



# Intel<sup>®</sup> UPnP<sup>™</sup>-Based Remote I/O Developer's Guide

Version 0.4 September 12, 2003

# DISCLAIMER

---

The information contained herein is not a license, either expressly or impliedly, to any intellectual property owned or controlled by the developer of this specification. The information contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this information is provided AS IS AND WITH ALL FAULTS. INTEL AND ALL OTHER DEVELOPERS OF THIS INFORMATION HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY (IF ANY) IMPLIED WARRANTIES, DUTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OR COMPLETENESS OF RESPONSES, OF RESULTS, OF WORKMANLIKE EFFORT, OF LACK OF VIRUSES, AND OF LACK OF NEGLIGENCE, ALL WITH REGARD TO THE INFORMATION AND THEIR CONTRIBUTION THERETO. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE INFORMATION CONTAINED HEREIN.

THE USER ASSUMES THE FULL RISK OF USING THE INFORMATION CONTAINED HEREIN. IN NO EVENT WILL INTEL OR ANY OTHER DEVELOPER OF THIS INFORMATION BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, PUNITIVE OR SPECIAL DAMAGES, WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF ANY USE OF THIS SPECIFICATION INFORMATION, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Any user of this specification information is not authorized to publicly state or claim that its products or applications conform with the Intel 2003 Networked Media Product Requirements unless and until a conformance program is developed and implemented. By requesting or using the specification information herein, user agrees to this limitation and condition.

This document is subject to change without notice.

Intel is a registered trademark of Intel Corporation.

Copyright 2003 © Intel Corporation. All rights reserved.

# CONTENTS

---

<b>Intel UPnP-Based Remote I/O .....</b>	<b>1</b>
1.1 Introduction .....	1
1.2 Device Overview .....	2
1.3 Remote I/O Service.....	2
1.3.1 State Variables .....	2
1.3.2 Actions.....	2
1.4 Channel Manager Service.....	4
1.4.1 State Variables .....	5
1.4.2 Actions.....	5
1.5 Remote Input Service.....	6
1.5.1 State Variables .....	6
1.5.2 Actions.....	7
1.6 Remote IO – Theory of Operation.....	8
1.6.1.1 Discovery .....	8
1.6.1.2 Description and Event Subscription .....	8
1.6.1.3 Determining Application Compatibility .....	8
1.6.1.4 Registering and Administering Channels on the Remote Client.....	8
<b>2 The Extended PC Remoting Technology Protocol v2 (XRT2).....</b>	<b>10</b>
2.1 Introduction .....	10
2.2 Protocol Overview .....	10
2.2.1 Command Encoding .....	10
2.2.2 Command Types .....	10
2.2.3 Jumbo Commands .....	11
2.3 Command Descriptions.....	11
2.3.1 Setup Commands.....	11
2.3.2 Basic Display Remoting Commands .....	12
2.3.3 Input Remoting Commands.....	13
2.3.4 Scan Codes for KEY_PRESS Server Command .....	14
2.3.5 Additional Commands .....	15
2.3.6 Vendor Extensions to XRT2 .....	15
<b>Appendix A: Device and Service SCPD .....</b>	<b>16</b>
A1: Remote Device SCPD .....	16
A2: Remote I/O Service SCPD .....	17
A3: Channel Manager Service SCPD .....	19
A4: Remote Input Service SCPD .....	20



# Intel UPnP-Based Remote I/O

## 1.1 Introduction

UPnP Remote I/O is not yet a ratified UPnP standard. However, there exists a clear need to enable extremely low cost devices for the digital home. Towards this goal, Intel has provided requirements for implementing technology that achieves the intent of UPnP Remote I/O before the standard is ratified. This technology is referred to as Intel UPnP-based Remote I/O, or in shortened form as Intel Remote I/O.

It is anticipated that vendors who implement Intel Remote I/O technology according to this document should be in a position to make minor modifications in their products necessary to achieve compliance with the UPnP Remote I/O 1.0 standard when it is ratified.<sup>1</sup>

Both the Intel Remote I/O and the developing UPnP Remote I/O standard provide the high-level commands needed to discover devices and setup sessions. Remote I/O implementations must also choose an on-the-wire protocol to transport information between the remote client device and the associated application. XRT2 is an Intel binary transport and command protocol designed for transporting display and input update information between remote clients and remoted applications (see Figure 1). XRT2 is used in conjunction with Intel Remote I/O-enabled devices and applications. Future guidelines for remote-enabled applications and devices will be based on the UPnP Remote I/O standard. These future guidelines will include support for vendor-specific remoting protocols, enumerated by the standard framework.

