grouped together in one Secure Transport. How is traffic identified and grouped for one Association? There are three possible approaches:

    a) Add an Association_Identifier to the Header of all PDUs.

    b) Use the Secure Transport to group traffic within one Association. The Secure Session can be used to group different Transports together that make up one Association. The multiple TCP and UDP communication channels that make up one Association can be grouped by including them all in the same Secure Session, typically IPsec.

c) Prevent multiple Associations between the same Host/Client pair

Approach	Advantages	Disadvantages
Association ID in Header	• Consistent Solution • Association ID could be used for reconnection • Association switching between Network Interfaces	• Wastes Bandwidth: Adds another word to every PDU Header
Group in Secure Transport	• Simple Approach	• Association not explicit • Secure Transport ID may not be present for other Security Transports
Prevent Multiple Associations between same Host/Client pair	• Simple Approach	• Limits operation

B.8 Net2Display PDU Header

B.8.1 Net2Display Version Header Field Group

B.8.1.1 Version Number – 3 Bits

The Version bits allow the Net2Display header to be improved in the future. The Version Number must be set to 0 for pre-standardization versions. The Version Number shanll be set to 1 for implementations conforming to this first version of the standard. The Version Field should allow one version for prototyping, one for the original standard and two for future upgrades, for a minimum of two bits.

B.8.2 Transport Properties Header Field Group

B.8.2.1 Unordered Bit (U Bit) – 1 Bit

The Unordered Bit indicates that in-order delivery service is not required and is used to indicate to the transport layer that UDP can be used instead of TCP. The Unordered Bit may be beneficial to have present after TCP and UDP header is removed to indicate the type of processing.

B.8.2.2 Priority Bits – 3 Bits

The Priority Bits express the prioritization of the PDU. Generally, the priority bits are set the same for a given Virtual Channel in the same direction.The Priority bits are included because they give hints to priority of processing for the PDU at the Net2Display source before the PDU is encapsulated in an TCP or UDP packet and gives hints to the Net2Display destination after the TCP or UDP packet header is removed. These bits also give direction to the transport layer for the setting of the DiffServ bits.

B.8.2.3 Data Transport Priorities

Different requirements for the transport of data exist for each different type of data. These need to be met by the transport protocol. A unit of information transferred between one Net2Display instance and another Net2Display instance is called a Protocol Data Unit (PDU). The different types of data used by Net2Display are transported using different Virtual Channels, different Transport Protocols and different priorities of IP Differentiated Services as shown in Table B-5Table B-5: Summary of Virtual Channel Priorities. Four different levels are expected to be used to differentiate prioritization of network traffic and it would be beneficial to have more internal resolution of priorities inside a Net2Display Node, as THINC does in its transmission queues. Eight different levels of prioritization are expected to be sufficient for prioritizing PDUs.

Table B-5: Summary of Virtual Channel Priorities

Virtual Channel	Preferred Transport Protocol	Priority	Priority Level (7 to 0)	Priority Level (3 to 0)	Source	Inter-active	Real time	In Order Rqd	Example
Pointer	TCP	Very High	7	3	Both	X		X	Mouse, Mouse Clicks & Pointer Icons
Keyboard	TCP	Very High	7	3	Client	X		X	Keyboard
USB Interrupt	TCP	Very High	6	3	Both	X	X		USB mouse or keyboard
Display	TCP	High	5	2	Host			X	Windows
Audio	UDP	High	4	2	Both		X	X	Speaker Out & Microphone
USB Isochronous	UDP	High	3	2	Both	X	X		USB Audio or Video
USB Primary/ Control	TCP	High	2	1	Both				USB Setup and Status transactions
Video	UDP	Medium	1	1	Host		X	X	Video Stream
USB Bulk	TCP	Low	0	0	Both	-			Disk, Flash
Other Virtual Channel	TCP or UDP	Varies	Varies	Varies	Both	Varies		Varies	Vendor Specific

B.8.3 Virtual Channel Setup and Control Header Field Group

B.8.3.1 Control/Data Bit (C Bit) – 1 Bit

The Control/Data bit indicates whether the PDU contains control or data. This field allows any Virtual Channel ID number to be associated with the Control stream, providing flexibility in the assignment of Virtual Channel IDs.

B.8.4 Virtual Channel Identification Header Field Group

B.8.4.1 Virtual Channel ID – 24 Bits

The Virtual Channel ID identifies the traffic in a particular Virtual Channel. The Virtual Channel ID is needed to separate the different types of traffic into different streams. For both TCP and UDP, Virtual Channel numbers are needed to separate different types of data and for assigning different priorities to the data

For supporting a large number of concurrent videos and multiple displays, 8 bits of field yielding 256 Channels is unlikely to be sufficient. Sixteen bits is likely to provide sufficient numbers for future expansion. SCTP also provides for 16 bits of Virtual Channel ID. Twenty-four bits are allocated to provide flexibility in the assignment of Virtual Channel IDs.

B.8.4.2 Reserved Bits (Rsv Bits) – Minimum of 2 Bits

Reserved bits allow additions to the standard as the standard is developed or in future versions. Two bits allow for two additional features to be added. One of the bits could designate an additional header if necessary.

B.8.4.3 Extended Bit (D Bit) – 1 Bit

The Extended bit indicates if the protocol is to be interpreted in a vendor specific manner or in a Net2Display standard specified manner. The Extended Bit should be included in the header because it allows the designation of Vendor specific Virtual Channels. Alternatively, this could be specified only on initial Virtual Channel setup, but including the Extended designation in a bit allows the PDU to be self-describing, allowing it to be interpreted without doing a lookup first.

B.8.4.4 Virtual Channel Protocol Type – 5 bits

The ProtocolCode field indicates the protocol being passed within the PDU. When the Extended bit is set, the ProtocolCode is vendor specific. Including this field allows the protocol to be self describing and requires no lookup to interpret the PDUs. This field is equivalent to the Payload Protocol Identifier field in SCTP.

Currently, there are 9 different static Virtual Channel protocols defined: display, video, audio, keyboard, pointer, USB Primary/Control, USB interrupt, USB synchronous, and USB bulk . Leaving room for sufficient expansion would call for 5 bits. Alternatively, this could be specified only on initial Virtual Channel setup, but including the protocol type, like in SCTP, allows the PDU to be self-described, interpreted without doing a lookup first.

B.8.5 Virtual Channel Specific Control Header Field Group

B.8.5.1 Reserved Bits – Minimum of 2 Bits

Reserved bits allow additions to the standard as the standard is developed or in future versions. Two bits allow for two additional features to be added. One of the bits could designate an additional header if necessary.

B.8.5.2 Response Bit (Rq Bit) – 1 Bit

The Response Bit differentiates requests and responses. This field should be included because it allows response codes to be sent back and associated with requests. This is particularly useful for setting up Virtual Channels or for commands that require resources that may be limited. Alternatively, a different command code could be defined for each different response, which is not as general an approach.

B.8.5.3 Acknowledgment Bit (A Bit) – 1 Bit

The Acknowledgment Bit specifies when additional PDUs have been received and are being acknowledged by the setting of the Ack Sequence Number field.

B.8.5.4 Command Code – 5 Bits

The Command Code specifies the specific command carried by the PDU. The Command Code should be include in the PDU header because it gives an early cue on how to process the PDU data before the data is encountered. Since every PDU contains a Command Code, there is no disadvantage in including it in the header. Alternatively, the Command Code could alternatively be included in the Data portion of the PDU. Currently, there are expected to be at least 5 draw commands. Allowing for an expansion to 4 times this number would require 5 bits to represent. SCTP uses a 16 bit stream sequence number.

