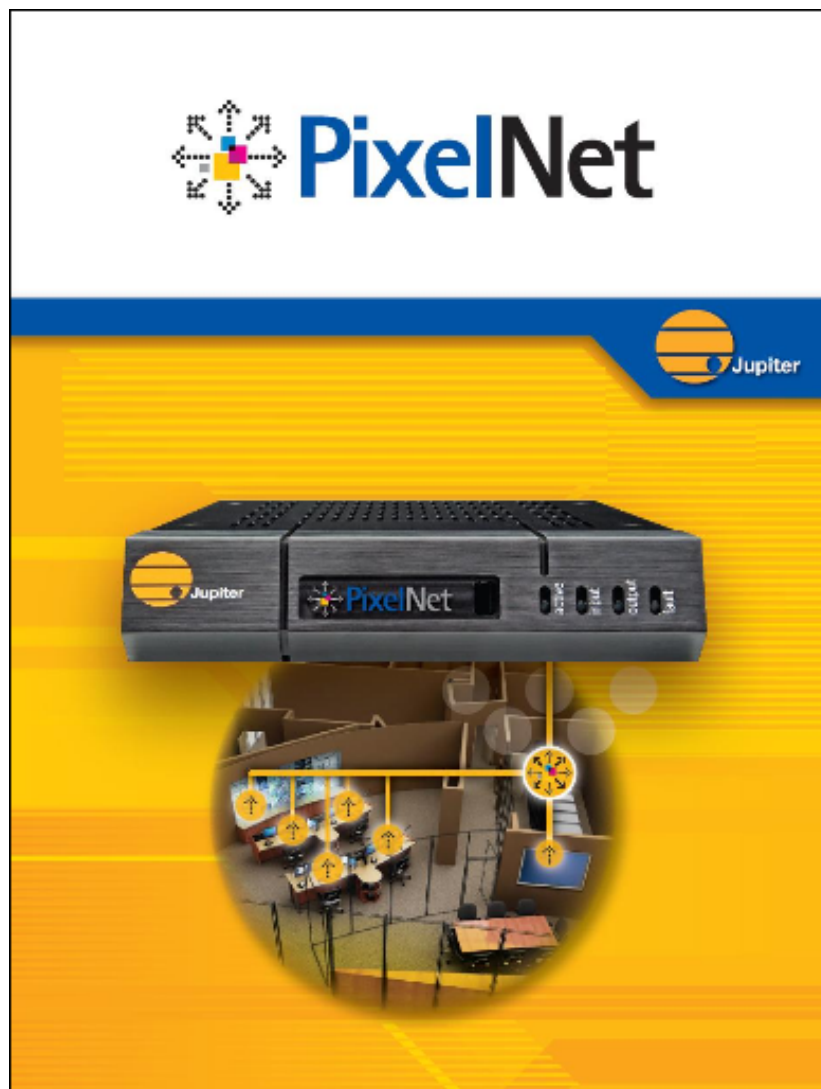


Jupiter Systems

PDC Protocol Programming Manual

Version 1.4



March 15, 2011
A-900-011-03, Rev. E



Copyright

Copyright

Copyright © 2011 Jupiter Systems. This document is copyrighted with all rights reserved.

The entire risk of the use or the result of the use of this Hardware and Software and documentation remains with the User. Information in this document is subject to change without notice. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means, electronic or mechanical, including photocopying without express written permission of Jupiter Systems. See also “**Statement of Limited Warranty**” on page v.

Notice of Regulatory Compliance

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency (RF) energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference. The user is cautioned that changes or modifications not approved by Jupiter Systems can void the user’s authority to operate this equipment.

Warning

PixelNet nodes should be power-cycled if they become unresponsive and PDC software should be re-started to bring node(s) back to an operational state.

Acknowledgements

All non-Jupiter brands and names are the property of their respective owners.

Jupiter, Jupiter logo, and PixelNet are registered trademarks of Jupiter Systems. PDC and TeamMate are trademarks of Jupiter Systems.

Jupiter Systems
31015 Huntwood Avenue
Hayward, CA 94544-7007
510-675-1000 (v)
510-675-1001 (f)
info@jupiter.com
support@jupiter.com
510-675-1007 (v)



Warranty

Statement of Limited Warranty

PixelNet Hardware

Jupiter Systems warrants that the PixelNet Hardware sold by Jupiter are free from defects in material and workmanship and will perform in accordance with the product specification for a period of 24 months from the date of shipment from Jupiter Systems. This warranty is in effect whether the product was purchased directly from Jupiter or through an authorized Jupiter distributor. Any product becoming defective within the time period specified will be repaired or replaced, at Jupiter's option and at Jupiter's factory or authorized repair center. The defective product must be returned to Jupiter or to the Jupiter authorized repair center at the expense of the customer. Expense for the return shipment of the product to the customer within the U.S. will be borne by Jupiter.

Products returned to Jupiter must have a Return Merchandise Authorization (RMA) number. To obtain an RMA number contact the Jupiter repair service center at the phone number listed on the Copyright page. **PRODUCTS SHIPPED TO JUPITER WITHOUT A RETURN AUTHORIZATION NUMBER WILL NOT BE ACCEPTED.**

JUPITER'S TOTAL LIABILITY UNDER THIS WARRANTY SHALL BE LIMITED TO THE REPAIR OR REPLACEMENT OF THE DEFECTIVE PRODUCT OR, AT JUPITER'S OPTION, RETURN OF THE PRODUCT TO JUPITER FOR A REFUND OF THE FULL PURCHASE PRICE. THE ABOVE WARRANTY IS THE ONLY WARRANTY APPLICABLE TO JUPITER'S PRODUCTS AND IS THE CUSTOMER'S SOLE AND EXCLUSIVE REMEDY FOR ANY DEFECT IN THE PRODUCTS.

Jupiter does not warrant the product for fitness for any particular purpose or application. Jupiter has no liability for statements of functionality, performance, or configurability beyond the written product specification for the specific Jupiter product. Jupiter shall not be held liable for incidental, indirect, consequential, general or special damages resulting from the use or the inability to use or the failure of a Jupiter product used in any application. No warranty, including this warranty, shall apply to any Jupiter products that have been modified in any way, by any organization other than the Jupiter factory. The warranty is void for products that have been subjected to misuse, improper maintenance, negligence, and/or damage by excessive current, temperature, or accident.

Jupiter neither assumes nor authorizes any representative or other person to assume for Jupiter any other warranty or liability in connection with the sale or shipment of Jupiter products. Jupiter reserves the right to make changes or improvements in its products without incurring any obligation to similarly alter products previously purchased.

Warranty

Software Warranty and Special Provisions

Limited Warranty

Jupiter Systems warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of ninety (90) days from the date of sale. Any implied warranties on the SOFTWARE are limited to ninety (90) days.

Customer Remedies

Jupiter Systems' entire liability and your exclusive remedy shall be, at Jupiter Systems' option, either (a) return of the price paid, or (b) repair or replacement of the SOFTWARE that does not meet this Limited Warranty and which is returned to Jupiter Systems with a copy of your receipt or purchase order number. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

No Other Warranties

To the maximum extent permitted by applicable law, Jupiter Systems disclaims all other warranties, either expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE and the accompanying written materials.

No Liability for Consequential Damages

To the maximum extent permitted by applicable law, in no event shall Jupiter Systems or its suppliers be liable for any damages whatsoever (including without limitation, special, incidental, consequential, or indirect damages for personal injury, loss of business, profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if Jupiter Systems has been advised of the possibility of such damages. In any case, Jupiter Systems' entire liability under any provision of this agreement shall be limited to the amount actually paid by you for the SOFTWARE.

Using this Manual

Using this Manual

Introduction

Chapter titles are at the top of every page to assist you in finding sections.

The Table of Contents is a section, chapter, and heading outline of the manual; the index at the end of the manual provides a comprehensive list of subjects, figures, and tables.

Note and Caution

This manual uses two special entries to get your attention:

- Note
- Caution

These entries are listed in their ascending order of importance. The examples shown are found throughout this manual.

Note	Notes are entries that bring your attention to specific items that you must see, read, and understand before continuing.
-------------	--

Caution	Cautions are entries that alert you to items that may cause the operating system to not operate properly. For instance, tasks that were either done out of sequence or not supposed to be done at all may cause the system to malfunction. Cautions also alert you about physical connections that can cause the system to not operate properly.
----------------	--





Table of Contents

Copyright	iii
Warranty	v
Using this Manual	vii
Chapter 1. Introduction	1
1.1 Network Connection	1
1.1.1 Getting Connected – Ethernet – Windows XP	1
1.1.2 Getting Connected – Ethernet – Windows Vista/Windows 7	2
Chapter 2. Protocol Syntax	5
2.1 Syntax	5
2.2 Request Line	6
2.3 Response Line	6
2.4 Arguments and Data Types	7
Chapter 3. PDC Commands	9
3.1 Configuration Commands	10
3.1.1 QueryConfigurations	10
3.1.2 QueryCurrentConfiguration	11
3.1.3 LoadConfiguration	12
3.1.4 QuerySystemStatus	13
3.2 Wall Commands	14
3.2.1 QueryAllWalls	14
3.2.2 QueryWallInfo	16
3.2.3 ActivateWall	17
3.3 Layout Commands	19

Contents

3.3.1	QueryLayouts	19
3.3.2	LoadLayout	19
3.3.3	LoadLayoutOnWall	20
3.3.4	QueryCurrentLayout	21
3.3.5	QueryActiveLayouts	22
3.3.6	SaveLayout	23
3.3.7	SaveLayoutOnWall	24
3.4	Source Commands	25
3.4.1	CreateDynamicSource	25
3.4.2	DeleteDynamicSource	27
3.4.3	QueryDynamicSources	28
3.4.4	QuerySources	32
3.4.5	GetSource	33
3.4.6	GetSourceFreeCount	34
3.4.7	SelectSource	35
3.5	Window Commands	37
3.5.1	QueryAllWindows	37
3.5.2	GetWindowState	39
3.5.3	SetWindowState	41
3.5.4	CloseWindow	43
3.5.5	CloseMulticastWindow	43
3.5.6	CloseAllWindows	44
3.5.7	NewWindow	45
3.5.8	NewWindowOnWall	47
3.5.9	NewWindowWithID	49
3.5.10	NewMulticastWindow	51
3.5.11	NewMulticastWindowOnWall	53
3.5.12	NewWindowWithIDOnWall	54
3.5.13	QueryWindowMulticastState	56
3.5.14	GetMulticastWindowSize	58
3.5.15	SetMulticastWindowSize	59
3.5.16	GetMulticastWindowLocation	60
3.5.17	SetMulticastWindowLocation	61
3.5.18	FreezeInput	62
3.5.19	SetFrameInfo	63
3.5.20	SetCropInfo	65
3.5.21	SetTitleInfo	67
3.5.22	SetTitleOverlay	70
3.6	Notification Commands	73
3.6.1	RegisterNotification	73
3.6.2	UnregisterNotification	74
3.6.3	EventNotification	75

Contents

3.7 Audio Commands	76
3.7.1 PixelNet AssignSourceToAudioNode	76
3.7.1.1 Unassigning Source	77
3.7.2 PixelNet PlayAudio	77
3.7.3 PixelNet GetVolume	78
3.7.4 PixelNet SetVolume	79
3.7.5 PixelNet GetBalance	80
3.7.6 PixelNet SetBalance	81
3.7.7 PixelNet GetConnector	82
3.7.8 PixelNet SetConnector	83
3.7.9 PixelNet QueryAudioNodeStatus	84
3.7.10 PixelNet QueryAllAudioNodes	85
Appendix A. Error Codes	87
Index of Tables	89
Index	91





Chapter 1—Introduction

1. Introduction

PixelNet Domain Control (PDC) is a utility to manage PixelNet nodes. The PDC Commands are used to manage PixelNet windows on the display walls. PDC protocol is developed on top of the Remote Method Call (RMC) inter-process communication protocol, implemented over a TCP/IP connection. This chapter discusses the PDC Protocol, its syntax, structure, and use.