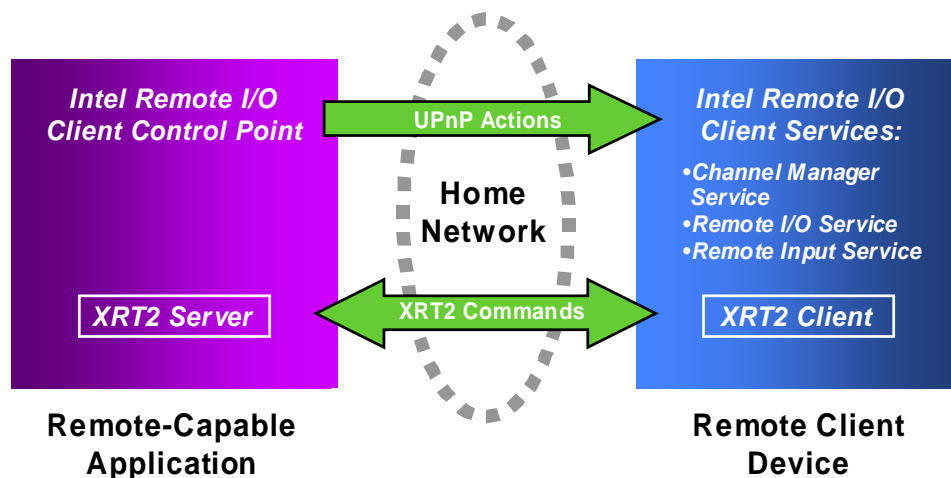


Figure 1. Remote I/O Application and Client Device

<sup>1</sup> The UPnP Forum is a democratic body, so there is always a possibility that a given standard will not be ratified. For this reason, Intel makes no assurances or guarantees that the UPnP Remote I/O specification will be ratified by the Forum.

## 1.2 Device Overview

All Intel Remote I/O client devices must expose the following device type in the XML device description:

**urn:schemas-upnp-org:device:RemoteIO:1**

Intel Remote I/O clients are allowed to embed a device of the above type in a root device of a different type. In any case, the root device XML description must include the following service types:

**urn:schemas-upnp-org:service:RemoteIO:1**

**urn:schemas-upnp-org:service:ChannelManager:1**

The Intel Remote I/O client device may also optionally include the RemoteInput service:

**urn:schemas-upnp-org:service:RemoteInput:1**

## 1.3 Remote I/O Service

The Intel Remote I/O service is identified in the service description XML by the ServiceURN “**urn:schemas-upnp-org:service:RemoteIO:1**”. The Intel **Remote I/O Service** allows Remote I/O Control Points associated with applications to control the user interface channel of a Intel Remote I/O client. This service includes an evented state variable that events a string corresponding to the currently connected channel URI. All of the actions in this service are required for compliant implementations. When referred to, the action names, state variables and arguments in the table below are preceded by ‘RIOS’ to indicate that they are part of the Intel **Remote I/O Service**.

### 1.3.1 State Variables

Variable Name	Required or Optional	UPnP dataType	Default Value	Evented YES/NO
<i>MaxCommandSize</i>	R	ui4	Not specified	NO
<i>ConnectionPort</i>	R	ui2	Not specified	NO
<i>DeviceInformation</i>	R	string	Not specified	NO
<i>PeerConnection</i>	R	string	Not specified	YES
<i>ApplicationIdentifier</i>	R	string	Not specified	NO
<i>DisplayEncoding</i>	R	int	Not specified	NO
<i>DisplaySize</i>	R	ui4	Not specified	NO

### 1.3.2 Actions

Action Name & Usage	Arguments & Description	IN / OUT	Related State Variable
<b>ForceDisconnect</b> Causes the Remote I/O client to disconnect from the currently connected user interface. If the Remote I/O client is not connected, this action has no effect. Repeated use of	No Arguments	N/A	N/A