B.8.5.5 PDU Length – 16 Bits

While a PDU Length is not needed in UDP packets, it is needed when the PDUs are transported over TCP to separate the TCP byte stream into individual PDUs. The PDU Length is included in both TCP and UDP for providing a consistent interface to the Transport layer. Since TCP is a byte stream interface, it requires a size to be included in the PDUs so that the message boundaries can be located. The PDU Length should be large enough to support Super Jumbo packets of 64000 bytes as will likely be used in the future for higher speed Ethernets.

B.8.5.6 Virtual Channel Sequence Number – 16 Bits

The Virtual Channel Sequence Number indicates the ordering of the PDUs in the stream. The Virtual Stream Sequence Number is mainly used by UDP, which can present PDUs to Net2Display out of order. The Virtual Stream Sequence Number should be included in all PDUs, both TCP and UDP so that the format of all Net2Display PDU headers is the same. UDP can deliver PDUs to Net2Display out of order, so a Sequence Number is useful determining when packets are missing. Alternatively, the VC_Timestamp could just be included in audio and video packets.

B.8.5.7 Ack Sequence Number – 16 Bits

The Ack Sequence Number indicates the Virtual Channel Sequence Number of the PDU being acknowledged by this PDU. The Ack Sequence Number is used for providing flow control.

B.8.5.8 VC_Timestamp – 32 Bits

The VC_Timestamp field is needed in audio and motion video packets to synchronize audio and video and to allow it to be played at an even pace. It prevents out of order and late received packets from being played. The VC_Timestamp should be included in all the headers to allow for consistency in headers and header processing. For both TCP and UDP, a VC_Timestamp field is useful for the real time playing of video and audio to ensure that material is not replayed late. Alternatively, the VC_Timestamp could just be included in audio and video packets. The VC_Timestamp is 32 bits to match the size and processing used by the Real Time Protocol (RTP).

B.8.6 Omitted PDU Header Fields

B.8.6.1 Host Source Bit – 1 Bit

The Host Source Bit would indicate when PDUs are coming from a Host instead of a Client (or possibly CMS server) Since the same pair of nodes cannot be a Host and Client in the opposite direction at the same time, this bit is not really needed.

B.8.6.2 Sub-Channel ID Number – 8 Bits

Sub-Channel ID Numbers could have been used to differentiate multiple displays or multiple video streams. Since the same protocol can be used for different Virtual Channels, Virtual Channel numbers can be used to provide the same function.

B.8.7 Virtual Channels IDs

B.8.7.1 Which End Should Initialize the Formation of a Virtual Channel?

For consistency, the same end, Host or Client, should always initialize all Virtual Channels. This prevents 1) either race conditions of two Virtual Channels being formed in opposite directions with the same Virtual Channel ID or 2) having to add a bit in the header to indicate when end initiated the Virtual Channel. A request packet can be defined to allow a Client to request a Host to initialize a Virtual Channel.

Recommendation: Host initializes all Virtual Channels. A Client can request the initialization of a Virtual Channel. RDP also initiates it at the Host end.

B.8.7.2 Which End Should Choose the Virtual Channel ID?

The end that chooses the Virtual Channel ID can perform a quicker lookup on a incoming packet if the Virtual Channel IDs are chosen either sequentially or according to the associated Virtual Machine. It is beneficial for the Host to assign Virtual Channel IDs to more easily associate them with VMs. It may be beneficial for the Client to assign Virtual Channel IDs because the Client will typically be receiving larger volumes of traffic.

Options:

1) Host chooses – allows Host do a quicker lookup on reception of Net2Display traffic and to separate traffic by VM.
2) Client chooses – allows Client to do a quicker lookup on reception. The predominant flow of traffic will be from Host to Client
3) Host chooses portion and Client chooses portion – This could allow both the Host and the Client to choose a unique identifier that allows a direct lookup. For example, the Host could choose the first 16 bits and the Client set the last 8 bits of a 24 bit Virtual Channel ID. This would allow 64k unique identifiers assigned by the Host and 256 assigned by a Client.

Whichever approach is choosen, it needs to be done in a standard specified manner.

Decision: Will be specified at a later time.

B.8.7.3 How Large Should the Virtual Channel ID be?

This depends on which end chooses the Virtual Channel ID and whether both ends provide a portion of the Virtual Channel ID.

Recommendation: Make Virtual Channel ID to be three bytes to allow for flexibility in assignment

B.8.8 Services provided by Net2Display Transport over UDP

B.8.8.1 Congestion Control

Net2Display Remoting will need to do its own Congestion Control for the UDP traffic to reduce bandwidth when congestion occurs. Bandwidth can be reduced by reducing frames per second, using a better compression algorithm or using lossier compression. Issues:

1) What information should be included in Headers to provide Congestion Control?

To match existing approach in TCP, Net2Display Remoting should include an Ack bit and an Ack Sequence Number of 16 bits.

2) How are losses differentiated due to congestion and losses due to wireless links?

B.8.8.2 Reliable Delivery

Once Ack Sequence Numbers have been included in headers, sufficient information is provided in the header for UDP to be used for reliable delivery. Does Net2Display allow UDP to be used for reliable delivery or does it limit the use of the Ack to Congestion Control?

Recommendation: Net2Display can allow optional reliable delivery over UDP, not specified by the standard. This is done as long as the Host and the Client agree to use it.

B.8.9 Should Header Fields Only Used by UDP be Optional?

Fields used only by UDP include the Sequence Number and the Ack Sequence Number. Options are:

1) Always require these fields
2) Always include space for these fields in the header and only require these fields to be filled when UDP is used by a transport. Include a bit to indicate when these fields are valid.
3) Include a bit to indicate when these fields are included and valid and make the UDP fields optional and only included when this bit is set.

Recommendation: Always include space for these fields in the header and only require these fields to be filled when UDP is used by a transport. Include a bit to indicate when these fields are valid.

B.9 Node Association Management (NAM)

B.9.1 TBD: Does Net2Display Work on a Computer with Multiple NICs, Using the High Bandwidth Available Connected Network?

Yes, for the following scenarios:

- Laptop with Ethernet and wireless which switches between the two depending upon which is available
- Dual Network Interface server, which has two Network Interface s for reliability and bandwidth. Net2Display should be able to use whichever of the Network Interface s it chooses.
- Dual NIC Integrated Display, with both Ethernet and wireless Network Interfaces and Net2Display should choose which ever one is connected allowing both wired and wireless operation
- Multiple Ethernet Network Interface PC could be designed for supporting Displays on one Network Interface

Decision: It will need to be able to work on a system with multiple active NICs

B.9.2 Does Net2Display Work with Multiple Active Network Interfaces Simultaneously? Will it Spread out Traffic Across Multiple Network Interfaces?

Decision: Each Association between a Host and a Client must continue to use the same IP addresses as they were established with. Other Associations could use different IP addresses to exit the same computer.

B.9.3 If one NIC goes Down and Network Connectivity is Picked up on Another, can a Net2Display Association Continue on that Other Network Interface?

If a Client loses connectivity and then switches to another NIC and thus switches to another IP address, it could attempt to reestablish the connection with the Host using the new IP address.

This is a feature of RDP called automatic reconnection (http://technet2.microsoft.com/windowsserver/en/library/2cb5c8c9-cadc-44a9-bf39-856127f4c8271033.mspx?mfr=true).