1.1 Network Connection

Protocol communication can only be established with PDC over the network. There is no access via the PDC's serial port.

1.1.1 Getting Connected – Ethernet – Windows XP

Follow the procedure below to create an Ethernet connection under Windows XP.

1. Start by finding the IP address of the PDC computer. From the **Start** button click **Run** and enter **cmd**. A black and white command window will appear. At the end of the last prompt (>), type the word "ipconfig" and press the **Enter** key. You should see a list with a domain name and then some IP addresses. The first IP address will be the one for your PDC system. Write down this address.
2. Your cursor should be at the end of another prompt. Type the word **telnet**. You will be placed into the telnet program and your prompt will change to "Microsoft Telnet>".

1—Introduction

Note	Telnet is just one of the ways to connect to PDC. Other methods, such as Putty, can also be used.
-------------	---

3. Type **set localecho**, so you can see what you type. Type the command **open <IP address from step 1 above> 30326**.
4. Once connected, press **Enter**.
5. Press **Enter** again (no password).
6. You can now enter one of the + commands available in the PDC Protocol for controlling the display wall - all commands **MUST** start with a plus sign and end with a **Return**.
7. If you do not enter the plus sign, the PDC will disconnect you.
8. Enter a plus and a few random characters and press **Enter** and you should get a returned error message (bad parameter error). Check "[Error Codes](#)" on page 87 for a listing of all error messages.

Once you have connected successfully, you should be able to send a command and see the returned response within the **Telnet** window.

9. In general, to quit the telnet session, type "quit" at the telnet prompt. To exit the black and white command window, type "exit" at the command prompt.

1.1.2 Getting Connected – Ethernet – Windows Vista/Windows 7

Follow the procedure below for getting connected with an Ethernet connection under Windows Vista or Windows 7.

Note	Windows Vista and Windows 7, by default, do not install the Telnet client.
-------------	--

1. Click **Start**, then select Control Panel.
2. Select **Programs and Features**.
3. Select **Turn Windows features on or off**.
4. Select the **Telnet Client** option.
5. Click **OK**. A dialog box will appear to confirm installation. The telnet command should now be available.

Network Connection

6. From the **Start** button click **Run** and enter **cmd**.
7. Get the **IP address**.
8. Run **Telnet**.
9. Type **set localecho**, so you can see what you type.
Type the command **open [IP address] 30326**.
10. Once connected, press **Enter**.
11. Press **Enter** again (no password).
12. You can now enter one of the + commands available in the PDC Protocol for controlling the display wall - all commands **MUST** start with a plus sign and end with a **Return**.
13. If you do not enter the plus sign PDC will disconnect you.
14. Enter a plus and a few random characters and press **Enter** and you should get a returned error message (bad parameter error). Check ["Error Codes" on page 87](#) for a listing of all error messages.





Chapter 2—Protocol Syntax

2. Protocol Syntax

The RMC protocol is a text-based streaming protocol that allows peers to invoke methods of objects and receive asynchronous notifications. For each **request** there is a **response**. Requests and responses are synchronous and a new command can not be issued before the current call completes.

2.1 Syntax

The **PDC** protocol is text-based. Each command line terminates with **CR/LF**. Tokens are space separated. **All** identifiers are case-sensitive. There are two types of command lines that can be sent with the protocol. Their first character determines the type of these commands:

1. A request which starts with a plus sign [+]
2. A response which starts with an equal sign [=]

Requests are data going to the PDC from a remote client or device.

Responses are incoming data notifications from PDC. The request-response mechanism is synchronous. Every **Request** requires a **Response**.

Caution A **Response** must be received before the next **Request** is sent.

2—Protocol Syntax

The first character of the command line determines the type – **Request** or **Response**.

Request:

+<object name> <method> [<value list>]

Response:

=<result code> [<value list>]

Any text line not starting with '+' is treated as a network software (telnet) command and processed. For example, entering "connect" without the requisite + sign will be processed as follows:

```
connect <CR><LF>
```

2.2 Request Line

Every command must start with the plus character. This is how PDC knows if the command is to be relayed to the PDC server for processing.

A **Request** command starts with a plus sign [+], followed by PixelNet, [method name], and [method arguments].

Syntax:

+**[ObjectName] [MethodName] [Arguments]**

Command:

+PixelNet SetWindowState 12 {0 0 960 120 720 450 0}

2.3 Response Line

A **Response** starts with an equal sign [=], followed by an 8-digit hex number of the result code, optionally followed by other response data. The result code should be treated as a 32-bit signed integer number in two's-complement form.

If the response is =00000000, it is a success. Anything but the eight-zeroes response is an error.

=00000000 **[Arguments]** (a success response)

=80000000 **[Arguments]** (these are error codes; refer to ["Error Codes" on page 87](#) for more information.)

Arguments and Data Types

2.4 Arguments and Data Types

The Argument list consists of argument tokens separated by a space. The following argument types are supported:

- **Integer number** –
 - a 32-bit integer number in decimal format;
 - a 32-bit integer number in hex format (starts with **Ox**) (Ox123A)
 - a **Decimal Floating Point Number**: Decimal Floating Point Numbers work directly with decimal (base 10) fractions and can avoid the rounding errors that can occur when converting decimal fractions to binary (base 2) fractions.
- **String**, is a value surrounded by quotation marks - " " ("now is the time")

Caution Encoding and Decoding code must be in UTF-8 for strings.

- **Structure**, is a value surrounded by braces - { xx }
- **Array**, represented by a structure with the first field set to the number of elements to follow.
- **Boolean** – Boolean operators are either 0 or 1 (0 = false – 1 = true)
- **Bit Fields** - There are several special data constructs called **bit fields** used in the PDC Protocol. One such set is **Flags**, which are described in the following section and elsewhere.

Note All returned data is in base 10 format (decimal). Entered data may be decimal or base 16 (Hex) signified by the starting characters Ox. For example: Ox12AB3400



Chapter 3—PDC Commands

3. PDC Commands

This chapter discusses the PDC Protocol, its syntax, data structure, and use. The PDC commands are organized in seven sections:

- [Configuration Commands](#)
- [Wall Commands](#)
- [Layout Commands](#)
- [Source Commands](#)
- [Window Commands](#)
- [Notification Commands](#)
- [Audio Commands](#)

3—PDC Commands

3.1 Configuration Commands

3.1.1 QueryConfigurations

QueryConfigurations requests the number and names of saved configuration files. The number of saved configurations is returned, followed by the names of each configuration.

Command:

+PixelNet QueryConfigurations (ref string[] configs)

returns:

= 00000000 {"number of saved configurations" "name of configuration 1" "name of configuration 2"}

Example:

+PixelNet QueryConfigurations

=00000000 {2 "c1" "c2"}

Note The **ref** parameter appears only in **query** or **get** request commands that expect values to be returned.

Table 1: QueryConfigurations

Syntax	+PixelNet QueryConfigurations
Response	=00000000 {2 "c1" "c2"}
Response Explanation	<p>2 = number of saved configurations</p> <p>c1 = name of configuration 1</p> <p>c2 = name of configuration 2</p>

Configuration Commands

3.1.2 QueryCurrentConfiguration

QueryCurrentConfiguration returns the name of the currently active configuration.

Command:

+PixelNet QueryCurrentConfiguration (ref string configName)

returns:

=00000000 "Name of Current Configuration"

Example:

+PixelNet QueryCurrentConfiguration

=00000000 {"Headquarters Configuration"}

Table 2: QueryCurrentConfiguration

Syntax	+PixelNet QueryCurrentConfiguration "Headquarters Configuration"
Response	=00000000 {"Headquarters Configuration"}
Response Explanation	Headquarters Configuration = Name of Current Configuration that is being queried

3—PDC Commands

3.1.3 LoadConfiguration

LoadConfiguration specifies the configuration (by name) to be loaded.

Command:

+PixelNet LoadConfiguration (string ConfigName)

returns:

=00000000

Example:

+PixelNet LoadConfiguration "c2"

=00000000

Table 3: LoadConfiguration

Syntax	+PixelNet LoadConfiguration "c2"
Elements Explanation	c2 = name of the configuration to be loaded
Response	=00000000
Response Explanation	OK response.

Configuration Commands

3.1.4 QuerySystemStatus

Returns one of the following five status indicators:

- =00000000 0
The layout has been loaded and system is okay.
- =00000000 1
In the process of loading a configuration
- =00000000 2
The configuration has been loaded and the system is okay.
- =00000000 3
Currently loading a layout
- =00000000 4
The configuration and layouts were completely loaded, but something went wrong. The user will need to use the PDC UI and logs to figure out what is going on.
- =00000000 5
Firmware is a mismatch. Follow the instructions on the GUI dialog to resolve the Firmware issue.

Command:

+PixelNet QuerySystemStatus (ref int systemStatus)

returns:

=00000000 2

Example:

+PixelNet QuerySystemStatus

=00000000 2

Table 4: QuerySystemStatus

Syntax	+PixelNet QuerySystemStatus
Response	=00000000 2
Response Explanation	The configuration has been loaded and the system is okay.

3—PDC Commands

3.2 Wall Commands

3.2.1 QueryAllWalls

QueryAllWalls requests a list of all the display walls including their name, dimension, and resolution information.

Command:

```
+PixelNet QueryAllWalls (ref TWallInfo[] wallInfoList)
```

returns:

```
PixelNet.TWallInfo wallInfoList
```

Example:

```
+PixelNet QueryAllWalls
```

```
=00000000 { 3 { 12 "Wall1" 2 2 1680 1040 } { 13 "Wall2" 2 1 1680  
1040 } { 14 "Wall3" 1 1 1920 1200 } }
```

Data Structure

Table 5: Data Structure—TWallInfo

Structure Name	Type	Elements
TWallInfo	integer	wallID
	string	wallName
	integer	dw
	integer	dh
	integer	rw
	integer	rh

Data Structure Elements—TWallInfo

wallID wall ID number

wallName name given to wall

dw X dimension, number of columns (x-axis)

dh Y dimension, number of rows (y-axis)

Wall Commands

rw resolution width of the wall

rh resolution height of the wall

Table 6: QueryAllWalls

Syntax	+PixelNet QueryAllWalls
Response	=00000000 {3 {12 "Wall1" 2 2 1680 1040} {13 "Wall2" 2 1 1680 1040} {14 "Wall3" 1 1 1920 1200}}
Response Explanation	<p>{3 {12 "Wall1" 2 2 1680 1040}}</p> <p>3 = number of walls in the active configuration</p> <p>12 = system-generated wall ID</p> <p>"Wall1" = wall name</p> <p>2 = number of horizontal displays; this is the Screens X value set in the PixelNet Domain Configuration dialog</p> <p>2 = number of vertical displays; this is the Screens Y value set in the PixelNet Domain Configuration dialog</p> <p>1680 = resolution width (rw) set for the wall for a single display</p> <p>1040 = resolution height (rh) set for the wall for a single display</p>

3—PDC Commands

3.2.2 QueryWallInfo

QueryWallInfo requests the name, size, and resolution information for a display wall specified by WallName.