Action Name & Usage	Arguments & Description	IN / OUT	Related State Variable
<b>RIOS:ForceDisconnect</b> is strongly discouraged.			
<p><b>GetDeviceInformation</b></p> <p>This action is used by Remote I/O control points to acquire information about the device. Information obtained with this action should complement information about the device already available in the UPnP device description (manufacturer, manufacturer URI, etc.)</p>	<p><b>Application</b> Vendor defined string.</p> <p><b>MaxCommandSize</b> The maximum size of an out-of-band command. In the case of XRT2, the JUMBO command must only be used if this value is greater than 64k.</p> <p><b>DisplayEncoding</b> An integer corresponding to the type of display encoding formats supported by the device. In the case of XRT2, JPEG is currently the only supported format (see the &lt;Remote I/O XRT2 binary image type support&gt; requirement for more information). The following value of <i>DisplayEncoding</i> is reserved: 1 = JPEG</p> <p><b>DisplayWidth</b> The total width, in pixels, of the Remote I/O client display.</p> <p><b>DisplayHeight</b> The total height, in pixels, of the Remote I/O client display.</p> <p><b>DeviceInformation</b> Additional vendor defined information about the Remote I/O client.</p>	<p>OUT</p> <p>OUT</p> <p>OUT</p> <p>OUT</p> <p>OUT</p> <p>OUT</p>	<p><i>ApplicationIdentifier</i></p> <p><i>MaxCommandSize</i></p> <p><i>DisplayEncoding</i></p> <p><i>DisplaySize</i></p> <p><i>DisplaySize</i></p> <p><i>DeviceInformation</i></p>
<p><b>GetPeerConnection</b></p> <p>This action is used to obtain the current connection URI. Since <i>RIOS.PeerConnection</i> is also evented, a control point may choose to subscribe to events instead of calling this action.</p>	<p><b>PeerConnection</b> The URI of the channel to which the Remote I/O client is currently connected. This value is empty if the client is currently disconnected.</p>	<p>OUT</p>	<p><i>PeerConnection</i></p>
<p><b>SetPeerInterlock</b></p> <p>This action is used to cause the Remote I/O client to connect to a user interface URI if, and only if,</p>	<p><b>PeerConnection</b> The URI to which the Remote I/O client should connect if the Remote I/O client is not currently connected.</p>	<p>IN</p>	<p><i>PeerConnection</i></p>

Action Name & Usage	Arguments & Description	IN / OUT	Related State Variable
<p>the Remote I/O client is not already connected to a user interface. This action should be used when connecting to a Remote I/O client that is not currently connected. Use of this action is encouraged as it mitigates the potential problem of multiple servers “racing” to connect to a single Remote I/O client.</p> <p>Any call to <b>RIOS:SetPeerInterlock</b> must set the <i>RIOS.PeerConnection</i> state variable to the requested connection URI. If at some later point the client fails to successfully connect to the requested connection URI, the client will reset the <i>RIOS.PeerConnection</i> state variable to indicate the EMPTY state, and an event will be issued.</p>	<p><b>ActivePeerConnection</b></p> <p>The URI of the current connection after the effect of this action invocation</p>	OUT	<i>PeerConnection</i>
<p><b>SetPeerOverride</b></p> <p>This action is used to cause the Remote I/O client to connect to a user interface URI. If the Remote I/O client is already connected, this action causes the previous connection to be dropped, and a new one to be established.</p> <p>Any call to <b>RIOS:SetPeerOverride</b> must set the peer connection state variable to the requested connection URI. If at some later point the client fails to successfully connect to the requested connection URI, the client will reset the <i>RIOS.PeerConnection</i> state variable to indicate the EMPTY state, and an event will be issued.</p>	<p><b>PeerConnection</b></p> <p>The URI to which the Remote I/O client should connect if the Remote I/O client is not currently connected.</p>	IN	<i>PeerConnection</i>

## 1.4 Channel Manager Service

The Intel **Channel Manager Service** is identified in the XML service description by the ServiceURN “**urn:schemas-upnp-org:service:ChannelManager:1**”. The Intel **Channel Manager Service** allows Remote I/O Control Points to publish a list of channels compatible with

the Intel Remote I/O client. It also allows control points to retrieve the complete list of compatible channels. This service is an evented state variable that events a string corresponding to the published channel list. All of the actions in this service are required. When referred to, the action names, state variables and arguments in the table below are preceded by 'CMS' to indicate that they are part of the Intel **Channel Manager Service**.