B.9.3.1 RDP Automatic Reconnection

Automatic Reconnection adds resilience to the Remote Desktop Connection client is Windows Server 2003. It is designed to recover from temporary connection losses due to network problems. Automatic Reconnection enables disconnected Terminal Services sessions to automatically re-authenticate to a Terminal Server without prompting the user for credentials.

Mobile users can greatly benefit from this feature. For example, with the previous versions of Terminal Server client, a user working with a wireless laptop which momentarily loses connectivity would be disconnected from the session and greeted with the message “The connection was ended because of a network error. Please try connecting to the remote computer again.” Now with Automatic Reconnection, the user resumes the original session without needing to re-enter a password.

B.9.3.2 RDP Reconnection Process:

The Automatic Reconnection process is managed using a cookie that is sent to the client. When reconnection is initiated on the client, it sends this cookie to the server as a token for validating the connection. The auto reconnection cookie is generated at the server, and is flushed and regenerated any time a user logs in to a session or when a session is reset. This ensures that after the user connects to the session from a different computer, the original computer cannot reconnect. The server also invalidates and updates the cookie at hourly intervals, sending an updated cookie to the client as long as the session is active.

Decision: Automatic Reconnection needs to be supported by Net2Display

B.9.4 When there are Multiple NICs, does the Discovery Process Happen Through all the NICs or Just the One that is Connected with the Best Bandwidth?

Decision: If a PC has multiple NICs and the internet is connected on one and displays are connected to the other, then the PC should do Discovery on both NICs

B.9.5 What if Discovery finds a CMS on one NIC but Desired Display is Connected Directly to the other NIC?

Does the CMS prevent finding the Client Service through the other NIC? Does one just seek services through the NIC where the CMS is found?

B.9.6 Should the Host and Client State Machines be the Same or Unified?

Benefits

- One State Machine can control when a system is acting as both the Host and Client
- The initialization, acquisition of an address and the discovery of CMS is already the same for both Hosts and Clients
- One state machine to verify
- One set of code could be written for Host and Client
- One Source implementation could be one set of code

Disadvantages

- Adds complexity to the state machine
- Makes State machines more difficult to verify
- A Client should be as thin as possible and should not be required to support Host functionality

Could initialization just be made the same?

Could one state machine be copied to form the other?

One option would be to make a common high level machine that launches the other state machines

Decisions:

1. Should be one core state machine that runs on both Clients and Hosts

2. Should be separate state machines for Host and Client function
3. Host machine should cover all Host Modes and Client Machine should cover all Client Modes. State Machines should not be started or stopped simply because a mode changed

B.9.7 Service Discovery: MDNS/DNS SD is Required, UPNP is Optional and Recommended for Home Environment and all other Service Discovery Approaches are Optional

Unanimous Agreement: The Net2Display Task Group voted to require mDNS/DNS SD and to make UPNP optional with a recommendation that UPNP be provided in the home environment and logoed when provided in products. All other service discovery approaches are optional.

The Client may optionally use a number of standard protocols to locate the Host including:

1. Dynamic Host Configuration Protocol (DHCP)
2. Service Location Protocol (SLP)
3. Domain Name Service (DNS)
4. HTTP Redirect
5. Web Services Using XML
6. Net2Display Connection Management Server

Unanimous Agreement: UPnP and Zeroconf will be added as discovery mechanisms for Net2Display Remoting.
Unanimous Agreement: All of the required Discovery mechanisms must be supported for Net2Display Discovery to be interoperable.
Unanimous Agreement: Net2Display will provide a Discovery service that can also be used for setting up RDP and ICA connections.
Unanimous Agreement: Net2Display needs to define default parameters for the setup for Net2Display Clients and Hosts so that systems can find default Connection Management Servers and Hosts with an out of the box configuration.

B.9.7.1 Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) uses a DHCP server to provide configuration parameters specific to the DHCP client host requesting, generally, information required by the client host to participate on an IP network. DHCP also provides a mechanism for allocation of IP addresses to client hosts.

DHCP has a number of unassigned optional parameters. One or more of the unassigned DHCP options could be requested from IANA to provide identification of the Connection Management Server or the Host Server statically for that network or dynamically per network attached device.

B.9.7.2 Service Location Protocol (SLP)

Service Location Protocol (SLP) allows computers and other devices to find services in a local area network without prior configuration. SLP scales from small, unmanaged networks to large enterprise networks. It is defined in RFC 2608. SLP is frequently used for locating printers. Service Location Protocol (SLP) which allows devices to find services, such as printers, works within a local area network, but will not work over wide area networks, leased lines or the Internet.

SLPv2 provides extensions to SLP and is designed to work within an enterprise environment

Note: To use SLP Services, Net2Display should request a SLP service type assigned by IANA for Net2Display services usage.

B.9.7.3 Domain Name Service (DNS)

Domain Name Service (DNS) provides a correspondence between a hostname and IP addresses. One hostname may correspond to many IP addresses, allowing DNS to serve as a load balancing mechanism. DNS works across local area networks, wide area networks, leased lines and the Internet. DNS will pass through most firewalls.

B.9.7.4 HTTP Redirect

HTTP redirect is used by a server to tell a Client that it should go to a different address, which can be used to locate a Host.

B.9.7.5 Net2Display Connection Management Server

Net2Display Connection Management Server may optionally be defined and used for locating the Host. This would be a Connection Management Server defined specifically for the Net2Display standard. This approach would only be used if it provides required functionality or intellectual property advantages that are not provided by existing discovery approaches.

B.9.7.6 Should the Host and Client State Machines be the Same or Unified?

Benefits

- One State Machine can control when a system is acting as both the Host and Client
- The initialization, acquisition of an address and the discovery of CMS is already the same for both Hosts and Clients
- One state machine to verify
- One set of code could be written for Host and Client
- One Source implementation could be one set of code

Disadvantages

- Adds complexity to the state machine
- Makes State machines more difficult to verify

Could initialization just be made the same?

Could one state machine be copied to form the other?

One option would be to make a common high level machine that launches the other state machines

B.9.7.7 Discovery Comparison Table

These different approaches for the location of the Host are compared in Table B-6. It is expected that not all of these Host location methods will be included in the standard. It is expected that one of these approaches that works for all networks will be required and other approaches will be either required, optional or omitted from the Net2Display Standard.

Table B-6: Comparison of Discovery Approaches

Advantages:	Works across WAN and Internet	Existing Standards	No intellectual property issues	Simple to implement	Works independent of Networking Infrastructure
Service Location Protocol (RFC 2608)		X	X	X	
Domain Name Service (RFC 882)	X	X	X	X	
LDAP		X	X	X	
DHCP		X	X	X	
HTTP Redirect	X	X	X	X	X
Web Services using XML messaging with SOAP or REST	X	X	X		X
Net2Display Connection Management Server	X		X	X	X

Note: The IETF Intellectual Rights policy is stated here: http://www.ietf.org/html.charters/ipr-charter.html. Searches on the Intellectual Property associated with particular RFCs can be performed here: https://datatracker.ietf.org/public/ipr_search.cgi.

B.9.7.8 All Steps in the Discovery Process are Required

The Discovery Process is required because it is a standard specified way for a Client to locate a Host. If a Client or Host does not participate in the Discovery Process, then it unnecessarily constrains a Net2Display administrator or user to use manual configuration for Associating Clients and Hosts.