Command:

+PixelNet QueryWallInfo (string WallName, ref PixelNet.TWallInfo WallInfo)

returns:

PixelNet.TWallInfo WallInfo

Example:

+PixelNet QueryWallInfo "Wall2"

=00000000 { 13 "Wall2" 2 1 1680 1040}

Data Structure

Table 7: Data Structure—TWallInfo

Structure Name	Type	Elements
TWallInfo	integer	wallID
	string	wallName
	integer	dw
	integer	dh
	integer	rw
	integer	rh

Data Structure Elements—TWallInfo

wallID wall ID number

wallName name given to wall

dw X dimension, number of columns (x-axis)

dh Y dimension, number of rows (y-axis)

rw resolution width of the wall

rh resolution height of the wall

Wall Commands

Table 8: QueryWallInfo

Syntax	+PixelNet QueryWallInfo "Wall2"
Response	=00000000 {13 "Wall2" 2 1 1680 1040}
Response Explanation	<p>13 = system-generated wall ID</p> <p>"Wall2" = wall name</p> <p>2 = display width (dw) or number of horizontal displays; this is the Screens X value set in the PixelNet Domain Configuration dialog</p> <p>1 = display height (dh) or number of vertical displays; this is the Screens Y value set in the PixelNet Domain Configuration dialog</p> <p>1680 = resolution width (rw) set for the wall for a single display</p> <p>1040 = resolution height (rh) set for the wall for a single display</p>

3.2.3 ActivateWall

ActivateWall requests that a display wall specified by wall name be activated.

Command:

+PixelNet ActivateWall (string WallName)

returns:

=00000000

Example:

+PixelNet ActivateWall "Wall2"

=00000000

3—PDC Commands

Table 9: ActivateWall

Syntax	+PixelNet ActivateWall "Wall2"
Elements Explanation	Wall2 = name of the wall to be activated
Response	=00000000
Response Explanation	OK response

Layout Commands

3.3 Layout Commands

3.3.1 QueryLayouts

QueryLayouts specifies a count of layouts and a list of each layout by name.

Command:

+PixelNet QueryLayouts (ref string[] layoutList)

returns:

=00000000

Example:

+PixelNet QueryLayouts

=00000000 {5 "Layout1" "Layout2" "Layout3" "Layout4" "Layout5"}

Table 10: QueryLayouts

Syntax	+PixelNet QueryLayouts
Response	=00000000 {5 "Layout1" "Layout2" "Layout3" "Layout4" "Layout5"}
Response Explanation	5 = number of discovered layouts "Layout1" to "Layout5"= name of each saved layout

3.3.2 LoadLayout

LoadLayout specifies the layout (by name) to be loaded on the currently active display wall. Refer to ["ActivateWall" on page 17](#) for more information.

Command:

+PixelNet LoadLayout (string layout name)

returns:

=00000000

3—PDC Commands

Example:
+PixelNet LoadLayout "Layout2"
=00000000

Table 11: LoadLayout

Syntax	+PixelNet LoadLayout "Layout2"
Elements Explanation	Layout2 = the specific layout to be loaded
Response	=00000000
Response Explanation	OK response

Note The **ActivateWall** command may need to be invoked before the **LoadLayout** command can be used.

3.3.3 LoadLayoutOnWall

LoadLayoutOnWall specifies the layout (by name) to be loaded on a specific wall designated by a Wall ID.

Note When a layout is loaded with sources (either streaming or node) that are no longer in the source list, a failed error code (=000000001) will be returned. However, the rest of the windows in the layout will be loaded.

Command:
+PixelNet LoadLayoutOnWall (int wallID, string layout name)

returns:
=00000000

Layout Commands

Example:
+PixelNet LoadLayoutOnWall "Layout2" "Wall2"
=00000000

Table 12: LoadLayoutOnWall

Syntax	+PixelNet LoadLayoutOnWall "Layout2" "Wall2"
Elements Explanation	<p>Layout2 = the specific layout to be loaded</p> <p>Wall2 = the specific wall designated by name on which the specific layout must be loaded</p>
Response	=00000000
Response Explanation	OK response

Note The **ActivateWall** command may need to be invoked before the **LoadLayout** command can be used.

3.3.4 QueryCurrentLayout

Returns the name of the currently active layout.

Command:
+PixelNet QueryCurrentLayout (ref string wallName, string layoutName)

returns:
=00000000 "Name of Current Layout"

Example:
+PixelNet QueryCurrentLayout
=00000000 {"Seventh Layout"}

3—PDC Commands

Table 13: QueryCurrentLayout

Syntax	+PixelNet QueryCurrentLayout "Seventh Layout"
Response	=00000000 {"Seventh Layout"}
Response Explanation	Seventh Layout = Name of current layout

3.3.5 QueryActiveLayouts

Returns the name of the currently active layouts.

Command:

+PixelNet QueryActiveLayouts (refTLayoutInfo[] LayoutInfos)

returns:

=00000000 "Name of Active Layouts"

Example:

+PixelNet QueryActiveLayouts

=00000000 3 { "Wall1" "Layout 1" 0} {13 "Wall2" "Layout for Wall 2" 0} { "Wall3" "Layout 3" 1 }

Data Structure

Table 14: Data Structure—TLayoutInfo

Structure Name	Type	Elements
TLayoutInfo	string string	wallName layoutName

Layout Commands

Table 15: QueryActiveLayouts

Syntax	+PixelNet QueryActive Layouts
Response	=00000000 3 { "Wall1" "Layout 1" 0} { 13 "Wall2" "Layout for Wall 2" 0} { "Wall3" "Layout 3" 1 }
Response Explanation	3 = number of discovered active layouts "Layout 1" to "Layout 3" = name of each saved layout

3.3.6 SaveLayout

SaveLayout specifies the layout (by name) to be saved on the active display wall. Refer to ["ActivateWall" on page 17](#) for more information.

Command:

+PixelNet SaveLayout (string FileName)

returns:

=00000000

Example:

+PixelNet SaveLayout "Layout12"

=00000000

Table 16: SaveLayout

Syntax	+PixelNet SaveLayout "Layout12"
Elements Explanation	Layout12 = the specific layout to be saved
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.3.7 SaveLayoutOnWall

SaveLayoutOnWall specifies the layout (by name) to be saved on a specific wall designated by a Wall ID.

Command:

+PixelNet SaveLayoutOnWall (string FileName, int wallID)

returns:

=00000000

Example:

+PixelNet SaveLayoutOnWall "Layout2" "Wall2"

=00000000

Table 17: SaveLayoutOnWall

Syntax	+PixelNet SaveLayoutOnWall "Layout2" "Wall2"
Elements Explanation	Layout2 = the specific layout to be saved Wall2 = the designated wall where the layout must be saved
Response	=00000000
Response Explanation	OK response

Source Commands

3.4 Source Commands

3.4.1 CreateDynamicSource

CreateDynamicSource requests a count of saved Dynamic Sources and a list of each Source path to Sources folder (including all the subfolders under it).

Command:

+PixelNet CreateDynamicSource (string path)

returns:

ref SourceNodeAttribute[] items

Example:

+PixelNet CreateDynamicSource "name=My Custom Live Source;protocol=Mpeg4RTP;IP=192.168.20.81;port=49152"

=00000000

DataStructure

Table 18: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string	name
	string	protocol
	string	ip
	string	payloadtype
	string	port
	string	over
	string	impathuserdata
	string	enablertcp
	string	decoderconfig
	string	url

3—PDC Commands

Data Structure Elements—SourceNodeAttribute

name	name of Dynamic Source
protocol	type of streaming protocol
ip	source ip address
payloadtype	This is the Payload Type for the RTP video stream. If the RTP payload type is unknown, use a default value of -1.
port	defines the encoder port
over	dictates which connection protocol to use, i.e. play RTSP over UDP (default), over HTTP, over TCP
impathuserdata	returns video stream user data inserted by impath encoder
enablertcp	enable rtcp protocol
decoderconfig	config data provided by encoder to prepend to video frame
url	path to the video stream

Table 19: CreateDynamicSource

Syntax	+PixelNet CreateDynamicSource "name=My Custom Live Source;protocol=Mpeg4RTP;IP=192.168.20.81;port=49152"
Response	=00000000
Response Explanation	OK response

Source Commands

3.4.2 DeleteDynamicSource

DeleteDynamicSource deletes the Dynamic Source named in the command.

Command:

+PixelNet DeleteDynamicSource (string name)

returns:

=00000000

Example:

+PixelNet DeleteDynamicSource "My Custom Live Source"

returns:

=00000000

Table 20: SaveLayoutOnWall

Syntax	+PixelNet DeleteDynamicSource "My Custom Live Source"
Elements Explanation	My Custom Live Source = the specific Dynamic Source to be deleted
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.4.3 QueryDynamicSources

QueryDynamicSources requests a count of saved Dynamic Sources and a list of each Source path to Sources folder (including all the subfolders under it).

QueryDynamicSources command can be issued with three different filters for distinctly different results:

- **+PixelNet QueryDynamicSources "All"**
"All" is the filter which returns all the dynamic sources.
- **+PixelNet QueryDynamicSources "Recent"**
"Recent" means this node has been discovered since the last configuration load. This is the default filter.
- **+PixelNet QueryDynamicSources "NotSavedInConfig"**
"NotSavedInConfig" means not present in the static sources in the current configuration.

Command:

+PixelNet QueryDynamicSources (string path)

returns:

ref SourceNodeAttribute[] items

Example:

+PixelNet QueryDynamicSources "All"

```
=00000000 { 2 { "My DynamicSource1" "/MyDynamicSource1" }
{ "My DynamicSource2" "/MyDynamicSource2" } }
DataStructure
```

Table 21: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string string	name fullpath

Source Commands

Data Structure Elements—SourceNodeAttribute

name name of saved Dynamic Source

fullpath "/" generates the full source path

Table 22: QueryDynamicSources "All"

Syntax	+PixelNet QueryDynamicSources "All"
Response	=00000000 {2{"My DynamicSource1" "/" MyDynamicSource1"}} {"My DynamicSource2" "/" MyDynamicSource2"}}
Response Explanation	2 = number of saved Sources "MyDynamicSource1" = name of saved Dynamic Source "/MyDynamicSource1" = full path to Dynamic Source (including all the subfolders under it)

3—PDC Commands

Command:

+PixelNet QueryDynamicSources (string path)

returns:

ref SourceNodeAttribute[] items

Example:

+PixelNet QueryDynamicSources "Recent"

=00000000 { 2 { "DVI 10" "" } {"HD 5E" " " } }

DataStructure

Table 23: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string string	name path

Data Structure Elements—SourceNodeAttribute

name name of saved Dynamic Source

path path of source

Table 24: QueryDynamicSources "Recent"

Syntax	+PixelNet QueryDynamicSources "Recent"
Response	=00000000 { 2 { "DVI 10" "" } {"HD 5E" " " } }
Response Explanation	<p>2 = number of recent Sources</p> <p>"DVI 10" = name of recent Dynamic Source since configuration load</p> <p>"HD 5E" =name of recent Dynamic Source since configuration load</p>

Source Commands

Command:

+PixelNet QueryDynamicSources (string path)

returns:

ref SourceNodeAttribute[] items

Example:

+PixelNet QueryDynamicSources "notsavedinconfig"

=00000000 { 2 { "DVI 4" "" } {"HD 8E" " " } }

DataStructure

Table 25: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string string	name path

Data Structure Elements—SourceNodeAttribute

name name of Dynamic Source not saved in configuration

path path of source

Table 26: QueryDynamicSources "NotSavedInConfig"

Syntax	+PixelNet QueryDynamicSources "NotSavedInConfig"
Response	=00000000 { 2 { "DVI 4" "" } {"HD 8E" " " } }
Response Explanation	<p>2 = number of Dynamic Sources not saved in configuration</p> <p>"DVI 4" = name of Dynamic Source not saved in configuration</p> <p>"HD 8E" = name of Dynamic Source not saved in configuration</p>

3—PDC Commands

3.4.4 QuerySources

QuerySources requests a count of saved Sources and a list of each Source path to Sources folder (including all the subfolders under it).