### 1.4.1 State Variables

Variable Name	Required or Optional	UPnP dataType	Default Value	Evented YES/NO
<i>ChannelName</i>	R	string	Not specified	NO
<i>RegisteredChannelList</i>	R	string	Not specified	YES
<i>ChannelTimeout</i>	R	int	Not specified	NO
<i>ChannelPeerConnection</i>	R	string	Not specified	NO

### 1.4.2 Actions

Action Name & Usage	Arguments & Description	IN / OUT	Related State Variable
<p><b>ClearAllChannels</b></p> <p>This action is used by control points to clear the list of published channels on the Remote I/O client. Generally, control points should monitor Remote I/O clients and re-register available channels if they are removed from the published list. Repeated use of this action is discouraged.</p>	No Arguments	N/A	N/A
<p><b>GetRegisteredChannelList</b></p> <p>This action is used to obtain the list of currently published channels compatible with this Remote I/O client. An XRT2 client device must support a minimum of 32 channels.</p>	<p><b>ChannelList</b></p> <p>The list of channels currently published on the Remote I/O client. This list is a text document where each odd line is the channel name and each even line is the channel URI. Each line is terminated with &lt;CR&gt;.</p>	OUT	<i>RegisteredChannelList</i>
<p><b>RegisterChannel</b></p> <p>This action is used to register a channel to the published channel list. A Remote I/O control point associated with an application must be reasonably assured that a channel is compatible with this Remote I/O client before registering the channel. Vendors</p>	<p><b>Name</b></p> <p>Friendly name for this channel. This name must never exceed 512 bytes.</p> <p><b>PeerConnection</b></p> <p>Channel URI. In the case of XRT2, the URI must start with "XRT2://". The channel URI must never exceed 1024 bytes in length.</p>	IN  IN	<i>ChannelName</i>  <i>ChannelPeerConnection</i>

Action Name & Usage	Arguments & Description	IN / OUT	Related State Variable
<p>should use the following criteria to establish compatibility between a remote-capable application and a Remote I/O client device:</p> <ul style="list-style-type: none"> <li>All information contained in the UPnP device description (Manufacturer, Manufacturer UR, Serial number, UDN, etc.)</li> <li>Vendor-specific information provided by the string parameter <i>RIOS.Application</i> in the <b>RIOS:GetDeviceInformation()</b> action.</li> </ul> <p>To keep the channel in the published channel list, a control point must invoke this action again before the channel timeout period expires.</p> <p>Channel URI's are always unique and are used as a key in the published channel list. Friendly names are not unique and are provided only for convenience.</p>	<p><b>Timeout</b></p> <p>The registration timeout in seconds. After this timeout period expires, the Remote I/O client will automatically remove the channel from the list of published channels.</p>	IN	<i>ChannelTimeout</i>
<p><b>UnregisterChannel</b></p> <p>This action causes a channel with a given URI to be removed from the published channel list.</p>	<p><b>PeerConnection</b></p> <p>Channel URI to be removed from the published list.</p>	IN	<i>ChannelPeerConnection</i>

## 1.5 Remote Input Service

The Intel **Remote Input Service** is identified in the XML service description by the ServiceURN “**urn:schemas-upnp-org:service:RemoteInput:1**”. The Intel **Remote Input Service** allows Remote I/O Control Points to send user input to the device just as if it had been initiated by the local user of the device. This service has no events. All of the actions in this service are required. When referred to, the action names, state variables and arguments in the table below are preceded by ‘RI’ to indicate that they are part of the Intel **Remote Input Service**.

### 1.5.1 State Variables

Variable Name	Required or Optional	UPnP dataType	Default Value	Evented YES/NO
<i>KeyCode</i>	R	int	Not specified	NO
<i>MousePositionX</i>	R	int	Not specified	NO
<i>MousePositionY</i>	R	int	Not specified	NO
<i>MouseButton</i>	R	int	Not specified	NO

## 1.5.2 Actions

Action Name & Usages	Arguments & Description	IN / OUT	Related State Variable
<p><b>InputKeyPress</b></p> <p>This action causes the Remote I/O client to consider this user input just as if it had been received from the local user.</p>	<p><b>Key</b></p> <p>Key code for the user Key Press input.</p>	IN	<i>KeyCode</i>
<p><b>InputMouseDown</b></p> <p>This action causes the Remote I/O client to consider this user mouse input just as if it had been received from the local user.</p>	<p><b>X</b></p> <p>X coordinate, in pixels, of the Mouse Down input.</p>	IN	<i>MousePositionX</i>
	<p><b>Y</b></p> <p>Y coordinate, in pixels, of the Mouse Down input.</p>	IN	<i>MousePositionY</i>
	<p><b>Button</b></p> <p>Button mask, 1 = left, 2 = right.</p>	IN	<i>MouseButton</i>
<p><b>InputMouseUp</b></p> <p>This action causes the Remote I/O client to consider this user mouse input just as if it had been received from the local user.</p>	<p><b>X</b></p> <p>X coordinate, in pixels, of the Mouse Up input.</p>	IN	<i>MousePositionX</i>
	<p><b>Y</b></p> <p>Y coordinate, in pixels, of the Mouse Up input.</p>	IN	<i>MousePositionY</i>
	<p><b>Button</b></p> <p>Button mask, 1 = left, 2 = right.</p>	IN	<i>MouseButton</i>

