UPnP AV Architecture:1

For UPnP Device Architecture 1.0
Status: Standardized DCP
Date: June 12th, 2002

This Standardized DCP has been adopted as a Standardized DCP by the Steering Committee of the UPnP Forum, pursuant to Section 2.1(c)(ii) of the UPnP Forum Membership Agreement. UPnP Forum Members have rights and licenses defined by Section 3 of the UPnP Forum Membership Agreement to use and reproduce the Standardized DCP in UPnP Compliant Devices. All such use is subject to all of the provisions of the UPnP Forum Membership Agreement.

THE UPNP FORUM TAKES NO POSITION AS TO WHETHER ANY INTELLECTUAL PROPERTY RIGHTS EXIST IN THE STANDARDIZED DCPS. THE STANDARDIZED DCPS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". THE UPNP FORUM MAKES NO WARRANTIES, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE STANDARDIZED DCPS, INCLUDING BUT NOT LIMITED TO ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE, OF REASONABLE CARE OR WORKMANLIKE EFFORT, OR RESULTS OR OF LACK OF NEGLIGENCE.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

Author	Company
John Ritchie	Intel Corporation
Thomas Kuehnel	Microsoft Corporation

Contents

1. INTRODUCTION 3

2. GOALS 3

3. NON-GOALS 3

4. OVERVIEW 3

5. PLAYBACK ARCHITECTURE 5

5.1. MEDIA SERVER 6
5.1.1. Content Directory Service 7
5.1.2. ConnectionManager Service 7
5.1.3. AVTransport Service 7
5.2. MEDIARENDERER 7
5.2.1. RenderingControlService 8
5.2.2. ConnectionManagerService 8
5.2.3. AVTransport Service 8
5.3. CONTROL POINT 9

6. EXAMPLE PLAYBACK SCENARIOS 11

6.1. ISOCHRONOUS-PUSH TRANSFER PROTOCOLS (IEC61883 / IEEE1394) 11
6.2. ASYNCHRONOUS-PULL TRANSFER PROTOCOLS (E.G. HTTP GET) 12
6.3. NO CM::PREPAREFORCONNECTION() ACTION 15
6.4. RENDERER COMBO DEVICE USING ISOCHRONOUS-PUSH (E.G. IEEE-1394) 17
6.5. RENDER COMBO DEVICE USING ASYNCHRONOUS-PUSH (E.G. HTTP GET) 18
6.6. SERVER COMBO DEVICE USING ASYNCHRONOUS-PUSH (E.G. HTTP GET) 18
6.7. SERVER COMBO DEVICE USING ISOCHRONOUS-PUSH (E.G. IEEE-1394) 20
6.8. SIMPLEST INTERACTION MODEL SUPPORTED 21

7. RECORDING ARCHITECTURE 21

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

1. Introduction

This document describes the overall UPnP AV Architecture, which forms the foundation for the UPnP AV Device and Service templates. The AV Architecture defines the general interaction between UPnP Control Points and UPnP AV devices. It is independent of any particular device type, content format, and transfer protocol. It supports a variety of devices such as TVs, VCRs, CD/DVD players/jukeboxes, settop boxes, stereos systems, MP3 players, still-image cameras, camcorders, electronic picture frames (EPFs), and the PC. The AV Architecture allows devices to support different types of formats for the entertainment content (such as MPEG2, MPEG4, JPEG, MP3, Windows Media Architecture (WMA), bitmaps (BMP), NTSC, PAL, ATSC, etc.) and multiple types of transfer protocols (such as IEC-61883/IEEE-1394, HTTP GET, RTP, HTTP PUT/POST, TCP/IP, etc.). The following sections describe the AV Architecture and how the various UPnP AV devices and services work together to enable various end-user scenarios.

2. Goals

The UPnP AV Architecture was explicitly defined to meet the following goals:

- To support arbitrary transfer protocols and content formats.
- To enable the AV content to flow directly between devices without any intervention from the Control Point.
- To enable Control Points to remain independent of any particular transfer protocol and content format. This allows Control Points to transparently support new protocols and formats.
- Scalability, i.e. support of devices with very low resources, especially memory and processing power as well as full-featured devices.

3. Non-Goals

The UPnP AV Architecture does not enable any of the following:

- Two-way Interactive Communication, such as audio and video conferencing, Internet gaming, etc.
- Access Control, Content Protection, and Digital Rights Management
- Synchronized playback to multiple rendering devices.