Command:

+PixelNet QuerySources (string path)

returns:

ref SourceNodeAttribute[] items

Example:

+PixelNet QuerySources "/"

=00000000

{2{ "My Source1" "/MySource1"} {"My Source2" "/MySource2"} }

Data Structure

Table 27: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string string	name fullpath

Data Structure Elements—SourceNodeAttribute

name name

path fullpath of source including subfolders

Source Commands

Table 28: QuerySources

Syntax	<code>+PixelNet QuerySources "/"</code>
Response	<code>=00000000 { 2{ "My Source1" "/MySource1"} {"My Source2" "/MySource2"} }</code>
Response Explanation	<p>2 = number of saved Sources</p> <p>"MySource1" = name of saved Source</p> <p>"/MySource1" = full path to Source (including all the subfolders under it)</p>

3.4.5 GetSource

GetSource returns the name and path of the Source associated with the window specified by window ID.

Command:

`+PixelNet GetSource (uint (wid))`

returns:

`ref SourceNodeAttribute source`

Example:

`+PixelNet GetSource 30010`

`=00000000 { Camera1 "/MySource1" }`

Data Structure

Table 29: Data Structure—SourceNodeAttribute

Structure Name	Type	Elements
SourceNodeAttribute	string string	name fullpath

3—PDC Commands

Data Structure Elements—SourceNodeAttribute

name name
path fullpath

Table 30: GetSource

Syntax	+PixelNet GetSource 30010
Response	=00000000 {"Camera1" "/MySource1"}
Response Explanation	<p>30010 = Window ID</p> <p>Camera1 = name of requested source</p> <p>"/MySource1" = path to Source name</p>

3.4.6 GetSourceFreeCount

GetSourceFreeCount returns a number denoting the maximum number of streams that can still be created on the source before its stream count reaches maximum capacity (i.e. the stream count given in parenthesis--the number of streams available--will show zero).

The query targets the source root folder. If the source is not in the root folder, provide the appropriate path in the command.

Command:

+PixelNet GetSourceFreeCount (string srcName, ref int vacancyNumber)

returns:

=00000000 "number of streams that can still be created"

Example:

+PixelNet GetSourceFreeCount "GroupSource"

=00000000 2

Source Commands

Table 31: GetSourceFreeCount

Syntax	+PixelNet GetSourceFreeCount "GroupSource"
Response	=00000000 2
Response Explanation	GroupSource = name of requested source

3.4.7 SelectSource

SelectSource displays a Source in a window (specified by window ID and Source name) on the active display wall. Refer to [“ActivateWall” on page 17](#) for more information.

Command:

+PixelNet SelectSource (uint wid, string srcName)

returns:

=00000000

Example:

+PixelNet SelectSource 30004 "MySource4"

=00000000

3—PDC Commands

Table 32: SelectSource

Syntax	+PixelNet SelectSource 30004 "MySource4"
Elements Explanation	30004 = system or user generated window ID (uint wid) "MySource4" = name of Source (string srcName)
Response	=00000000
Response Explanation	OK response

Window Commands

3.5 Window Commands

3.5.1 QueryAllWindows

QueryAllWindows requests a list of all the display windows on the wall (with the given name) with their name, state, position, and size information.

Command:

+PixelNet QueryAllWindows (string WallName)

returns:

PixelNet.TWindowState[] WndStates

Example:

+PixelNet QueryAllWindows "Wall 1"

=00000000 {2 {12 0 0 960 120 720 450 0}{11 0 0 0 0 1920 1200 1}}

Data Structure

Table 33: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid window ID number

nState not currently used

Set not currently used

x, y X position, Y position – window position in pixels

3—PDC Commands

w, h width, height – window size in pixels

ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

Table 34: QueryAllWindows

Syntax	+PixelNet QueryAllWindows "Wall 1"
Response	=00000000 {2 {12 0 0 960 120 720 450 0}{11 0 0 0 1920 1200 1}}
Response Explanation	<p>{2 {12 0 0 960 120 720 450 0}} 2 = number of discovered windows 12 = system or user-generated window ID 0 = nState (not currently used) 0 = Set (not currently used) 960 = X-axis position in pixels 120 = Y-axis position in pixels 720 = window width in pixels 450 = window height in pixels 0 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>

Window Commands

3.5.2 GetWindowState

GetWindowState requests the window information such as state, position, size, and ZOrder specified by Window ID.

Command:

+PixelNet GetWindowState (uint wid)

returns:

ref PixelNet.TWindowState { 30002 0 0 960 120 720 450 0}

Example:

+PixelNet GetWindowState 30002

=00000000 { 30002 0 0 960 120 720 450 0}

Data Structure

Table 35: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid window ID number

nState not currently used

Set not currently used

x, y X position, Y position – window position in pixels

w, h width, height – window size in pixels

3—PDC Commands

ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

Table 36: GetWindowState

Syntax	+PixelNet GetWindowState 30002
Response	=00000000 { 30002 0 0 960 120 720 450 0 }
Response Explanation	<p>{ 30002 0 0 960 120 720 450 0 }</p> <p>30002 = system or user generated window ID</p> <p>0 = nState (not currently used)</p> <p>0 = Set (not currently used)</p> <p>960 = X-axis position in pixels</p> <p>120 = Y-axis position in pixels</p> <p>720 = window width in pixels</p> <p>450 = window height in pixels</p> <p>0 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>

Window Commands

3.5.3 SetWindowState

SetWindowState sets the window information such as state, position, size, and ZOrder specified by Window ID.

Command:

+PixelNet SetWindowState (uint wid, PixelNet.TWindowState WS)

returns:

=00000000

Example:

+PixelNet SetWindowState 30002 { 26 0 0 960 120 720 450 0 }

=00000000

Data Structure

Table 37: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid	window ID number
nState	not currently used
Set	not currently used
x, y	X position, Y position – window position in pixels
w, h	width, height – window size in pixels
ZOrder	(0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

3—PDC Commands

Table 38: SetWindowState

Syntax	+PixelNet SetWindowState 30002 {26 0 0 960 120 720 450 0}
Elements Explanation	<p>30002 {26 0 0 960 120 720 450 0}</p> <p>30002 = system or user generated window ID</p> <p>26 = system or user generated window ID (not used)</p> <p>0 = nState (not currently used)</p> <p>0 = Set (not currently used)</p> <p>960 = X-axis position in pixels</p> <p>120 = Y-axis position in pixels</p> <p>720 = window width in pixels</p> <p>450 = window height in pixels</p> <p>0 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.4 CloseWindow

CloseWindow specifies a window (by window ID) to be closed. This command operates on the currently active display wall. Refer to [“ActivateWall” on page 17](#) for more information.

Command:

+PixelNet CloseWindow (uint wid)

returns:

=00000000

Example:

+PixelNet CloseWindow 30002

=00000000

Table 39: CloseWindow

Syntax	+PixelNet CloseWindow 30002
Response	=00000000
Response Explanation	OK response

3.5.5 CloseMulticastWindow

CloseMulticastWindow specifies a window instance (by window ID and instance ID) to be closed. This command operates on the currently active display wall. Refer to [“ActivateWall” on page 17](#) for more information.

Command:

+PixelNet CloseMulticastWindow (uint wid, uint instanceid)

returns:

=00000000

Example:

+PixelNet CloseMulticastWindow 30002 3

=00000000

3—PDC Commands

Table 40: CloseMulticastWindow

Syntax	+PixelNet CloseMulticastWindow 30002 3
Response	=00000000
Response Explanation	OK response

3.5.6 CloseAllWindows

CloseAllWindows specifies that all windows be closed on the specified display wall (by WallName).

Command:

+PixelNet CloseAllWindows (string WallName)

returns:

=00000000

Example:

+PixelNet CloseAllWindows "Wall1"

=00000000

Table 41: CloseAllWindows

Syntax	+PixelNet CloseAllWindows "Wall1"
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.7 NewWindow

NewWindow creates a new window (with a Source name, state, position, size, and ZOrder) from a specific Source.

Command:

+PixelNet NewWindow (string srcName, PixelNet.TWindowState WS, ref int wid)

returns:

=00000000 "Window ID"

Example:

+PixelNet NewWindow "MySource3" {0 0 0 500 300 1000 700 0}

=00000000 "14"

Note The **NewWindow** command returns the window ID that was just created. Use this window ID to refer to the window hereafter. However, this window ID is random and will not be the same across other instances of PDC. Hence, refer to **NewWindowWithID** to solve this problem in case the layout is saved and reloaded.

Data Structure

Table 42: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid window ID number (not used)

nState not currently used

Set not currently used

x, y X position, Y position – window position in pixels

3—PDC Commands

w, h width, height – window size in pixels

ZOrder ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

Table 43: NewWindow

Syntax	+PixelNet NewWindow "MySource3" {0 0 0 500 300 1000 700 0}
Elements Explanation	<p>"MySource3" {0 0 0 500 300 1000 700 0}</p> <p>"MySource3" = Source name</p> <hr/> <p>Note If the source name is in a folder in the sources panel, use the full path not just the source name.</p> <hr/> <p>0 = this can be any number since the system-generated window ID will overwrite this number later</p> <p>0 = nState (not currently used)</p> <p>0 = Set (not currently used)</p> <p>500 = X-axis position in pixels</p> <p>300 = Y-axis position in pixels</p> <p>1000 = window width in pixels</p> <p>700 = window height in pixels</p> <p>0 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>
Response	=00000000 14 (Window ID just created)
Response Explanation	OK response

Window Commands

3.5.8 NewWindowOnWall

NewWindowOnWall creates a new window with a user-specified Window ID (with state, position, size, and ZOrder).

Command:

+PixelNet NewWindowOnWall (int wallID, uint WindowID, uint x, uint y, ref int instanceID)

returns:

=00000000

Example:

+PixelNet NewWindowOnWall 3 "JaneAAB" {0 0 0 256 192 512 384 2}

=00000000

3—PDC Commands

Table 44: NewWindowOnWall

Syntax	+PixelNet NewWindowOnWall 3 "JaneAAB" {0 0 0 256 192 512 384 2}
Elements Explanation	<p>3 = Wall ID</p> <p>"JaneAAB" = user-specified window ID</p> <p>{0 0 0 256 192 512 384 2}</p> <p>0 = Window Instance ID</p> <p>0 = nState</p> <p>0 = Set (not currently used)</p> <p>256 = X-axis position in pixels</p> <p>192 = Y-axis position in pixels</p> <p>512 = window width in pixels</p> <p>384 = window height in pixels</p> <p>2 = ZOrder of the window order, i.e. the display on the surface of the window stack order; 1 = behind window; 0 = on top; 2 = behind window 1, etc.</p>
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.9 NewWindowWithID

NewWindowWithID creates a new window with a user-specified window ID (with state, position, size, and ZOrder) from the specified Source.