All steps are required. The Discovery Process is sufficiently simple that any additional complexity that is present due to each of the Discovery mechanisms is far outweighed by the interoperability with existing resource discovery previously configured.

B.9.7.9 Can a Host Locate and Associate with a Client and if so, How?

It may be useful for a Host to locate and Associate with Clients in the following configurations:

1. Secondary Clients may need to be joined with a Primary Client and may have no associated I/O to complete a user sign in.
2. Display wall element Clients may not have attached I/O and so Association from the Host end may be preferred. This could be preconfigured for Display wall elements.

The disadvantages of allowing a Host to locate and Associate with a Client are the following:

1. This is inconsistent with the usual process of a Client locating and Associating with a Host
2. There is a risk that a display wall could be hijacked by another Host located on the network
3. A log in session on a Client could be spoofed by a malicious Host and login authentication information could potentially be obtained

Decision: Have Host inform the Client to connect to the Host, providing consistency so that Associations are always formed from Clients to Hosts.

B.9.7.10 The Exchange of Configuration Information is Required

Yes. Not requiring the exchange of Configuration Information would severely limit interoperability of future versions of the standard.

B.9.8 Name Resolution without DNS

Agreement: At the May 10, 2007 meeting, it was agreed to use mDNS for Service Discovery, so mDNS will be available for name resolution.

There are two very similar ways of figuring out which networked item has a certain name when DNS is not present. Apple Computer's Multicast DNS (mDNS) is in use, and is published freely, though not by a standardization body. Microsoft's Link-local Multicast Name Resolution (LLMNR) is little used and was rejected as an IETF standard due to excessive contention.

The two protocols have minor differences. mDNS allows a network device to choose a domain name in the ".local" namespace and announce it using a special multicast IP address. This introduces special semantics for the .local namespace, which is considered a problem by some members of the IETF (http://www1.ietf.org/mail-archive/web/ietf/current/msg37126.html) The current LLMNR draft allows a network device to choose any domain name, which is considered a security risk by some members of the IETF. mDNS is compatible with DNS-SD as described in the next section, while LLMNR is not.

B.9.8.1 mDNS

mDNS is that name resolution provided by ZeroConf and is an abbreviation for Multicast DNS. Multicast DNS suffers from a number of scalability issues (imagine asking for a printer in a building with 200 printers) that Simple Service Discovery Protocol, in theory, was intended to work around. In reality Simple Service Discovery Protocol never added in the critical proxy functionality so in practice the difference is not that large.

B.9.8.2 LLMNR

Link-local Multicast Name Resolution is the name resolution proposed by Microsoft. It was rejected as an IETF standard due to lack of agreement on behavior compared to mDNS.

B.9.8.3 Security Note

Confusion between security issues and namespace separation can cause problems. In peer-to-peer name resolution protocols, it is possible for a responder to demonstrate ownership of a name, via mechanisms such as DNSSEC. It is also possible for a responder to demonstrate membership in a trusted group, such as via TSIG or IPsec. If DNSSEC is available, spoofing attacks are not possible, and querying for FQDNs does not expose the sender to additional vulnerabilities. Both the mDNS and LLMNR specifications agree on this point.

B.9.8.4 Name Resolution Comparison

Table B-7: Name Resolution Approach Comparison Table

Name	mDNS	LLMNR
Advocate	Apple	Microsoft
Discovery Mechanism	multicast DNS and DNS Service Discovery	
Attributes	Local lookups should be distinguishable from global lookups and accomplishes this through the use of a special local domain (.local).	Makes local lookup mechanism an extension of the DNS
Disadvantages	1) Multicast turned into Broadcast by many switches	1) Multicast turned into Broadcast by many switches 2) Cannot Use DNS level security 3) Security concerns that applications on hosts that use LLMNR cannot tell if a response has come from the DNS or from a local lookup. Attacks can substitute local lookup for DNS lookup. (LLMNR could have used .local to differentiate this) 4) LLMNR, while describing the use of special sub-trees, does not prescribe the special sub-tree that implementations should use. Some implementations might designate the single-label space as special; others might use ".local".
Advantages	The difference between mDNS and LLMNR is not in their lookup of global names, but that mDNS *also* designates a special sub-tree of the namespace where users explicitly have different security expectations.	
License	Open Source	
Implementation (example)	Bonjour, Avahi	Windows CE 5.0
Attributes searchable	no (subtypes only)	
Directory Support	yes (optional)	
Service Announcement	Multicast Announce or DNS Update	
Service Registration Lifetime	TTL for RR	
Access to Service	Service type (protocol, port)	

B.10 Service Discovery

The following could be used for Net2Display Service Discovery:

1) DNS Service Discovery (DNS-SD) using DNS SRV records
2) Universal Plug and Play (uPNP) Simple Services Discovery Protocol (SSDP)
3) Service Location Protocol (SLP)
4) Search for a well-known string defined for Net2Display using conventional DNS lookup

B.10.1 DNS Service Discovery (DNS-SD)

DNS Service Discovery (DNS-SD) is a way of using standard DNS programming interfaces, servers, and packet formats to browse the network for services. DNS Service Discovery is compatible with, but not dependent on, Multicast DNS. It is used in Apple products, many network printers and a number of third party products and applications on various operating systems. It is considered simpler and easier to implement

than Microsoft's competing technology, SSDP, because it uses DNS rather than HTTP. It uses DNS SRV (RFC 2782), TXT, and PTR records to advertise Service Instance Names. The hosts offering the different services publish details of available services like instance, service type, domain name and optional configuration parameters. Service types are given informally on a first-come basis. A service type registry is maintained and published by DNS-SD.org. IANA list of assigned port names and numbers are here http://www.iana.org/assignments/port-numbers. See http://www.dns-sd.org/ServiceTypes.html for instructions on acquiring an IANA assigned port name.

DNS SRV Records: Currently, one must either know the exact address of a server to contact it, or broadcast a question. The SRV RR allows administrators to use several servers for a single domain, to move services from host to host with little fuss, and to designate some hosts as primary servers for a service and others as backups.

Clients ask for a specific service/protocol for a specific domain (the word domain is used here in the strict RFC 1034 standard sense), and get back the names of any available servers. In general, it is expected that SRV records will be used by clients for applications where the relevant protocol specification indicates that clients should use the SRV record. Such specification MUST define the symbolic name to be used in the Service field of the SRV record as described below. It also MUST include security considerations. Service SRV records SHOULD NOT be used in the absence of such specification.

DNS SRV Example: If a SRV-cognizant LDAP client wants to discover a LDAP server that supports TCP protocol and provides LDAP service for the domain example.com., it does a lookup of

_ldap._tcp.example.com

as described in [ARM]. The example zone file near the end of this memo contains answering RRs for an SRV query.

Note: LDAP is chosen as an example for illustrative purposes only, and the LDAP examples used in this document should not be considered a definitive statement on the recommended way for LDAP to use SRV records. As described in the earlier applicability section, consult the appropriate LDAP documents for the recommended procedures.

DNS SRV Usage

SRV records are often used by Microsoft Windows 2000 clients to find the Domain Controller for a given service - which is controversial as it is not standardized usage. Further, SRV records are common in conjunction with the following standardized protocols:

- XMPP (Jabber)
- SIP
- LDAP

B.10.2 Universal Plug and Play Simple Service Discovery Protocol (uPNP) (SSDP)

Simple Service Discovery Protocol (SSDP) is a UPnP protocol, used in Windows XP and several brands of network equipment. SSDP uses HTTP notification announcements that give a service-type URI and a Unique Service Name (USN). Service types are regulated by the Universal Plug and Play Steering Committee.