## 1.6 Remote IO – Theory of Operation

### 1.6.1.1 Discovery

The Intel Remote IO network solution is composed of any combination of UPnP Intel Remote IO clients and Intel UPnP client Control Points associated with Remote IO servers. When an Intel Remote IO client control point is first turned on, it issues a query to locate all Intel Remote IO client devices on the network. The query is issued in the form of an SSDP search for all devices of the following type:

*“urn:schemas-upnp-org:device:RemoteIO:1”.*

### 1.6.1.2 Description and Event Subscription

When an Intel Remote IO client is found on the network, an Intel Remote IO client control point subscribes to it in order to obtain events from the client’s **RemoteIO** (abbreviated as **RIOS**) and **ChannelManager** (abbreviated as **CMS**) services. The Remote IO client control point then calls the **RIOS:GetDeviceInformation** action to obtain additional information about the Remote IO client. This information supplements information obtained from the Remote IO client device description document.

When the RemoteIO client control point subscribes to events from a client device, it receives an initial value for each of the two evented state variables, *RIOS.PeerConnection* and *CMS.RegisteredChannelList*. From these values, the control point learns what URI the Remote IO client is connected to and what compatible Remote IO “channels” are available. As an alternate method to directly obtain the same information, the RemoteIO client control point may call **RIOS:GetPeerConnection** and **CMS:GetRegisteredChannelList**.

### 1.6.1.3 Determining Application Compatibility

Remote IO client control point vendors can use all the information provided by the Intel Remote IO client to determine compatibility between the client and remoteable applications. One flexible way to determine compatibility is to have an application programmatic API for compatibility checking within that application. The API is called with all the information on the targeted RemoteIO client and “true” is returned if the client is compatible, or “false” if not.

### 1.6.1.4 Registering and Administering Channels on the Remote Client

A Remote IO control point may find one or more compatible applications. For each compatible application, the Remote IO server registers the application with the Remote IO client using the **CMS:RegisterChannel** action in the **ChannelManager** service. The Remote IO client control point must call this action at periodic intervals—otherwise the channel will be removed from the client’s list of active channels. The registration duration is set in the *Timeout* argument (see F4.2, above) of the **CMS:RegisterChannel** action. Since a timeout value of 30 minutes is generally recommended, the Remote IO client control point must re-register at least every 25 minutes.

Remote IO client control points must keep monitoring the registered channel list on all Remote IO clients under control. If a compatible channel is not present on the list, it must be re-registered. When a control point calls **CMS:ClearAllChannels** on a Remote IO client, all compatible channels are immediately re-registered.

When the **RIOS:GetPeerConnection** action returns a NULL value for the *PeerConnection* argument and there is no built-in, local user interface on the client device, a Remote IO client control point may decide to automatically connect the Remote IO client to a compatible remoted application. When doing this, it must use the **RIOS:SetPeerInterlock** action in the RemoteIO service to avoid Remote IO “flicker”. This results when multiple Remote IO client control points all try to connect to the same user interface at the same time.

## 2 The Extended PC Remoting Technology Protocol v2 (XRT2)

---

### 2.1 Introduction

The XRT (Extended PC Remoting Technology) v2 protocol (XRT2) is a very simple TCP-based command encapsulation protocol for passing messages back and forth between two network nodes. It is most often used to send and receive display drawing commands and user input, but can also be used for other purposes.

### 2.2 Protocol Overview

#### 2.2.1 Command Encoding

The XRT2 protocol starts when a TCP connection is established between two network nodes. Each network node sends commands using the following format:

2 bytes    Command Length (including this header)  
2 bytes    Command Identifier  
n bytes    Command data

The command length, in bytes, must always equal the length of the data + 4 bytes, except for Jumbo commands (see below). When a command has no data, the command length field is set to 4. All fields are 2 bytes encoded in little-endian. All values are decimal values unless otherwise specifically noted.

When decoding commands, the network node will read the first two bytes to determine how long the command is, and read the rest of the command before processing it.

Each client has a maximum inbound command length that servers must never exceed. The maximum allowed inbound command length must be obtained using the Remote I/O client action **RIOS:GetDeviceInformation()**.

#### 2.2.2 Command Types

- Client commands are XRT2 commands that are sent from the server to the client.
- Server commands are XRT2 commands that are sent from the client to the server.