Command:

+PixelNet NewWindowWithID (string srcName, PixelNet.TWindowState WS)

Example:

+PixelNet NewWindowWithID "MySource4" { 30001 0 0 960 120 720 450 0}

returns:

=00000000

Note The **NewWindowWithID** is similar to the **NewWindow** command except that the user specifies the window ID for the **NewWindowWithID** command.

Data Structure

Table 45: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid window ID number (must be in the range of 30,001 to 400,000)

nState not currently used

Set not currently used

3—PDC Commands

x, y X position, Y position – window position in pixels

w, h width, height – window size in pixels

ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

Table 46: NewWindowWithID

Syntax	+PixelNet NewWindowWithID "MySource4" { 30001 0 0 960 120 720 450 0 }
Elements Explanation	<p>"MySource4" = name of Source (string srcName)</p> <p>{ 30001 0 0 960 120 720 450 0 }</p> <p>30001 = The window ID must be in the range of 30,001 to 400,000.</p> <p>0 = nState</p> <p>0 = Set (not currently used)</p> <p>960 = X-axis position in pixels</p> <p>120 = Y-axis position in pixels</p> <p>720 = window width in pixels</p> <p>450 = window height in pixels</p> <p>0 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.10 NewMulticastWindow

NewMulticastWindow returns a Multicast Window Instance ID (The window ID is the same as the multicastableWindowID but the instance is unique).

Command:

+PixelNet NewMulticastWindow (uint multicastableWindowID, string targetWallName, TWindowState newMulticastWindowState, ref int multicastWindowInstanceID)

returns:

=00000000 "multicastWindowInstanceID"

Example:

+PixelNet NewMulticastWindow 9 600 800

=00000000 2

Data Structure

Table 47: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer integer	multicastableWindowID x, y

3—PDC Commands

Data Structure Elements—TWindowState

WindowID/multicastWindowInstanceID

ID for the multicastable window **OR**
multicastWindowInstanceID

x, y X position, Y position – window position in pixels

Table 48: NewMulticastWindow

Syntax	+PixelNet NewMulticastWindow 9 600 800
Elements Explanation	9 = multicastableWindowID 600 = X-axis position in pixels 800 = Y-axis position in pixels
Response	=00000000 2
Response Explanation	OK response 2 = the second instance of the multicast window (as every instance of a window will have the same WindowID, enumerating the instances will help distinguish them from one another)

Window Commands

3.5.11 NewMulticastWindowOnWall

NewMulticastWindowOnWall creates a new multicast window with a user-specified Window ID (with state, position, size, and ZOrder).

Command:

+PixelNet NewMulticastWindowOnWall (int wallID, uint multicastableWindowID, uint x, uint y, ref int instanceID)

returns:

=00000000 "multicastWindowInstanceID"

Example:

+PixelNet NewMulticastWindowOnWall 3 9 { 600 800 }

=00000000 2

Table 49: NewMulticastWindowOnWall

Syntax	+PixelNet NewMulticastWindowOnWall 3 9 { 600 800 }
Elements Explanation	<p>3 = Wall ID</p> <p>9 = multicastableWindowID</p> <p>600 = X-axis position in pixels</p> <p>800 = Y-axis position in pixels</p>
Response	=00000000 2
Response Explanation	<p>OK response</p> <p>2 = the second instance of the multicast window (as every instance of a window will have the same WindowID, enumerating the instances will help distinguish them from one another)</p>

3—PDC Commands

3.5.12 NewWindowWithIDOnWall

NewWindowWithIDOnWall creates a new window with a user-specified window ID (with state, position, size, and ZOrder) from the specified Source.

Command:

+PixelNet NewWindowWithIDOnWall (int wallID, string srcName, PixelNet.TWindowState WS)

returns:

=00000000

Example:

**+PixelNet NewWindowWithIDOnWall 3 "JaneAAB" "MySource4"
{ 30001 0 0 256 192 512 384 1 }**

=00000000

Data Structure

Table 50: Data Structure - TWindowState

Structure Name	Type	Elements
TWindowState	integer unsigned unsigned integer integer	wid nState Set x, y, w, h ZOrder

Data Structure Elements—TWindowState

wid window ID number (must be in the range of 30,001 to 400,000)

nState not currently used

Set not currently used

x, y X position, Y position – window position in pixels

w, h width, height – window size in pixels

Window Commands

ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.

Table 51: NewWindowWithIDOnWall

Syntax	+PixelNet NewWindowWithIDOnWall 3 "JaneAAB" "MySource4" { 30001 0 0 960 120 720 450 0 }
Elements Explanation	<p>3 = Wall ID</p> <p>"JaneAAB" = user-specified window ID</p> <p>"MySource4" = name of Source (string srcName)</p> <p>{ 30001 0 0 960 120 720 450 0 }</p> <p>30001 = The window ID must be in the range of 30,001 to 400,000.</p> <p>0 = nState</p> <p>0 = Set (not currently used)</p> <p>256 = X-axis position in pixels</p> <p>192 = Y-axis position in pixels</p> <p>512 = window width in pixels</p> <p>384 = window height in pixels</p> <p>1 = ZOrder (0 = on top) of the window order, i.e. the display on the surface of the window stack order; 1 = behind window 0; 2 = behind window 1, etc.</p>
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.5.13 QueryWindowMulticastState

Returns the window state which contains the valid position and size of the new multicast window.

Returns one of the following responses:

- 1=CannotMulticastWindow
- 0=CanMulticastWindow
- 0=CanMulticastWindow with adjusted position*

*The latter response means that multicasting window is possible as long as the adjusted coordinates returned in the response are used.

When the query coordinates for X and Y axis exceed the allocated region on the wall, the response to the query will contain coordinates that are adjusted just enough to fit the upper-right corner of the window within the wall.

Command:

+PixelNet QueryWindowMulticastState (uint WindowID, string TargetWallName, TWindowState newMulticastWindowState, ref TWindowState validWindowState)

returns:

=00000000 0

Example 1:

+PixelNet QueryWindowMulticastState 9 1100 100

=00000000 1100 100

Example 2:

+PixelNet QueryWindowMulticastState 9 2000 100

=00000000 1536 100

Window Commands

Note Notice that Example-1 is a success response as the queried coordinates (1100 x 100) have been accepted and included in the response.
 Although the queried coordinates of 2000 x 100 will not be possible on the current wall, those coordinates have been adjusted to fit the current wall (1536 x 100) and included in the query response, hence Example-2 is also a success response.

Table 52: QueryWindowMulticastState

Syntax	+PixelNet QueryWindowMulticastState 9 2000 100
Elements Explanation	9 = system or user generated window ID 2000 = X-axis 100 = Y-axis
Response	=00000000 1536 100
Response Explanation	OK response; multicasting window is possible as long as the adjusted coordinates returned in the response (1536 x 100) are used (i.e. the query coordinates 2000 x 100 will extend the window out of the current wall and therefore cannot be accepted)

3—PDC Commands

3.5.14 GetMulticastWindowSize

GetMulticastWindowSize returns the window size of the multicast window specified by Window ID.

Command:

+PixelNet GetMulticastWindowSize (uint wid)

returns:

ref int x, ref int y

Example:

+PixelNet GetMulticastWindowSize 30002

returns:

=00000000 30002 720 450

Table 53: GetMulticastWindowSize

Syntax	+PixelNet GetMulticastWindowSize 30002
Response	=00000000 30002 720 450
Response Explanation	<p>30002 720 450</p> <p>30002 = system or user generated window ID</p> <p>720 = window width in pixels</p> <p>450 = window height in pixels</p>

Window Commands

3.5.15 SetMulticastWindowSize

SetMulticastWindowSize sets the window size specified by Window ID.

Command:

+PixelNet SetMulticastWindowSize (uint wid, int x, int y)

returns:

=00000000

Example:

+PixelNet SetMulticastWindowSize 30002 820 450

returns:

=00000000

Table 54: SetMulticastWindowSize

Syntax	+PixelNet SetMulticastWindowSize 30002 820 450
Elements Explanation	30002 820 450 30002 = system or user generated window ID 820 = window width in pixels 450 = window height in pixels
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.5.16 GetMulticastWindowLocation

GetMulticastWindowLocation returns the x and y values for the specified Window ID and Instance ID.

Command:

+PixelNet GetMulticastWindowLocation (uint wid, uint instanceID)

returns:

ref int x, ref int y

Example:

+PixelNet GetMulticastWindowLocation 30004 1

=00000000 30004 1 1000 0

Table 55: GetMulticastWindowLocation

Syntax	+PixelNet GetMulticastWindowLocation 30004 1
Response	=00000000 30004 1 1000 0
Response Explanation	<p>30004 = system or user generated window ID</p> <p>1 = window instance ID</p> <p>1000 = X-axis position in pixels</p> <p>0 = Y-axis position in pixels</p>

Window Commands

3.5.17 SetMulticastWindowLocation

SetMulticastWindowLocation sets the x and y values for the specified Window ID and Instance ID.

Command:

+PixelNet SetMulticastWindowLocation (uint wid, uint instanceID, int x, int y)

returns:

=00000000

Example:

+PixelNet SetMulticastWindowLocation 30004 1 1200 10

=00000000

Table 56: SetMulticastWindowLocation

Syntax	+PixelNet SetMulticastWindowLocation 30004 1 1200 10
Elements Explanation	<p>30004 = system or user generated window ID</p> <p>1 = window instance ID</p> <p>1200 = X-axis position in pixels</p> <p>10 = Y-axis position in pixels</p>
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.5.18 FreezeInput

FreezeInput specifies an input channel (by window ID) to freeze on the active display wall.

Command:

+PixelNet FreezeInput (uint wid, int freeze)

returns:

=00000000

Example:

+PixelNet FreezeInput 30009 0

=00000000

Table 57: FreezeInput

Syntax	+PixelNet FreezeInput 30009 0
Elements Explanation	30009 = system or user generated window ID 1 = freeze, 0 = not freeze
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.19 setFrameInfo

SetFrameInfo sets frame parameters for a specific window (by window ID).

Command:

+PixelNet setFrameInfo (uint wid, PixelNet.TFrameInfo frameInfo)

returns:

=00000000

Example:

+PixelNet setFrameInfo 30010 {7 1 20 1081456}

=00000000

Data Structure

Table 58: Data Structure - TFrameInfo

Structure Name	Type	Elements
TFrameInfo	integer	Set
	integer	ShowFrame
	integer	FrameWidth
	integer	FrameColor

Data Structure Elements—TFrameInfo

Set * bit field - low order 3 bits (0000 0111)

ShowFrame indicates show/hide frame (show= 1, hide = 0; default = 1)

FrameWidth frame width - 0 to 100 (default = 4)

FrameColor frame color – (Alpha, R, G, B) to the size of the wall
 Alpha = 0 to 255 (denotes the transparency/opacity level of the color; 0=0% Opacity/100% Transparency, whereas 255=100% Opacity/0% Transparency)
 Red = 0 to 255
 Blue = 0 to 255

3—PDC Commands

Green = 0 to 255
 White is all color on – (256) x (256) x (256) = 0xFFFFFFFF or 16,777,215
 Black is all color off – 000 = 0x000000.
 Example – green only – 00 FF 00 = 65,280 = 0x00FF00

* Set argument is used to determine which fields are to be changed (are valid).