SSDP is supported in many SOHO firewall appliances, where host computers behind it may pierce holes for applications. It is also used in media center systems, where media exchange between host computers and the media center are facilitated using SSDP.

Simple Service Discovery Protocol (SSDP) is an expired IETF Internet draft by Microsoft and Hewlett-Packard. SSDP is the basis of the discovery protocol of Universal plug-and-play. SSDP provides a mechanism through which network clients can use to discover network services. Clients can use SSDP with

little or no static configuration. SSDP provides multicast discovery support, server-based notification, and discovery routing. SSDP uses XML UDP unicast and multicast packets to advertise their services. The multicast address is 239.255.255.250. SSDP uses port 1900.

B.10.3 Service Location Protocol (SLP)

Service Location Protocol (SLP), the only protocol for service discovery to have reached the IETF RFC status, is usually ignored by large vendors, except Hewlett-Packard's network printers, Novell, Sun Microsystems, and Apple Computer. SLP is described in RFC 2608; it is not yet an IETF Standard or Draft Standard, although implementations are available for both Solaris and Linux.

SLP Blocking by Routers: A router can be configured to block all SLP traffic or to block SLP multicast. In either case, a device using SLP will not be able to locate devices on the other side of the router. In order to use devices on the other side of a router that does not pass SLP traffic, one must specify the address or name of the device explicitly. Also, PPP (point-to-point protocol) does not pass SLP traffic, so one cannot locate devices using SLP over a dialup connection.

B.10.4 Search for a Well-known String using Conventional DNS Lookup

In order for a Host device to be managed, it has to be discovered and known to the connection broker. Discovery can be done in three ways:

- Network scan: Scan the network looking for devices. Currently multicast, SLP and probing IP addresses can be used. However this will not be allowed on a non-trivial network.
- Static configuration: Each device is human configured to talk to a particular connection broker.
- Dynamic configuration: Each device goes out and finds who to talk to on each (re)boot

Dynamic Configuration: The goal of dynamic configuration is that a device can be plugged into the work without any configuration and automatically work. It can also be physically moved around and still automatically work.

The selected mechanism is to use DNS looking for a known string traversing the hierarchy until a suitable connection broker is found. There are many products already that can manage DNS information for availability and balancing purposes, and all recent DNS servers automatically round-robin the addresses they return. DNS will also handle the introduction to IPv6 easily and can potentially be used to distribute other information in TXT records should it be needed. DNS also typically matches administrative domains within a company, especially if Windows/Active Directory is used.

This example uses cms as the prefix looked for. The actual string used will be agreed later and will be part of the flash, surviving factory resets. The device receives the following via DHCP:

Table B-8: Received DHCP Information

DHCP option	Value
12 (hostname)	h-1-2-3-4.sales.na.example.com
15 (domainname)	sales.na.example.com

Algorithm:

- If there is a static configuration, then try that. Timeout after 30 seconds of being unable to contact a connection broker, and reset.
- Try to use each of these based on the DHCP hostname returned (if there are dots in it)
    - cms.h-1-2-3-4.sales.na.example.com

    - cms.sales.na.example.com
    - cms.na.example.com
    - cms.example.com (stop when only one dot is left in the domain)
- To use each of these based on the DHCP domainname returned
    - cms.sales.na.example.com
    - cms.na.example.com
    - cms.example.com (stop when only one dot is left in the domain)
- Go back to the beginning. This can be done in software, or by a system reset.

DNS queries may return more than one IP address. It is acceptable to only try the first address listed, but desirable to try them all. (If only the first is tried, then a different first address will be returned in the next iteration.)

Considerations: Do factory resets discard static configuration? A good question is if the dynamic configuration is tried after connections using the static configuration fail.

B.10.4.1 Try Dynamic After Static

+ Provides a fallback to the static configuration

+ If the static configuration is incorrect, does not require visiting every device to correct

- Some places may consider the dynamic behavior undesirable/a security risk/counter to their static configuration

B.10.4.2 Only try Dynamic if There is no Static

+ If factory resets clear static configuration, then this mode can always be reached

+ Simpler clearer behavior (easier to document and test)

B.10.4.3 DHCP Options

Extra information about which connection brokers should be used could be passed as DHCP options. This requires managing DHCP servers which may not have the functionality to add arbitrary options. It also makes troubleshooting considerably more difficult since packet sniffers have to be used to work out what has happened at the network level. There are also limited numbers of options available and potential interoperability issues.

B.10.4.4 Service Discovery Comparison

Table B-9: Service Discovery Comparison Table

Name	Multicast DNS/ DNS Service Discovery (mDNS/DNS-SD)	Service Location Protocol Version 2 (SLPv2)	Universal Plug and Play (uPNP) (uses Simple Service Discovery Protocol (SSDP))	Lightweight Directory Access Protocol Version 3 (LDAPv3)	DNS Search
General Approach	DNS Service Records (DNS SRV) are used to store the location of services either on DNS Servers or at the service provider location	User Agents search and retrieve information on Service Agents either directly or through Directory Agents	SSDP client multicasts HTTP UDP discovery request to SSDP multicast channel/Port. SSDP services listen to SSDP multicast channel/Port to hear such discovery requests. If a SSDP service hears a HTTP UDP discovery request that matches the service it offers, then it responds with a unicast HTTP UDP response.	DNS gives location of LDAP servers, wich provide directory services	DNS queries using reserved name cms.*.*.*
Discovery within Infra-struc- ture	Uses DNS SRV records stored on DNS servers to specify the location of services	User Agents requests the Directory Agent to provide the list of applicable Service Agents	Uses multicast discovery – No Directory support	DNS gives location of LDAP servers which return service location	DNS query
Discov-ery with-out Infra-struc-ture	Uses multicast DNS requests to retrieve the DNS SRV records from service providers	User Agent multicasts service query using multicast UDP and Service Agents with a match respond	Uses multicast discovery	Not Defined	Not Defined
Stan-dards	Uses DNS-SRV (RFC 2782) and DNS (RFC 1035)	SLP V2 defined in RFC 2608	Expired IETF Draft Service types regulated by uPNP Steering Committee	LDAPv3 defined in RFC 4510	DNS (RFC 1035)
Advan-tages	1. Already a part of DNS 2. DNS already present in infrastructures 3. Local lookups distinguishable from global lookups by use of special local domain (.local).	1. IETF approved service discovery standard 2. Scalability 3. Security 4. Semantically rich advertising with sufficient semantics to enable client side selection without requiring protocol changes client-server	1. Used by WinXP as part of uPnP		1. Looks like conven-tional DNS Query

Disadvantages	1. Multicast turned into Broadcast by many switches	1. Multicast is turned into a Broadcast by many switches 2. SLP is blocked by many routers 3. Lack of wide scale adoption and lack of deployment in many corporate infrastructures 4. Not easily extensible as SLP's design requires extension approval from IANA/IETF. 5. SLP replicates functionality in systems such as LDAP, HTTP, XML, URIs, DTDs, etc. without providing significant benefit over them. 6. Security - SLP use of X.509 requires complicated administrative structure, prevents decentralized implementation and requires use of ASN.1. 7. Scalability – SLP provides no mechanisms to prevent network overload when many SLP enabled clients and services exist on the same network without a directory. 8. Scalability of restart after power outage 9. Requires Additional Server Service	1. No use of Directory Server to lessen multicast traffic. Always uses multicast 2. Multicast turned into Broadcast by many switches 2. More complicated to implement than mDNS/DNS-SD or SLPv2	1. No discovery without infrastructure	1. Not Architected as Service Discovery means 2. No discovery without infrastructure
Advocate	Apple	IETF	Microsoft, HP		
Example	Bonjour, Avahi	OpenSLP	Windows		

B.11 Identification and Configuration Facilities

B.11.1 Web Services Using XML will use SOAP

Unanimous Agreement: As SOAP was agreed to be the preferred WSDL approach, SOAP will be used for the exchange of Net2Display Configuration information.