4. Overview

In most (non-AV) UPnP scenarios, a Control Point controls the operation of one or more UPnP devices in order to accomplish the desired behavior. Although the Control Point is managing multiple devices, all interactions occur in isolation between the Control Point and each device. The Control Point coordinates the operation of each device to achieve an overall, synchronized, end-user effect. The individual devices do not interact directly with each another. All of the coordination between the devices is performed by the Control Point and not the devices themselves.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

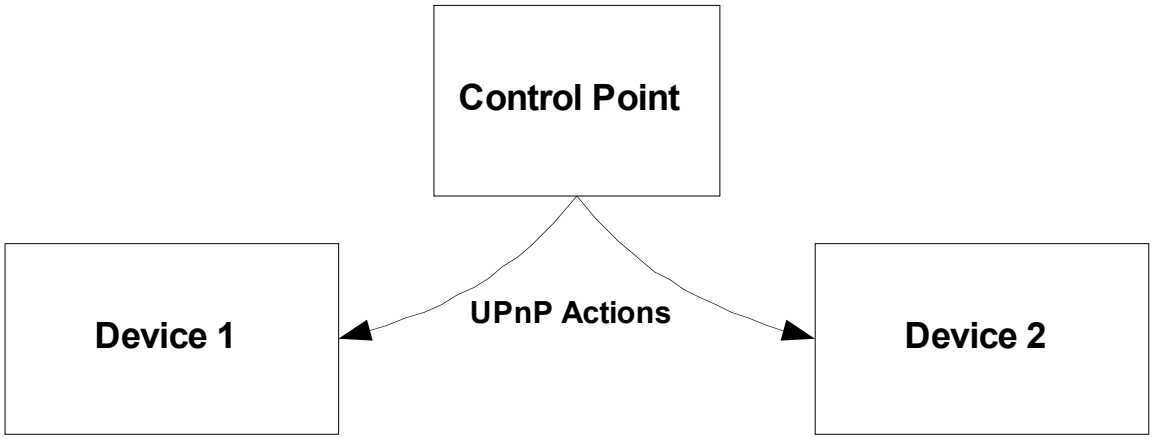


Figure 1: Typical UPnP Device Interaction Model

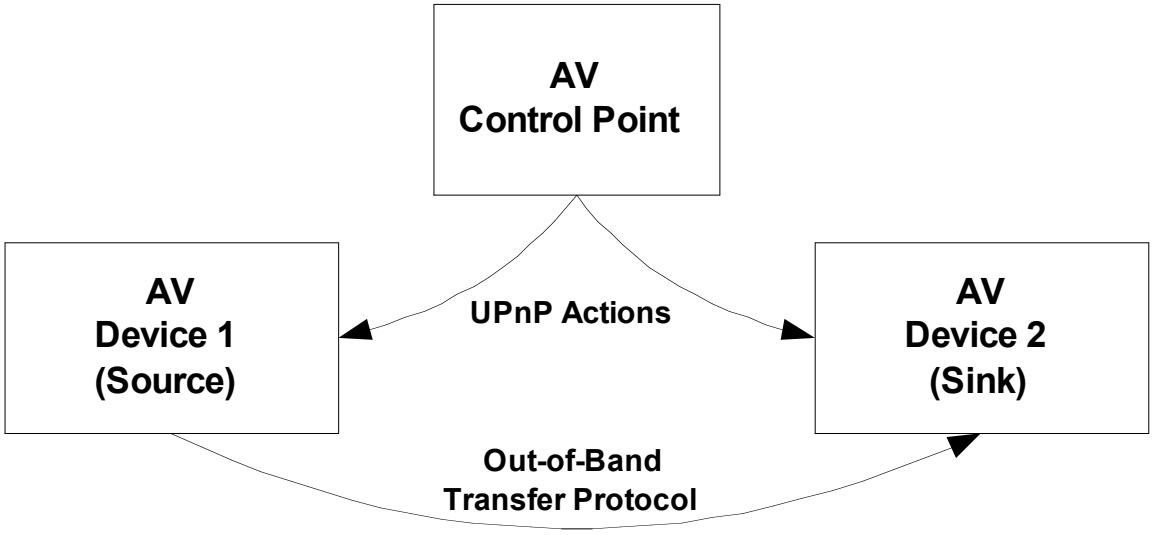


Figure 2: UPnP AV Device Interaction Model

Most AV scenarios involve the flow of (entertainment) content (i.e. a movie, song, picture, etc.) from one device to another. As shown in Figure 2, an AV Control Point interacts with two or more UPnP devices acting as source and sink, respectively. Although the Control Point coordinates and synchronizes the behavior of both devices, the devices themselves interact with each other using a non-UPnP ("out-of-band") communication protocol. The Control Point uses UPnP to initialize and configure both devices so that the desired content is transferred from one device to the other. However, since the content is transferred using an "out-of-band" transfer protocol, the Control Point is not directly involved in the actual transfer of the content. The Control Point configures the devices as needed, triggers the flow of content, then gets out of the way. Thus, after the transfer has begun, the Control Point can be disconnected without disrupting the flow of content. In other words, the core task (i.e. transferring the content) continues to function even without the Control Point present.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

As described in the above scenario, three distinct entities are involved: the Control Point, the source of the media content (called the "MediaServer"), and the sink for the content (called the "MediaRenderer"). Throughout the remainder of the document, all three entities are described as if they were independent devices on the network. Although this configuration may be common (i.e. a remote control, a VCR, and a TV), the AV Architecture supports arbitrary combinations of these entities within a single physical device. For example, a TV can be treated as a rendering device (e.g. a display). However, since most TVs contain a built-in tuner, the TV can also act as a server device because it could tune to a particular channel and send that content to a MediaRenderer (e.g. its local display or some remote device such as a tuner-less display). Similarly, many MediaServers and/or MediaRenderers may also include Control Point functionality. For example, an MP3 Renderer will likely have some UI controls (e.g. a small display and some buttons) that allow the user to control the playback of music.