Table 59: TFrameInfo Bit Positions

Set Bit Positions							
128	64	32	16	8	4	2	1
0	0	0	0	0	FC	FW	SF

Table 60: SetFrameInfo

Syntax	+PixelNet SetFrameInfo 30010 { 7 1 20 1081456 }
Elements Explanation	<p>30010 { 7 1 20 1081456 }</p> <p>30010 = user or system-generated window ID</p> <p>7 = Set (it means all three attributes--FC, FW, and SF or 4+2+1-- will be changed)</p> <p>1 = ShowFrame (0 hides frame)</p> <p>20 = FrameWidth</p> <p>1081456 = FrameColor (ARGB:0x00108070; A=0x00000000, R=0x00100000, G=0x00008000, B=0x00000070)</p>
Response	=00000000
Response Explanation	OK response

Window Commands

3.5.20 SetCropInfo

SetCropInfo sets cropping parameters for a specific window (by window ID).

Cropping works by relation to the normal image size. Left and Right specify the pixels removed from the left and right of the image. Top and Bottom specify the size of the cropped image from the top and bottom.

Command:

+PixelNet SetCropInfo (uint wid, PixelNet.TCropInfo cropInfo)

returns:

=00000000

Example:

+PixelNet SetCropInfo 30010 {0 10 20 30 40}

=00000000

Data Structure

Table 61: Data Structure - TCropInfo

Structure Name	Type	Elements
TCropInfo	integer integer integer integer integer	Set Left Right Top Bottom

Data Structure Elements—TCropInfo

Set determines which parameters are to be changed (are valid)

Left pixels removed from the left

Right pixels removed from the right

Top pixels removed from the top

Bottom pixels removed from the bottom

3—PDC Commands

Table 62: SetCropInfo

Syntax	+PixelNet SetCropInfo 30010 {0 10 20 30 40}
Elements Explanation	30010 {0 10 20 30 40} 30010 = system-generated window ID 0 = Set 10 = Left (pixels removed from left) 20 = Right (pixels removed from right) 30 = Top (pixels removed from top) 40 = Bottom (pixels removed from bottom)
Response	=00000000
Response Explanation	OK response

Table 63: TCropInfo Bit Positions

Set Bit Positions			
8	4	2	1
Bottom	Top	Left	Right

Window Commands

3.5.21 SetTitleInfo

SetTitleInfo sets Title parameters for the specific window (by window ID).

Command:

+PixelNet SetTitleInfo (uint wid, PixelNet.TTitleInfo titleInfo)

returns:

=00000000

Example:

+PixelNet SetTitleInfo 30012 {511 1 10 "security" 65280 "Arial" 18 1 1}

=00000000

Data Structure

Table 64: Data Structure - TTitleInfo

Structure Name	Type	Elements
TTitleInfo	integer	Set
	integer	ShowTitle
	integer	MinHeight
	string	TitleText
	integer	ForeColor
	string	FontName
	integer	FontSize
	integer	HorizontalJust
	integer	TitlePosition

Data Structure Elements—TTitleInfo

Set determines which parameters are to be changed (are valid)

ShowTitle sets/hides title (set = 1, hide = 0; default = 1)

MinHeight specifies the minimum designated height for the title

TitleText text content of the title

3—PDC Commands

ForeColor specifies the foreground color of the title
 forecolor – (R, G, B) to the size of the wall
 Red = 1 to 255
 Blue = 1 to 255
 Green = 1 to 255
 White is all color on – (256) x (256) x (256) =
 0xFFFFFFFF or 16,777,215
 Black is all color off – 000 = 0x000000.
 Example – green only – 00 FF 00 = 65,280 =
 0x00FF00

FontName specifies the font used for the title

FontSize specifies the font size for the title

HorizontalJust justifies the horizontal position of the title

TitlePosition justifies the vertical position of the title

Table 65: Elements Definition

Abbreviation	Elements
ST	ShowTitle
MH	MinHeight
TT	TitleText
FC	ForeColor
FN	FontName
FS	FontSize
HJ	HorizontalJust
TP	TitlePosition

Table 66: TTitleInfo Bit Positions

Set Bit Positions								
256	128	64	32	16	8	4	2	1
TP	HJ	FS	FN	0	FC	TT	MH	ST

Window Commands

Table 67: SetTitleInfo

Syntax	+PixelNet SetTitleInfo 30012 { 511 1 10 "security" 65280 "Arial" 18 1 1 }
Elements Explanation	<p>30012 { 511 1 10 "security" 65280 "Arial" 18 1 1 }</p> <p>30012 = user or system-generated window ID</p> <p>511 = Set</p> <p>1 = ShowTitle (0 hides frame)</p> <p>10 = MinHeight (Minimum Height)</p> <p>"security" = TitleText</p> <p>65280 = ForeColor (Text color) must be expressed in decimal</p> <p>"Arial" = FontName</p> <p>18 = FontSize</p> <p>1 = HorizontalJust (Horizontal Justification; 0 = Left, 1 = Center, 2 = Right)</p> <p>1 = TitlePosition (1 = Top, 2 = Middle, 3 = Bottom)</p>
Response	=00000000
Response Explanation	OK response

3—PDC Commands

3.5.22 SetTitleOverlay

SetTitleOverlay sets Title overlay parameters for the specified window (by window ID).

Command:

+PixelNet SetTitleOverlay (uint wid, PixelNet.TTitleOverlayInfo overlayInfo)

returns:

=00000000

Example:

+PixelNet SetTitleOverlay 30012 { 127 1 1 7A1200 1 45 30 0 }

=00000000

Data Structure

Table 68: Data Structure - TTitleOverlayInfo

Structure Name	Type	Elements
TTitleOverlayInfo	integer	Set
	integer	ShowOverlay
	integer	ShowBackColor
	integer	BackColor
	integer	Transparency
	integer	HOffset
	integer	VOffset
	integer	TextEffect

Data Structure Elements—TTitleOverlayInfo

Set determines which parameters are to be changed (are valid)

ShowOverlay indicates show/hide title overlay (show= 1, hide = 0; default = 1)

ShowBackColor indicates show/hide background color (show= 1, hide = 0; default = 1)

BackColor indicates the background color

Window Commands

BackColor – (R, G, B) to the size of the wall
 Red = 1 to 255
 Blue = 1 to 255
 Green = 1 to 255
 White is all color on – (256) x (256) x (256) =
 0xFFFFFFFF or 16,777,215
 Black is all color off – 000 = 0x000000.
 Example – green only – 00 FF 00 = 65,280 =
 0x00FF00

Transparency transparency of the title; transparency active = 1
 (when transparency is active it is between 0 and
 100%), no transparency = 0; default = 0

HOffset horizontal offset (pixels from left)

VOffset vertical offset (pixels from top)

TextEffect must be 0

Table 69: Elements Definition

Abbreviation	Elements
SO	ShowOverlay
SB	ShowBackColor
BC	BackColor
TR	Transparency
HO	HOffset
VO	VOffset
TE	TextEffect

Table 70: TTitleOverlay Bit Positions

Set Bit Positions							
128	64	32	16	8	4	2	1
0	TE	VO	HO	TR	BC	SB	SO

3—PDC Commands

Table 71: SetTitleOverlay

Syntax	+PixelNet SetTitleOverlay 30012 { 127 1 1 7A1200 1 45 30 0 }
Elements Explanation	<p>30012 { 127 1 1 7A1200 1 45 30 0 }</p> <p>30012 = user or system-generated window ID</p> <p>127 = Set</p> <p>1 = ShowOverlay (0 hides overlay)</p> <p>1 = ShowBackColor (0 hides background color)</p> <p>7A1200 = BackColor (background color) must be expressed in decimal</p> <p>1 = Transparency (0 = no transparency); Transparency is set between 0% and 100%.</p> <p>45 = HOffset (Horizontal Offset—pixels from left side)</p> <p>30 = VOffset (Vertical Offset —pixels from top)</p> <p>0 = TextEffect (reserved for future use; not currently supported)</p>
Response	=00000000
Response Explanation	OK response

Notification Commands

3.6 Notification Commands

To receive event notification from PDC, establish a connection to PDC as explained in [“Network Connection” on page 1](#).

With an established connection, issue the **RegisterNotification** command with appropriate flags to begin receiving notification of events in which you are interested. Use the **UnregisterNotification** command to stop receiving notifications for all events.

3.6.1 RegisterNotification

Flags are used to identify events in the **RegisterNotification** command. The flags each have only a single bit set and when used in **RegisterNotification** they can be combined together to identify multiple events. A value of 0 will enable all notifications.

Command:

+PixelNet RegisterNotification (int flags)

returns:

=00000000

Example:

+PixelNet RegisterNotification 0

=00000000

Flags

Table 72: Flags—RegisterNotification

Flag	Event	Parameter List
0	All	Only used with <i>RegisterNotification</i>
1	Load Layout	String WallName String LayoutName

3—PDC Commands

Table 73: RegisterNotification

Syntax	+PixelNet RegisterNotification 0
Elements Explanation	0 = register notification for <i>All</i> events
Response	=00000000
Response Explanation	OK response

3.6.2 UnregisterNotification

To cancel the event notification, either call the UnregisterNotification command without any parameter, or simply close the connection.

Command:

+PixelNet UnregisterNotification

returns:

=00000000

Example:

+PixelNet UnregisterNotification

=00000000

Table 74: UnregisterNotification

Syntax	+PixelNet UnregisterNotification
Elements Explanation	No parameters are needed to cancel the event notification
Response	=00000000
Response Explanation	OK response

Notification Commands

3.6.3 EventNotification

Different events generate different notifications, but each notification notifies only one event.

Format

Event Notifications use the following format:

:PDCNotify identifies this as an event notification

Flag identifies the notifying event (only one event)

Parameter List detailed information about the event. Different events may have different parameter lists.

Command:

:PDCNotify Flag [Parameter List]

returns:

=00000000

Example:

:PDCNotify 1 "Wall 1" "My Layout 123"

=00000000

Flags

Table 75: Flags—Event Notification

Flag	Event	Parameter List
1	Load Layout	String WallName String LayoutName

Table 76: EventNotification

Syntax	:PDCNotify 1 "Wall 1" "My Layout 123"
Elements Explanation	<p>1 = Load Layout event</p> <p>"Wall 1" = string identifying the name of the wall</p> <p>"My Layout 123" = string identifying the name of the layout which was just loaded.</p>

3—PDC Commands

Response	=00000000
Response Explanation	OK response

3.7 Audio Commands

Audio commands can be used to assign sources to audio nodes, pause and resume playback, and request or modify their settings.

3.7.1 PixelNet AssignSourceToAudioNode

AssignSourceToAudioNode assigns a Source to an audio node specified by name.

Command:

+PixelNet AssignSourceToAudioNode (string sourceName, string audioNodeName)

returns:

=00000000

Example:

+PixelNet AssignSourceToAudioNode "JaneAAB" "Real Audio"

=00000000

Table 77: AssignSourceToAudioNode

Syntax	+PixelNet AssignSourceToAudioNode "JaneAAB" "Real Audio"
Elements Explanation	"JaneAAB" = name of Source (string sourceName) "Real Audio" = name of audio node (string audioNodeName)
Response	=00000000
Response Explanation	OK response

Audio Commands

3.7.1.1 Unassigning Source

To unassign an assigned Source, use empty quotes in place of the Source name.

```
+PixelNet AssignSourceToAudioNode "" "Real Audio"
```

3.7.2 PixelNet PlayAudio

PlayAudio specifies the audio node (by name) to play.

Command:

```
+PixelNet PlayAudio (string audioNodeName, uint play)
```

returns:

```
=00000000
```

Example:

```
+PixelNet PlayAudio "Real Audio" 1
```

```
=00000000
```

Table 78: PlayAudio

Syntax	+PixelNet PlayAudio "Real Audio" 1
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName) 1 = PlayAudio command to play the audio node named "Real Audio"
Response	=00000000
Response Explanation	OK response.