XRT2 Client commands are listed in the following table:

Command Name	Code
RESET	1
EXIT	5
JUMBO	6
DRAWFILLBOX	1004
DRAWIMAGE	1005
PING	3010

Command Name	Code
ALLOCATE	1007

XRT2 Server commands are listed in the following table:

Command Name	Code
REQUEST	2
EXIT	5
JUMBO	6
REPAINT	1003
KEY_PRESS	2003
MOUSE_DOWN	2005
MOUSE_UP	2006
PONG	3011

### 2.2.3 Jumbo Commands

Jumbo commands allow the total length of Client and Server commands to exceed the normal maximum of 64k bytes. To send Client and Server commands longer than 64k bytes in length, a Jumbo command is sent immediately prior to sending the oversized command. The Jumbo command is formatted as follows:

- 2 bytes Command Length, set to value 8 (2+2+4) in this case
- 2 bytes Command Identifier, set to value 6 (“Jumbo”) in this case
- 4 bytes Command data length, set to the total length in bytes of the next command. This value includes the 4 byte header in the total.

The command that follows the jumbo command is therefore formatted as follows:

- 2 bytes Set to zero
- 2 bytes The command code for the command to be oversized.
- Next *n* bytes The oversized data.

All fields are encoded in little-endian format.

Jumbo commands should never:

- Precede a command that is 64k bytes or shorter in length.
- Be sent to a device that is known to have a maximum allowed command size of 64k bytes or less.

## 2.3 Command Descriptions

### 2.3.1 Setup Commands

Command Name	Command length, bytes	Code	Data	Description
RESET	4	1	none	Client command

Command Name	Command length, bytes	Code	Data	Description
				Used to reset the Remote I/O client. Should only be used for debugging purposes.
REQUEST	4+n	2	UTF-8 encoded URI (n-bytes)	Server command Once a Remote I/O client establishes an XRT2 connection to a server, the first command from the client must be a REQUEST command. The parameter of the request command is a UTF-8 encoded, non-escaped URI designating the XRT2 connection.
EXIT	4	5	none	Client command If supported, this command causes the Remote I/O client to immediately disconnect and return to its local user interface.
JUMBO	8	6	Total length, in bytes, of the next command (4 bytes)	Used by both the client and server to send commands larger than 64k bytes in length. This command is only allowed to precede a command that is larger than 64k bytes.

### 2.3.2 Basic Display Remoting Commands

Basic display remoting commands are used by the sender to send display drawing primitives to the receiver.

Command Name	Command length, bytes	Code	Data	Description
REPAINT	4	1003	none	Server command This command causes the Remote I/O server to fully repaint the Remote I/O client display. This command is useful when the client loses the display content.
DRAWFILLBOX	4+11	1004	- X position of fill box (2 bytes) - Y position of fill box (2 bytes) - Width of fill box (2 bytes) - Height of fill box (2 bytes) - Red color component	Client command This command is used to draw a rectangle on the screen at a given position, size and color. The position and color of the rectangle must never exceed the boundaries of the client's display. Coordinate (0,0) references the upper leftmost pixel of the display. Parameters (X,Y) are with respect to the upper leftmost pixel of the display. Width and height parameters exclude the border. A '1x1' fill box for example would thus describe a single pixel.

Command Name	Command length, bytes	Code	Data	Description
			(1 byte) - Green color component (1 byte) - Blue color component (1 byte)	
DRAWIMAGE	4+4+n	1005	- X position of image (2 bytes) - Y position of image (2 bytes) - Image data (n bytes)	Client command Used to draw an image at a given position on the screen. The image position and size must never exceed the client's display boundaries. Coordinate (0,0) references the upper leftmost pixel of the display. Parameters (X,Y) are with respect to the upper leftmost pixel of the display.
ALLOCATE	4+10	1007	- X position of allocation (2 bytes) - Y position of allocation (2 bytes) - Width of allocation (2 bytes) - Height of allocation (2 bytes) - Allocation Instance Number (2 bytes)	Client command Used to specify area region on the client's display for a video window. The allocate command can also be used to define multiple display areas if supported by the client. Each video area must be controlled using a separate sink device. De-allocation is accomplished by allocating to (0,0,0,0). It is recommended to maintain an allocated region even if no video is playing. The allocation excludes the border, just like a FILLBOX. Allocation instance number '1' is recommended to be used to designate the primary video display region.

### 2.3.3 Input Remoting Commands

Input commands are used to transmit user input from an XRT2 client to an XRT2 server.