SOAP: originally an acronym for Simple Object Access Protocol, SOAP is a protocol for exchanging XML-based messages over a computer network, normally using HTTP. SOAP forms the foundation layer of the Web services stack, providing a basic messaging framework that more abstract layers can build upon.

REST is used to describe any simple web-based interface that uses XML and HTTP without the extra abstractions of MEP-based approaches like the web services SOAP protocol.

It is an on-going investigation whether it is best to use SOAP or HTTP as the binding to communicate the Identification and Configuration Information. The advantages of SOAP are: readily available tools for embedded code generation such as gSOAP (http://sourceforge.net/projects/gsoap2), and potential of use for connection brokering, and usage by a number of companies for similar functions. The attributes of HTTP are: potentially simple to implement, upcoming tool availability, conformance to web Representational State Transfer (REST) model.

B.11.1.1 Configuration Information Exchange Options

Configuration Information can be exchanged using several different means and is necessary for forming an Association between a Host and a Client.

Table B-10: Comparison of Information Exchange Approaches

	WSDL SOAP	WSDL REST	RPC
Protocols Used			
Unidirectional			
Advantages			
Disadvantages	1. SOAP requires communication each way 2. SOAP fails with NAT 3. Denial of service weakness for connections in both directions		

B.12 Virtual Channels

Unanimous Agreement: Net2Display will support the different types of data that is transported using Virtual Channels.

Unanimous Agreement: The Host and the Client will negotiate the number of Virtual Channels. The Host and the Client will express to each other in the Configuration Information the number of channels that they support.

Unanimous Agreement: Video and audio may be transported within the same Virtual Channel to provide synchronization.

Unanimous Agreement 9/12/07: Net2Display will specify our own Virtual Channel approach and will not provide compatibility with RDP or ITU-T T.120.

Table B-11: Virtual Channel Requirements

Virtual Channel	Terminal Client	Display Client	Number	Alternative
Display	Required	Required	One per Display	One could be used for multiple displays
USB Primary/Control	Required	Optional	One	
USB Interrupts	Required	Optional	One	
USB Isochronous	Required	Optional	One	
USB Bulk	Required	Optional	One	
Audio	Required	Optional	One	Audio could be omitted or could be sent over USB.
Video	Optional	Optional	One per video stream	Same information could be sent on the Net Display Virtual Channel as a series of Display images
Keyboard	Optional	Optional	One	
Pointer	Optional	Optional	One	
Vendor Specific	Optional	Optional	One per Vendor Specific Protocol	Defined on a vendor by vendor basis.

Note: On 2/13/08, the above table was adopted as the set of Virtual Channels to be required.

B.12.1 Virtual Channel Remoting Protocol

The Virtual Channel mechanism of Net2Display Remoting is based on ITU-T T.120 standard for data protocols for multimedia conferencing. The following different Protocol Data Units (PDUs) are used to transfer data across the Virtual Channels:

- Connect-Initial - Initial connect request to form connection between Client and Host. Sent by Client to Host. This should include the parameters for the connection in XML format. This should include the Net2Display version number as one of the first parameters.
- Connect-Response – Host responds with the Connect-Response conveying the acceptance ot the connection. This should include the parameters for the Host and potentially the negotiated parameter values
- ErectDomainRequest – Client sends this to set the height of providers in the MCS hieracrchy and throughput enforcement interval. RDP sets both these to one. This PDU s not needed for Net2Display Remoting.
- DisconnectProviderUltimatum – Host sends this to the Client. When received, causes Rdesktop to drop out of sec_rec loop. Essentially used for ending a connection.
- AttachUserRequest – Sent by the Client to attach an individual user to the MCS. This PDU s not needed for Net2Display Remoting.
- AttachUserConfirm – Response to Attach user. This PDU s not needed for Net2Display Remoting.
- ChannelOpenRequest – Request from the Client to the Host to openm a new Virtual Channel
- ChannelOpenConfirm – Response to Channel Open request
- SendDataReqeust – Send data
- SendDataIndication – Indicate that data received

B.13 Made Coordinates 32 bits

- Made Display and Pointer commands use 32 bits for addressing. This was done only because 32 bits did not cause excessive overhead beyond 16 bit addresses. The overhead would be highest on Pointer Packet:
    - IPsec TCP Pointer Packet with 16 bit addresses: 14 words of IPsec/TCP/IP Header + 7 Words Data = 21 Words
    - IPsec TCP Pointer Packet with 32 bit addresses: 14 words of IPsec/TCP/IP Header + 9 Words Data = 23 Words
        - 9.5% Larger for 32 bits of address, which is the worst case. Typically, the overhead will be much less on average

B.14 Display Remoting

The Display Remoting command set is determined by the choice of rendering location for the Net2Display architecture. When rendering is performed on the Client, rendering commands are passed across the network, while Host rendering passes updates to the display across the network. The choices in rendering that Net2Display may support are:

1. Client is just a Framebuffer – All of the rendering takes place on the Host. The Display Remoting command set only includes commands that write to the Framebuffer. Host keeps a fully updated display surface.
2. Client has simple update commands – Commands write to Framebuffer, but also can copy from visible and offscreen areas. Host keeps a fully updated display surface.
3. Client has a partial set of the rendering command – Commands perform the simpler render operations on the Client

4. Client has a larger set of the rendering operations – Commands to the Client can perform most of the rendering operations
5. Client has a full set of the rendering commands – Commands to the client can perform all of the rendering operations

While the Client can have a range of different implementations in the amount of rendering supported, the question in the architecture of the protocol is how many rendering commands are defined for remoting in Net2Display? This can be:

1. No rendering commands are defined, only remote Framebuffers are supported
2. Rendering commands are supported, but are optional for implementations
3. A large set of rendering commands are supported and are encouraged for implementation

B.14.1 Criteria for Command Inclusion

How is it determined which commands are included? Are all commands included or are only the most effective commands included? Criteria for including a command are:

- Include commands that improve network bandwidth
- Include commands that improve network bandwidth by more than 1%.
- Include commands that improve display response time
- Include commands that improve total processing

B.14.2 Problem with Allowing Different Rendering Models

One problem with allowing rendering on either the Host or the Client is that one Host design may expect that the rendering is done on the Client and a Client from another manufacturer may expect that the rendering is performed on the Host. Thus, when the Host and Client from these two different manufacturers work together, neither end will assist in rendering and the performance will be greatly degraded. Thus, one rendering model must be chosen, that the rendering is done predominately on the Host or the rendering is done predominately on the Client.

So, the Net2Display standard needs to select between a model of predominately rendering on the Host or predominately rendering on the Client.

B.14.3 Rendering Location: Host or Client or Both?