PlayAudio command allows two possibilities:

- **+PixelNet PlayAudio "Real Audio" 1**
1 = Play the named audio node
- **+PixelNet PlayAudio "Real Audio" 0**
0 = Stop playing the named audio node

3—PDC Commands

3.7.3 PixelNet GetVolume

GetVolume returns the volume information for the audioNode specified.

Command:

+PixelNet GetVolume (string audioNodeName)

returns:

ref double Volume

Example:

+PixelNet GetVolume "Real Audio"

=00000000 1.0

Table 79: GetVolume

Syntax	+PixelNet GetVolume "Real Audio"
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName)
Response	=00000000 1.0
Response Explanation	OK response. 1.0=The audio is being played at the highest volume level.

Audio Commands

3.7.4 PixelNet SetVolume

SetVolume sets the audio volume as specified by audio node name and volume level.

Command:

+PixelNet SetVolume (string audioNodeName, double volume)

returns:

=00000000

Example:

+PixelNet SetVolume "Real Audio" 0.8

=00000000

Table 80: SetVolume

Syntax	+PixelNet SetVolume "Real Audio" 0.8
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName) 0.8 = exact volume level to be set (double volume)
Response	=00000000
Response Explanation	OK response

SetVolume command allows many possible volume levels designated in decimal units (in ones and tenths) within the range of 0.0 and 1.0.

- +PixelNet SetVolume "Real Audio" 0.0
0.0 = Mute the named audio node.
- +PixelNet SetVolume "Real Audio" 0.1
0.1 = Play the named audio node at the set volume level.
- +PixelNet SetVolume "Real Audio" 1.0
1.0 = Play the named audio node at the highest volume level.
- +PixelNet SetVolume "Real Audio" 1.1

3—PDC Commands

1.1 (or any value above 1.0 or below 0.0) = An error message states that values below 0.0 and 1.0 are not supported.

3.7.5 PixelNet GetBalance

GetBalance returns the audio balance information for the audioNode specified.

Command:

+PixelNet GetBalance (string audioNodeName)

returns:

ref double balance

Example:

+PixelNet GetBalance "Real Audio"

=00000000 0.8

Table 81: GetBalance

Syntax	+PixelNet GetBalance "Real Audio"
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName)
Response	=00000000 0.8
Response Explanation	OK response. The audio balance is set at 0.8 which is mostly to the right, i.e. 80% of the audio is heard on the right side and only 20% is coming through the left side.

Audio Commands

3.7.6 PixelNet SetBalance

SetBalance sets the audio volume as specified by audio node name and volume level.

Command:

+PixelNet SetBalance (string audioNodeName, double balance)

returns:

=00000000

Example:

+PixelNet SetBalance "Real Audio" 0.0

=00000000

Table 82: SetBalance

Syntax	+PixelNet SetBalance "Real Audio" 0.0
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName) 0.0 = exact balance level, equally divided between the left and right, is to be set (double balance)
Response	=00000000
Response Explanation	OK response

SetBalance command allows many possible balance levels designated in units within the range of -1.0 and 1.0.

- **+PixelNet SetBalance "Real Audio" 0.0**
0.0 = Set the named audio node with a perfect left and right side balance.
- **+PixelNet SetBalance "Real Audio" -1.0**
-1.0 = Set the audio node at the farthest left point; at this level, the audio node is left justified—no sound will come out of the right side.

3—PDC Commands

- +PixelNet SetBalance "Real Audio" +1.0
+1.0 = Set the audio node at the farthest right point; at this level, the audio node is right justified—no sound will come out of the left side.
- +PixelNet SetBalance "Real Audio" +1.1
1.1 (or any value above +1.0 or below -1.0 = An error message states that values below -1 and +1 are not supported.

3.7.7 PixelNet GetConnector

GetConnector returns the connector information for the audioNode specified.

Command:

+PixelNet GetConnector (string audioNodeName)

returns:

ref int connector

Example:

+PixelNet GetConnector "Real Audio"

=00000000 1

Table 83: GetConnector

Syntax	+PixelNet GetConnector "Real Audio"
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName)
Response	=00000000 1
Response Explanation	OK response. The audio node, "Real Audio" has been set to a Balanced stereo connector.

Audio Commands

3.7.8 PixelNet SetConnector

SetConnector sets the audio connection by audio node name and volume level.

Command:

+PixelNet SetConnector (string audioNodeName, int connector)

returns:

=00000000

Example:

+PixelNet SetConnector "Real Audio" 4

=00000000 { "JaneAAB" 1 0.8 0.0 4 }

Table 84: SetConnector

Syntax	+PixelNet SetConnector "Real Audio" 4
Elements Explanation	"Real Audio" = name of audio node (string audioNodeName) 4 = SPDIF
Response	=00000000
Response Explanation	OK response

SetConnector command allows five possible connector levels: 0 to 4.

- **+PixelNet SetConnector "Real Audio" 0**
0 = Set the named audio node to the default connector configuration.
- **+PixelNet SetConnector "Real Audio" 1**
1 = Set the named audio node to **Balanced Stereo** connector configuration.
- **+PixelNet SetConnector "Real Audio" 2**
2 = Set the named audio node to **Unbalanced Stereo** connector configuration.

3—PDC Commands

- +PixelNet SetConnector "Real Audio" 3
3 = Set the named audio node to **AES3id** connector configuration.
- +PixelNet SetConnector "Real Audio" 4
4 = Set the named audio node to **SPDIF** connector configuration.

3.7.9 PixelNet QueryAudioNodeStatus

QueryAudioNodeStatus requests all the status details related to a named audio node—such as assignedSource, playing status, volume, balance, and connection configuration.

Command:

+PixelNet QueryAudioNodeStatus (string audioNodeName)

returns:

ref string assignedSource, ref int playStatus, ref int volume, ref int balance, and ref int connector

Example:

+PixelNet QueryAudioNodeStatus "Real Audio"

=00000000 {"JaneAAB" 1 0.8 0.0 4 }

Table 85: QueryAudioNodeStatus

Syntax	+PixelNet QueryAudioNodeStatus "Real Audio"
Response	=00000000 {"JaneAAB" 1 0.8 0.0 4 }
Response Explanation	<p>{"JaneAAB" 1 0.8 0.0 4 }</p> <p>"JaneAAB" = name of Source (ref string assignedSource)</p> <p>1 = PlayAudio command (PlayAudio "audioNodeName" 1)</p> <p>0.8 = exact volume level (double volume)</p> <p>0.0 = exact balance level (double balance)</p> <p>4 = the audio node connector has been set to SPDIF</p>

Audio Commands

3.7.10 PixelNet QueryAllAudioNodes

QueryAllAudioNodes requests a list of all the audio nodes by their name based on the Wall (identified by Wall ID) to which they are assigned.

Command:

+PixelNet QueryAllAudioNodes (int WallID)

returns:

ref string[] audioNodes

Example:

+PixelNet QueryAllAudioNodes 1

=00000000 { 1 "Real Audio" }

Table 86: QueryAllAudioNodes

Syntax	+PixelNet QueryAllAudioNodes 1
Response	=00000000 { 1 "Real Audio" }
Response Explanation	{ 1 "Real Audio" } 1 = number of audio nodes assigned to the wall "Real Audio" = name of audio node



Appendix A—Error Codes

A. Error Codes

This appendix lists the error codes in the PDC Protocol.

Table 87: Error Codes

Error Codes	Description
00000000	Success
00000001	Failed
00000002	Function not supported
00000003	Window Bad Parameter—This code indicates that either the width or height of the window is too small or negative or that the window Z-Order is negative
00000004	Window not found
00000005	Input not found
00000006	Out of resources
00000007	Invalid source path
00000008	Source not found
00000009	Invalid window
0000000A	Window ID already exists
0000000B	Could not find window
0000000C	Window frame not shown

A—Error Codes

Table 87: Error Codes

0000000D	Window title not shown
0000000E	Not used currently
0000000F	Window is off screen
80040501	Not enough parameters supplied
80040502	Too many parameters supplied
80040503	Invalid RMC method ID
80040504	Invalid RMC object ID
80040505	Bad Parameter
80040507	Server error
8004050D	Execution terminated
80040516	Protocol Error



Index of Tables

Index of Tables

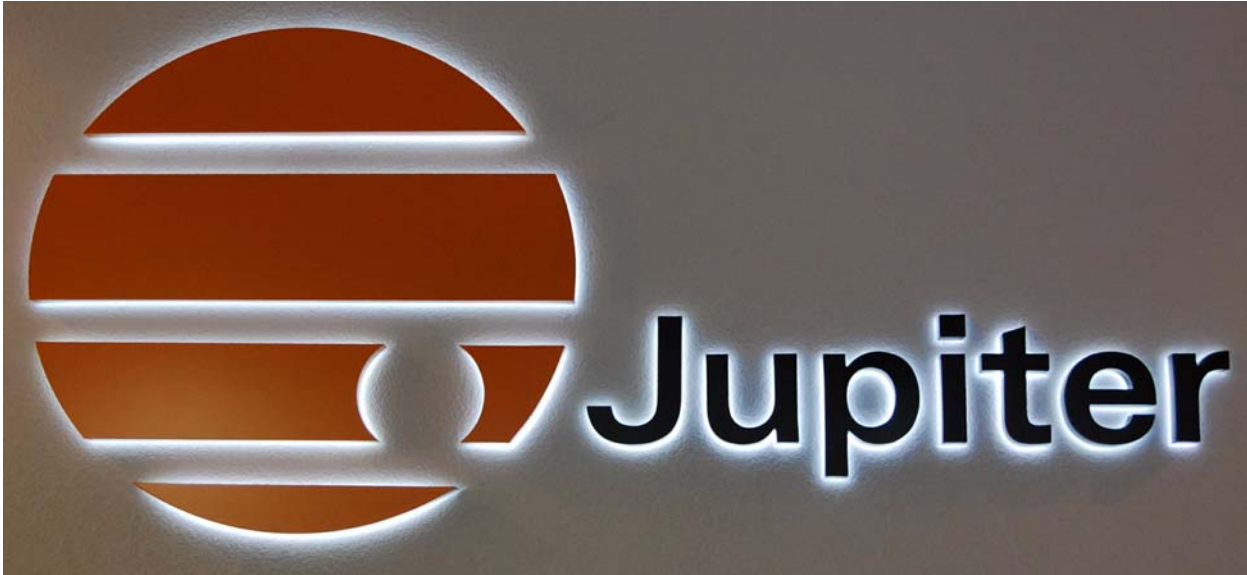
Table 1. QueryConfigurations	10
Table 2. QueryCurrentConfiguration	11
Table 3. LoadConfiguration	12
Table 4. QuerySystemStatus	13
Table 5. Data Structure—TWallInfo	14
Table 6. QueryAllWalls	15
Table 7. Data Structure—TWallInfo	16
Table 8. QueryWallInfo	17
Table 9. ActivateWall	18
Table 10. QueryLayouts	19
Table 11. LoadLayout	20
Table 12. LoadLayoutOnWall	21
Table 13. QueryCurrentLayout	22
Table 14. Data Structure—TLayoutInfo	22
Table 15. QueryActiveLayouts	23
Table 16. SaveLayout	23
Table 17. SaveLayoutOnWall	24
Table 18. Data Structure—SourceNodeAttribute	25
Table 19. CreateDynamicSource	26
Table 20. SaveLayoutOnWall	27
Table 21. Data Structure—SourceNodeAttribute	28
Table 22. QueryDynamicSources "All"	29
Table 23. Data Structure—SourceNodeAttribute	30