Command Name	Command length, bytes	Code	Data	Description
KEY_PRESS	4+4	2003	Key scan code (see above listing) (4 bytes)	Server command Send a 'key press' user input event.
MOUSE_DOWN	4+12	2005	- X position (4 bytes) - Y position (4 bytes) - Button mask (0x01 left, 0x02 right button) (4 bytes)	Server command Send a mouse or pointer position button 'press down' user input event.
MOUSE_UP	4+12	2006	- X position (4 bytes) - Y position (4 bytes) - Button mask (0x01 left, 0x02 right button) (4 bytes)	Server command Send a mouse or pointer position button 'press up' user input event.

### 2.3.4 Scan Codes for KEY\_PRESS Server Command

Hex-value scan codes used by the KEY\_PRESS command are shown in the following table. Vendors are allowed to extend XRT2 by adding vendor-specific scan codes. New scan codes must use values of 0x4000 or above.

Command	Scan Code
Up Arrow	0x61
Down Arrow	0x7a
Left Arrow	0x64
Right Arrow	0x66
Select	0x0D
Menu	0x6D
Play Pause	0x70
Previous Track	0x5B
Next Track	0x5D
Back	0x62
Page Up	0x68

Command	Scan Code
Page Down	0x6E
Options	0x6F

### 2.3.5 Additional Commands

Miscellaneous commands.

Command Name	Command length, bytes	Code	Description
PING	4	3010	Client command Causes the XRT2 client to send a PONG command back to the XRT2 server. This command should be sent to the client if no command has been sent to the client in the last 3 seconds in order to verify that the channel is still active. If the client does not respond to a PING with a PONG (below) within 5 seconds, it can be assumed that the channel is no longer active.
PONG	4	3011	Server command This command is only sent from the client to the XRT2 server in response to a PING command sent by the server.

### 2.3.6 Vendor Extensions to XRT2

Vendors are allowed to extend XRT2 by adding new command identifier codes and scan codes. New XRT2 commands above and beyond the ones defined in this table must use command identifier codes of 4000 or above. New scan codes must also use values of 4000 or above.

# Appendix A: Device and Service SCPD

---

## A1: Remote Device SCPD

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:RemotelIO:1</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelName>model number</modelName>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:RemotelIO:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:RemotelIO</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:ChannelManager:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:ChannelManager</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:RemotelInput:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:RemotelInput</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
      </service>
    </serviceList>
  </deviceList>
</deviceList>
  <presentationURL>URL for presentation</presentationURL>
</device>
</root>
```

## A2: Remote I/O Service SCPD

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>ForceDisconnection</name>
    </action>
    <action>
      <name>GetPeerConnection</name>
      <argumentList>
        <argument>
          <name>PeerConnection</name>
          <direction>out</direction>
          <relatedStateVariable>PeerConnection</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>ForceReset</name>
    </action>
    <action>
      <name>GetDeviceInformation</name>
      <argumentList>
        <argument>
          <name>Application</name>
          <direction>out</direction>
          <relatedStateVariable>ApplicationIdentifier</relatedStateVariable>
        </argument>
        <argument>
          <name>MaxCommandSize</name>
          <direction>out</direction>
          <relatedStateVariable>MaxCommandSize</relatedStateVariable>
        </argument>
        <argument>
          <name>DisplayEncoding</name>
          <direction>out</direction>
          <relatedStateVariable>DisplayEncoding</relatedStateVariable>
        </argument>
        <argument>
          <name>DisplayWidth</name>
          <direction>out</direction>
          <relatedStateVariable>DisplaySize</relatedStateVariable>
        </argument>
        <argument>
          <name>DisplayHeight</name>
          <direction>out</direction>
          <relatedStateVariable>DisplaySize</relatedStateVariable>
        </argument>
        <argument>
          <name>DeviceInformation</name>
          <direction>out</direction>
          <relatedStateVariable>DeviceInformation</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>SetPeerInterlock</name>
      <argumentList>
```

```

    <argument>
      <name>PeerConnection</name>
      <direction>in</direction>
      <relatedStateVariable>PeerConnection</relatedStateVariable>
    </argument>
    <argument>
      <name>ActivePeerConnection</name>
      <direction>out</direction>
      <relatedStateVariable>PeerConnection</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>SetPeerOverride</name>
  <argumentList>
    <argument>
      <name>PeerConnection</name>
      <direction>in</direction>
      <relatedStateVariable>PeerConnection</relatedStateVariable>
    </argument>
  </argumentList>
</action>
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="no">
    <name>MaxCommandSize</name>
    <dataType>ui4</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>ConnectionPort</name>
    <dataType>ui2</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>DeviceInformation</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>PeerConnection</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>ApplicationIdentifier</name>
    <dataType>string</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>DisplayEncoding</name>
    <dataType>int</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>DisplaySize</name>
    <dataType>ui4</dataType>
  </stateVariable>
</serviceStateTable>
</scpd>