The advantages of performing minimal rendering on the Client are:

- Simpler Client
- Lower Cost Client
- Simpler Interface to the Client
- Remote Rendering is of marginal benefit

B.14.4 Display Surface at Host End

B.14.5 Display Remoting Commands

Display Remoting is implemented in a number of existing protocols. A comparison of the different commands that are provided for the different thin client remoting protocols and graphics and display interfaces are shown in Table B-12.

Table B-12: Comparison of Commands for Current Graphics and Remoting

Type	Function	Microsoft Windows Remote Desktop (RDP)	Virtual Network Computing (VNC)	VESA DPVL	Columbia Univ. THINC[1]	Windows DDI Interface
Display	Update Pixel Data	DestBlt	Raw	Basic	RAW	DrvCopyBits
	Copy Data from one area of display	ScreenBlt	Copy Rectangle	BitBLT Extended	COPY	DrvCopyBits
	Fill a rectangular region with a single color	Rect	Raw – Foreground Specified	AreaFill Extended	SFILL	DrvBitBlt pbo-> SolidColor
	Fill a region with a pattern	PatBlt	none	Pattern Fill Extended	PFILL	DrvBitBlt
	Fill a region using a bitmap as a stipple in applying foreground and background colors	PatBlt	none	none	BITMAP	DrvBitBlt using psoMask
	Adjust Display Gamma Table	none	none	Gamma Table	none	none
	Select Framebuffer to Display	none	none	Framebuffer Control Extended	none	none
	Other Commands	Line, PolyLine, Text, MemBlt, DeskSave, TriBlt, BitMap Update	none		none	none
Video	Initialize a Video Stream	none	none	1st Video Packet header	INIT	none
	Next Video Frame Packet	none	none	Scene Syncing	NEXT	none
	Ends a Video Stream	none	none	none	END	none
	Move a Video Stream	none	none	Video	MOVE	none
	Change Source size of video stream (Reduces BW, but requires recoding video)	none	none	N/A	SRCSIZE	none
	Change Destination size of Video	none	none	Scaled Video Stream Setup	DSTSIZE	none
	Sync multi-monitors	none		Genlock		none
Cursor	Change Cursor shape	none	none	none	CHANGE	none
	Show or Hide Cursor	none	none	none	SHOWHIDE	none
	Move the cursor	Move	none	none	MOVE	none
	Change the cursor color	Color_Pointer	none	none	COLOR	none
Other	Ring a bell	none	Bell	none	none	none
	Audio	none	none	none	none	none

[1] Ricardo A. Baratto, Jason Nieh and Leo Kim, "THINC: A Remote Display Architecture for Thin-Client Computing", Columbia University Technical Report CUCS-027-04, July 2004

B.14.6 Compatibility

B.14.6.1 The Transport of Existing Remoting Protocols will not be Supported

However, the Net2Display Discovery Process can be used for establishing connections for existing remoting protocols.

B.14.6.2 Will Net2Display Remoting Support the Transport of Negotiated Proprietary Remoting Protocols?

B.15 USB I/O Remoting

- Is there an existing USB remoting model that Net2Display Remoting can use?

B.16 Keyboard and Pointer Remoting

B.17 Audio Remoting

Unanimous Agreement: The default audio for Net2Display will be Stereo PCM
Unanimous Agreement: The Client will provide facilities for mixing different audio streams and will express the number of audio channels that it supports in the Configuration information.

B.18 Compression Facilities

B.18.1 No Compression is Required for Net2Display, but a Compression Approach will be Recommended for Each Type of Data

Unanimous Agreement: No specific compression algorithm is required by Net2Display Remoting. Net2Display will define a recommend compression approach for each type of data (video, audio, bulk data, etc.). Net2Display will define code points for each of the common compression types. Net2Display will allow vendor defined compression types and associated code points. The means for expressing compression code points will be extensible for accommodating future compression schemes.
Unanimous Agreement: The priorities for compression types that will be supported for display data is from 1) uncompressed, 2) lossless and finally appropriately lossy depending on usage.

The breadth of the different compression algorithms that are required for a Net2Display implementation depends upon the decision of whether there will be different classes of Net2Display Hosts or Clients. If there are different classes of Net2Display Hosts or Clients, it is likely that the different levels will have different Video, Audio and I/O capabilities, so that all different types of compression may not be required.

The full set of potential compression types that may be required in the Net2Display standard are:

1. Static Image Compression – most useful for static or slowly changing displays. Compression occurs at the Host end and decompression at the Client end.
2. Motion Video Compression – most useful for motion video and potentially useful for 3-D graphics work. Compression occurs at the Host end and decompression at the Client end.
3. Audio Compression – only necessary when audio capabilities are included in the display. Compression occurs at the Host end and decompression at the Client end.
4. Data Compression – used in compression of USB I/O traffic. Due to the bi-directional nature of USB data traffic, data compression and decompression may be needed at both the Host and the Client

B.18.2 Open Issue: Will only one compression approach be specified as required for each data type?

Yes. The reasons are as follows:

1. Each different required compression approach will add licensing costs to Net2Diplay Hosts and Clients
2. Each different required compression approach will add complexity to a Net2Display implementation, without significantly improving performance.

B.18.3 Open Issue: What are the criteria for choosing compression approaches?

There are a number of different criteria that must be considered when selecting the required compression algorithms. These criteria can be ranked from most important to least important in the general order as follows:

1. Standardized – the compression algorithm must be specified by a recognized standards organization so that it can be implemented in an interoperable manner or so that existing implementations exist that can be used.
2. High compression ratio – Expressed in bits per pixel, the compression ratio should be within 25% of the best available compression algorithm.
3. Relatively low complexity – the algorithm should be low enough complexity that it can be decoded in real time using a 300 MHz processor for a display resolution of 1200x1024 and a frame rate of 30 frames per second.
4. Low Licensing Costs – Compression schemes required by the standard should have reasonable and non-discriminatory licensing terms

B.18.4 Open Issue: Which compression approaches are optional?

Other additional optional compression algorithms may be added to the standard or allowed for a vendor to implement in a vendor specific manner. Thus, there are two types of optional compressions that may be added to the standard, optional specified compression approaches and vendor specific compression not specified in the standard.

So the two questions are:

1. Which optional compressions will be specified by the standard?
2. Are vendor specific compressions to be allowed by the standard?

These questions are separable and will be addressed in the following two sub-sections.

B.18.4.1 Open Issue: Which optional compressions will be specified by the standard?

Additional optional compression approaches will be defined within the standard when they are requested by a vendor and they provide sufficient differentiation from the required standard compression. This differentiation may include:

1. Prevalent standard – When a compression scheme is not required, but it is sufficient prevalent in its usage that it is a dominant means of compression for the selected field, then it will be considered for inclusion
2. Higher Compression Ratio – when a different compression scheme has a significantly better compression ratio, better than 25% improvement, than the required compression approach, then the compression will be considered for inclusion in the standard.

3. Significantly lower complexity – when a different compression scheme has a significantly lower complexity, lower than 2x that of the required approach, then the compression approach will be considered for inclusion in the standard.

Note: A compression approach with lower licensing costs than the required approach will not lead to it being included in the standard as an optional compression means as any licensing fees would have already been paid for the required approach.

B.18.4.2 Open Issue: Are vendor specific compressions to be allowed by the standard?