Tables

Table 24. QueryDynamicSources "Recent"	30
Table 25. Data Structure—SourceNodeAttribute	31
Table 26. QueryDynamicSources "NotSavedInConfig"	31
Table 27. Data Structure—SourceNodeAttribute	32
Table 28. QuerySources	33
Table 29. Data Structure—SourceNodeAttribute	33
Table 30. GetSource	34
Table 31. GetSourceFreeCount	35
Table 32. SelectSource	36
Table 33. Data Structure - TWindowState	37
Table 34. QueryAllWindows	38
Table 35. Data Structure - TWindowState	39
Table 36. GetWindowState	40
Table 37. Data Structure - TWindowState	41
Table 38. SetWindowState	42
Table 39. CloseWindow	43
Table 40. CloseMulticastWindow	44
Table 41. CloseAllWindows	44
Table 42. Data Structure - TWindowState	45
Table 43. NewWindow	46
Table 44. NewWindowOnWall	48
Table 45. Data Structure - TWindowState	49
Table 46. NewWindowWithID	50
Table 47. Data Structure - TWindowState	51
Table 48. NewMulticastWindow	52
Table 49. NewMulticastWindowOnWall	53
Table 50. Data Structure - TWindowState	54
Table 51. NewWindowWithIDOnWall	55
Table 52. QueryWindowMulticastState	57
Table 53. GetMulticastWindowSize	58
Table 54. SetMulticastWindowSize	59
Table 55. GetMulticastWindowLocation	60
Table 56. SetMulticastWindowLocation	61
Table 57. FreezeInput	62
Table 58. Data Structure - TFrameInfo	63

Tables

Table 59. TFrameInfo Bit Positions	64
Table 60. SetFrameInfo	64
Table 61. Data Structure - TCropInfo	65
Table 62. SetCropInfo	66
Table 63. TCropInfo Bit Positions	66
Table 64. Data Structure - TTitleInfo	67
Table 65. Elements Definition	68
Table 66. TTitleInfo Bit Positions	68
Table 67. SetTitleInfo	69
Table 68. Data Structure - TTitleOverlayInfo	70
Table 69. Elements Definition	71
Table 70. TTitleOverlay Bit Positions	71
Table 71. SetTitleOverlay	72
Table 72. Flags—RegisterNotification	73
Table 73. RegisterNotification	74
Table 74. UnregisterNotification	74
Table 75. Flags—Event Notification	75
Table 76. EventNotification	75
Table 77. AssignSourceToAudioNode	76
Table 78. PlayAudio	77
Table 79. GetVolume	78
Table 80. SetVolume	79
Table 81. GetBalance	80
Table 82. SetBalance	81
Table 83. GetConnector	82
Table 84. SetConnector	83
Table 85. QueryAudioNodeStatus	84
Table 86. QueryAllAudioNodes	85
Table 87. Error Codes	87





Index

Index

A

ActivateWall	
Wall Commands	17
AES3id	
SetConnector	84
All (filter)	
QueryDynamicSources	28
Appendix A	
Error Codes	87
Arguments	
Bit Fields	7
Data Types	7
PDC Command	7
AssignSourceToAudioNode	
Audio Commands	76
Audio Commands	
AssignSourceToAudioNode	76
GetBalance	80
GetConnector	82
GetVolume	78
PDC	76
PlayAudio	77
QueryAllAudioNodes	85
QueryAudioNodeStatus	84
SetBalance	81

SetConnector	83
SetVolume	79
Unassign Source	77

B

Balanced Stereo	
SetConnector	83
Bit Fields	7
Bit Positions	
TCropInfo	66
TFrameInfo	64
TTitleInfo	68
TTitleOverlayInfo	71

C

CloseAllWindows	
Window Commands	44
CloseMulticastWindow	
Window Commands	43
CloseWindow	
Window Commands	43
Commands	
PDC	9
Communicating	
PDC	1
Configuration Commands	
LoadConfiguration	12
PDC Commands	10
QueryConfigurations	10
QueryCurrentConfiguration	11
QuerySystemStatus	13

Index

Copyright iii
 CreateDynamicSource
 Source Commands 25

D

Data Structure
 SourceNodeAttribute 25, 28, 30, 31, 32,
 33
 TCropInfo 65
 TFrameInfo 63
 TTitleInfo 67
 TTitleOverlayInfo 70, 73, 75
 TWallInfo 14, 16
 TWindowState 37, 39, 41, 45, 49, 51,
 54
 DeleteDynamicSource
 Source Commands 27

E

Error Codes
 Appendix A 87
 EventNotification
 Notification Commands 75

F

FreezeInput
 Window Commands 62

G

GetBalance
 Audio Commands 80
 GetConnector
 Audio Commands 82
 GetMulticastWindowLocation
 PDC Commands 60
 Window Commands 60
 GetMulticastWindowSize

PDC Commands 58
 Window Commands 58
 GetSourceFreeCount
 Source Commands 34
 GetSources
 Source Commands 33
 Getting Connected
 Network Communications 1, 2
 Windows Vista 2
 Windows XP 1
 GetVolume
 Audio Commands 78
 GetWindowState
 Window Commands 39

I

Index of Tables 89

L

Layout Commands
 LoadLayout 19
 LoadLayoutOnWall 20
 PDC Commands 19
 QueryActiveLayouts 22
 QueryCurrentLayout 21
 QueryLayouts 19
 SaveLayout 23
 SaveLayoutOnWall 24
 LoadConfiguration
 Configuration Commands 12
 LoadLayout
 Layout Commands 19
 LoadLayoutOnWall
 Layout Commands 20

N

Network Communication
 PDC 1
 Network Communications
 Getting Connected 1, 2

Index

NewMulticastWindow		GetWindowState	39
Window Commands	51	Layout Commands	19
NewMulticastWindowOnWall		LoadConfiguration	12
Window Commands	53	LoadLayout	19
NewWindow		LoadLayoutOnWall	20
Window Commands	45	NewMulticastWindow	51
NewWindowOnWall		NewMulticastWindowOnWall	53
Window Commands	47	NewWindow	45
NewWindowWithID		NewWindowOnWall	47
Window Commands	49	NewWindowWithID	49
NewWindowWithIDOnWall		NewWindowWithIDOnWall	54
Window Commands	54	Notification Commands	73
Notification Commands		QueryActiveLayouts	22
EventNotification	75	QueryAllWalls	14
PDC Commands	73	QueryAllWindows	37
RegisterNotification	73	QueryConfigurations	10
UnregisterNotification	74	QueryCurrentConfiguration	11
NotSavedInConfig (filter)		QueryCurrentLayout	21
QueryDynamicSources	28	QueryDynamicSources	28
		QueryLayouts	19
		QuerySources	32
		QuerySystemStatus	13
		QueryWallInfo	16
		QueryWindowMulticastState	56
		RegisterNotification	73
		SaveLayout	23
		SaveLayoutOnWall	24
		SelectSource	35
		SetCropInfo	65
		SetFrameInfo	63
		SetMulticastWindowLocation	61
		SetMulticastWindowSize	59
		SetTitleInfo	67
		SetTitleOverlay	70
		SetWindowState	41
		Source Commands	25
		UnregisterNotification	74
		Wall Commands	14
		Window Commands	37
		PDC Protocol	
		Introduction	1
		PlayAudio	
		Audio Commands	77
		Protocol Syntax	
		PDC	5

P

PDC

Audio Commands	76
Commands	9
Communicating	1
Network Communication	1
Protocol Syntax	5
PDC Command	
Arguments	7
Request Line	6
Response Line	6
PDC Commands	
ActivateWall	17
CloseAllWindows	44
CloseMulticastWindow	43
CloseWindow	43
Configuration Commands	10
CreateDynamicSource	25
DeleteDynamicSource	27
EventNotification	75
FreezeInput	62
GetMulticastWindowLocation	60
GetMulticastWindowSize	58
GetSourceFreeCount	34
GetSources	33

Index

Q

QueryActiveLayouts	
Layout Commands	22
QueryAllAudioNodes	
Audio Commands	85
QueryAllWalls	
Wall Commands	14
QueryAllWindows	
Window Commands	37
QueryAudioNodeStatus	
Audio Commands	84
QueryConfigurations	
Configuration Commands	10
QueryCurrentConfiguration	
Configuration Commands	11
QueryCurrentLayout	
Layout Commands	21
QueryDynamicSources	
All (filter)	28
NotSavedInConfig (filter)	28
Source Commands	28
QueryLayouts	
Layout Commands	19
QuerySources	
Source Commands	32
QuerySystemStatus	
Configuration Commands	13
PDC Commands	13
QueryWallInfo	
Wall Commands	16
QueryWindowMulticastState	
Window Commands	56

R

RegisterNotification	
Notification Commands	73
Request Line	
PDC Command	6
Response Line	
PDC Command	6

S

SaveLayout	
Layout Commands	23
SaveLayoutOnWall	
Layout Commands	24
SelectSource	
Source Commands	35
SetBalance	
Audio Commands	81
SetConnector	
AES3id	84
Audio Commands	83
Balanced Stereo	83
SPDIF	84
Unbalanced Stereo	83
SetCropInfo	
Window Commands	65
SetFrameInfo	
Window Commands	63
SetMulticastWindowLocation	
PDC Commands	61
Window Commands	61
SetMulticastWindowSize	
PDC Commands	59
Window Commands	59
SetTitleInfo	
Window Commands	67
SetTitleOverlay	
Window Commands	70
SetVolume	
Audio Commands	79
SetWindowState	
Window Commands	41
Software Warranty	vi
Source Commands	
CreateDynamicSource	25
DeleteDynamicSource	27
GetSourceFreeCount	34
GetSources	33
PDC Commands	25
QueryDynamicSources	28
QuerySources	32
SelectSource	35
SourceNodeAttribute	
Data Structure 25, 28, 30, 31, 32, 33	

Index

SPDIF
 SetConnector 84
Statement of Limited Warranty v

T

Table of Contents ix
TCropInfo
 Bit Positions 66
 Data Structure 65
TFrameInfo
 Bit Positions 64
 Data Structure 63
TTitleInfo
 Bit Positions 68
 Data Structure 67
TTitleOverlayInfo
 Bit Positions 71
 Data Structure 70, 73, 75
TWallInfo
 Data Structure 14, 16
TWindowState
 Data Structure 37, 39, 41, 45, 49, 51,
 54

CloseMulticastWindow 43
CloseWindow 43
FreezeInput 62
GetMulticastWindowLocation 60
GetMulticastWindowSize 58
GetWindowState 39
NewMulticastWindow 51
NewMulticastWindowOnWall 53
NewWindow 45
NewWindowOnWall 47
NewWindowWithID 49
NewWindowWithIDOnWall 54
PDC Commands 37
QueryAllWindows 37
QueryWindowMulticastState 56
SetCropInfo 65
SetFrameInfo 63
SetMulticastWindowLocation 61
SetMulticastWindowSize 59
SetTitleInfo 67
SetTitleOverlay 70
SetWindowState 41
Windows Vista
 Getting Connected 2
Windows XP
 Getting Connected 1

U

Unassign Source
 Audio Commands 77
Unbalanced Stereo
 SetConnector 83
UnregisterNotification
 Notification Commands 74

W

Wall Commands
 ActivateWall 17
 PDC Commands 14
 QueryAllWalls 14
 QueryWallInfo 16
Window Commands
 CloseAllWindows 44