```

## A3: Channel Manager Service SCPD

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>RegisterChannel</name>
      <argumentList>
        <argument>
          <name>Name</name>
          <direction>in</direction>
          <relatedStateVariable>ChannelName</relatedStateVariable>
        </argument>
        <argument>
          <name>PeerConnection</name>
          <direction>in</direction>
          <relatedStateVariable>ChannelPeerConnection</relatedStateVariable>
        </argument>
        <argument>
          <name>Timeout</name>
          <direction>in</direction>
          <relatedStateVariable>ChannelTimeout</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>UnregisterChannel</name>
      <argumentList>
        <argument>
          <name>PeerConnection</name>
          <direction>in</direction>
          <relatedStateVariable>ChannelPeerConnection</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>ClearAllChannels</name>
    </action>
    <action>
      <name>GetRegisteredChannelList</name>
      <argumentList>
        <argument>
          <name>ChannelList</name>
          <direction>out</direction>
          <relatedStateVariable>RegisteredChannelList</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>ChannelName</name>
      <dataType>string</dataType>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>RegisteredChannelList</name>
      <dataType>string</dataType>
    </stateVariable>
  </serviceStateTable>
</scpd>
```

```

    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ChannelTimeout</name>
      <dataType>int</dataType>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ChannelPeerConnection</name>
      <dataType>string</dataType>
    </stateVariable>
  </serviceStateTable>
</scpd>

```

## A4: Remote Input Service SCPD

```

<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>InputKeyDown</name>
      <argumentList>
        <argument>
          <name>key</name>
          <direction>in</direction>
          <relatedStateVariable>KeyCode</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>InputKeyUp</name>
      <argumentList>
        <argument>
          <name>key</name>
          <direction>in</direction>
          <relatedStateVariable>KeyCode</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>InputMouseUp</name>
      <argumentList>
        <argument>
          <name>X</name>
          <direction>in</direction>
          <relatedStateVariable>MousePositionX</relatedStateVariable>
        </argument>
        <argument>
          <name>Y</name>
          <direction>in</direction>
          <relatedStateVariable>MousePositionY</relatedStateVariable>
        </argument>
        <argument>
          <name>Button</name>
          <direction>in</direction>
          <relatedStateVariable>MouseButton</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>

```

```

<action>
  <name>InputMouseMove</name>
  <argumentList>
    <argument>
      <name>X</name>
      <direction>in</direction>
      <relatedStateVariable>MousePositionX</relatedStateVariable>
    </argument>
    <argument>
      <name>Y</name>
      <direction>in</direction>
      <relatedStateVariable>MousePositionY</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>GetInputSetup</name>
  <argumentList>
    <argument>
      <name>InputSetupIdentifier</name>
      <direction>out</direction>
      <relatedStateVariable>InputSetupIdentifier</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>InputKeyPress</name>
  <argumentList>
    <argument>
      <name>key</name>
      <direction>in</direction>
      <relatedStateVariable>KeyCode</relatedStateVariable>
    </argument>
  </argumentList>
</action>
<action>
  <name>InputMouseDown</name>
  <argumentList>
    <argument>
      <name>X</name>
      <direction>in</direction>
      <relatedStateVariable>MousePositionX</relatedStateVariable>
    </argument>
    <argument>
      <name>Y</name>
      <direction>in</direction>
      <relatedStateVariable>MousePositionY</relatedStateVariable>
    </argument>
    <argument>
      <name>Button</name>
      <direction>in</direction>
      <relatedStateVariable>MouseButton</relatedStateVariable>
    </argument>
  </argumentList>
</action>
</actionList>
<serviceStateTable>
  <stateVariable sendEvents="no">
    <name>KeyCode</name>
    <dataType>int</dataType>
  </stateVariable>
  <stateVariable sendEvents="no">
    <name>InputSetupIdentifier</name>
    <dataType>string</dataType>
  </stateVariable>
</serviceStateTable>

```

```
</stateVariable>
<stateVariable sendEvents="no">
  <name>MousePositionX</name>
  <dataType>int</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>MousePositionY</name>
  <dataType>int</dataType>
</stateVariable>
<stateVariable sendEvents="no">
  <name>MouseButton</name>
  <dataType>int</dataType>
</stateVariable>
</serviceStateTable>
</scpd>
```