Advantages:

- Allows extensibility to newer, better compression means not yet defined when the standard comes out
- Allows vendors to differentiate and to innovate within the standard

Disadvantages:

- Could divide the standard by only allowing high performance when the Host and the Client are made by the same vendor
- Other newer, better, optional means of compression could be defined in a later version of the standard
- Standards group does not provide guidance on which compression approaches are optionally included
- Could drive all vendors of the standard to support some optional compression approaches that may only be marginally beneficial
- May not allow interoperability for the highest performance versions of the standard

B.19 Other Features

Unanimous Agreement: Net2Display will provide optional commands that can allow a Client to control the remote play of Audio and Video.
Unanimous Agreement: Net2Display will support Client low power modes similar to the low power modes used by displays with wake on LAN capabilities.
Unanimous Agreement: Net2Display may include a SIP control channel for allowing device to device VoIP depending on an investigation.
Unanimous Agreement: Net2Display needs to define a common attention key, like RDP, for reverting back to the manual Host entry screen.
Unanimous Agreement: Net2Display will provide tags for providing Quality of Service delivery and throttling.

C Appendix: Performance Monitoring Virtual Channel

A separate Virtual Channel may be optionally setup between the Host and the Client for the monitoring of network performance. The Performance Monitoring Virtual Channel is optional and may be optionally supported by Hosts and Clients.

The following must be defined to make the Performance Monitoring complete:

1. Performance Monitoring protocol:
2. Parameters passed between Client and Host to indicate Performance Monitoring capability and whether to form a Performance Monitoring VC
3. Parameters passed between Client and Host on Performance Monitoring VC to exchange performance statistics
4. How do is the interval determined for sending the performance monitoring packets?
5. What can Performance Monitoring be used to adapt? Performance monitoring can change local echo, compression, frame rate, other adaptive communications
6. Does it go in one direction or in both directions?

Note: Performance monitoring is dependent on the operating protocol behavior. As a result, Performance monitoring will be developed as part of the reference design and specified in the next version of this standard.

C.1 Performance Monitoring Protocol

Alternate short and maximum length PDUs

C.1.1 Calculating Bandwidth and Latency

C.2 Adaptations Possible with Performance Monitoring

C.3 Performance Monitoring Parameters

C.4 Performance Statistics

C.5 Performance Monitoring PDUs

The format of Performance Monitoring PDUs is shown in Figure C-1.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x38	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4…	Performance Monitoring Data			

Figure C-1: Performance Monitoring PDU Format

PDU Length: The total size of this PDU, including PDU Header and Command Data

Command Code: The command code for the Performance Monitoring PDU is set to 0x01

Performance Monitoring Data: The performance monitoring test data being transferred

D Appendix: Motion Video Control Virtual Channel

The Motion Video Control Virtual Channel is set up to provide a control stream for a Video Virtual Channel, optionally providing stopping, pausing, forwarding and reversing capabilities.

D.1 Motion Video Control PDUs

The format of Motion Video Control PDUs is shown in

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x38	Command Code	0x00 14	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Motion Video Control Data			

Figure D-1: Motion Video Control PDU Format

PDU Length: The total size of this PDU, including PDU Header and Command Data

Command Code: The command code for the Motion Video Control PDU

Motion Video Control Data: Data associated with the Motion Video Control Command Code

E Appendix: Serial Port Virtual Channel

The Serial Port Virtual Channel is provided for Host systems that rely on serial ports and require their support. The format of Serial Port PDUs is shown in Figure E-1.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x30	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Serial Device Number	Codec Index	
5…	Serial Data			

Figure E-1: Serial Port PDU Format

PDU Length: The total size of this PDU, including PDU Header and Command Data

Command Code: The command code for the Serial Port Transfers is set to 0x01

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Serial Device ID: The Serial Device ID is a 8 bit field that indicates the serial device involved in the communication:

- 0 – No Serial Device ID present
- N – Serial Device Number

Codec Index: The optional Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channl Codex Capability List provided by the Client. The first item in the Virtual Channl Codex Capability List is numbered 1 and 0 indicates that no codec is used

Serial Data: The serial data being transferred and optionally compressed with the designated codec.

F Appendix: Parallel Port Virtual Channel

The Parallel Port Virtual Channel is provided for Host systems that rely on serial ports and require their support. The format of Parallel Port PDUs is shown in Figure F-1.

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	0x00	Virtual Channel ID		
1	0x34	0x01	PDU Length	
2	VC_Timestamp (Optional)			
3	Sequence Number		Received Sequence Number	
4	Reserved Byte	Parallel Device Number	Codec Index	
5…	Parallel Data			

Figure F-1: Parallel Port PDU Format

PDU Length: The total size of this PDU, including PDU Header and Command Data

Command Code: The command code for the Parallel Port Transfers is set to 0x01

Rsv Bits (2 bits): The Reserved Bits (Rsv Bits) are two bits set aside for future control bits.

Parallel Device ID: The Parallel Device ID is a 8 bit field that indicates the serial device involved in the communication:

- 0 – No Parallel Device ID present
- N – Parallel Device Number

Codec Index: The optional Codec Index specifies the codec that was used for encoding the image data at the sending end. The Codec Type is the index in the Virtual Channl Codex Capability List provided by the Client. The first item in the Virtual Channl Codex Capability List is numbered 1 and 0 indicates that no codec is used

Parallel Data: The parallel data being transferred and optionally compressed with the designated codec.

G Appendix: Network Setup for Net2Display Remoting (Informative)

G.1 Setup Network

The networking infrastructure for using Net2Display Remoting needs to be configured for using the Net2Display Remoting capabilities. This includes providing identification of the Net2Display Hosts and may include other network configurations, load balancing and the geographical distributions of Hosts. The setup of an appropriate networking infrastructure is described in Appendix C.

G.2 Distribute Security Keys

The advance configuration also includes the distribution of security keys for setting up a trusted encrypted channel.

G.3 Setup DNS Service Server

G.4 Setup Connection Management Server

G.5 Hosts identify themselves to Discovery Servers

Hosts use a communication channel unspecified by this standard to identify themselves as available for hosting Clients.

H Appendix: XML Descriptions for Example Configurations (Informative)

Note: Example Host and Client XML descriptions are implementation dependent and will be defined as part of the reference design and specified in the next version of this standard.

H.1 Integrated Display Client with Net2Display Interface

H.2 Net2Display Hub Client Supporting one or Multiple Displays

H.3 Net2Display Client Application Running on a Conventional Client PC

H.4 Net2Display Host Supported by Hardware Remoting

H.5 Net2Display Host Supported by Software Remoting

H.6 Net2Display Host Running on a Virtualized System Sharing Network Adaptors

I Appendix: Recommended Security Certificate Authorities and Certificates for Net2Display

Net2Display recommends accepting Certificates sourced from the current list of Certificate Authorities in Firefox, with the omission of the following:

- FreeSSL (free trial certificates offered by RapidSSL)
  C=US, ST=UT, L=Salt Lake City, O=The USERTRUST Network, OU=http://www.usertrust.com, CN=UTN-USERFirst-Network Applications
- RSA Data Security
  C=US, O=RSA Data Security, Inc., OU=Secure Server Certification Authority
- Thawte
  C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, OU=Certification Services Division, CN=Thawte Premium Server CA/emailAddress=premium-server@thawte.com
- verisign.co.jp
  O=VeriSign Trust Network, OU=VeriSign, Inc., OU=VeriSign International Server CA - Class 3, OU=www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign

As of the writing of the Net2Display standard, the above Certificate Authorities still use theMD5 hash, which has been demonstrated to be venerable to attack. See “MD5 considered harmful today: Creating a rogue CA certificate” by Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger at http://www.win.tue.nl/hashclash/rogue-ca/