5. Playback Architecture

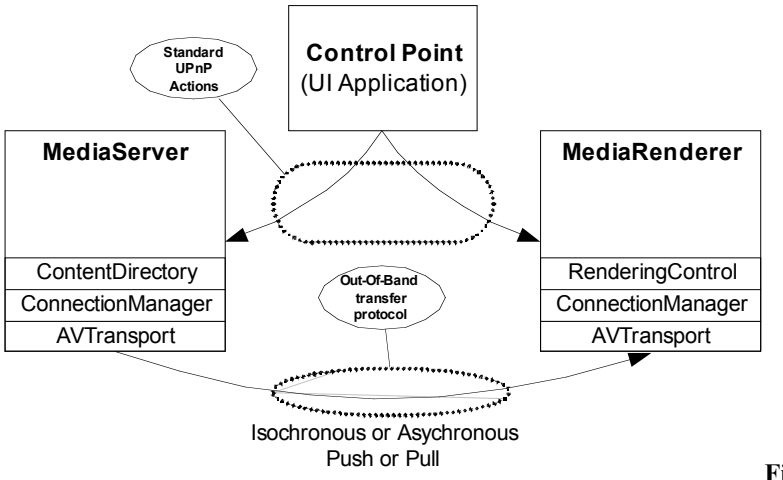


Figure 3

The most common task that end-users want to perform is to render (i.e. play) individual items of content on a specific rendering device. As shown in,

Figure 3, a content playback scenario involves three distinct UPnP components: a MediaServer, a MediaRenderer, and a UPnP Control Point. These three components (each with a well-defined role) work together to accomplish the task. In this scenario, the MediaServer contains (entertainment) content that the user wants to render (e.g. display or listen to) on the MediaRenderer. The user interacts with the Control Point's UI to locate and select the desired content on the MediaServer and to select the target MediaRenderer.

The MediaServer contains or has access to a variety of entertainment content, either stored locally or stored on an external device that is accessible via the MediaServer. The MediaServer is able to access its content and transmit it to another device via the network using some type of transfer protocol. The content exposed by the MediaServer may include arbitrary types of content including video, audio, and/or still images. The content is transmitted over the network using a transfer protocol and data format that is that is understood by the MediaServer and MediaRenderer. MediaServers may support one or multiple transfer protocols and data formats for each content item or be able to convert the format of a given content item into another formats on the fly. Examples of a MediaServer include a VCR, CD/DVD player/jukebox, camera, camcorder, PC, set-top box, satellite receiver, audio tape player, etc.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

The MediaRenderer obtains content from a MediaServer via network. Examples of a MediaRenderer include TV, stereo, network-enabled speakers, MP3 players, Electronic Picture Frame (EPF), a music-controlled water fountain, etc. The type of content that a MediaRenderer can receive depends on the transfer protocols and data formats that it supports. Some MediaRenderers may only support one type of content (e.g. audio or still images), where as other MediaRenderers may support a wide variety of content including video, audio, still images.

The Control Point coordinates and manages the operation of the MediaServer and MediaRenderer as directed by the user (e.g. play, stop, pause) in order to accomplish the desired task (e.g. play "MyFavorite" music). Additionally, the Control Point provides the UI (if any) for the user to interact with in order to control the operation of the device(s) (e.g. to select the desired content). The layout of the Control Point's UI and the functionality that it exposes is implementation dependent and determined solely by the Control Point's manufacturer. Some examples of a Control Point might include a TV with a traditional remote control or a wireless PDA-like device with a small display.

Note: The above descriptions talk about devices "sending/receiving content to/from the home network." In the context of the AV Architecture, this includes point-to-point connections such as an RCA cable that is used to connect a VCR to a TV. The AV Architecture treats this type of connection as a small part (e.g. segment) of the home network. Refer to the ConnectionManager Service Template for more details [?].

As described above, the AV Architecture consists of three distinct components that perform well-defined roles. In some cases, these components will exist as separate, individual UPnP devices. However, this need not be the case. Device manufacturers are free to combine any of these logical entities into a single physical device. In such cases, the individual components of these combo devices may interact with each other using either the standard UPnP control protocols (e.g. SOAP over HTTP) or using some private communication mechanism. In either case, the function of each logical entity remains unchanged. However, in the later case, since the communication between the logical entities is private, the individual components will not be able to communicate with other UPnP AV devices that do not implement the private protocol.

As shown in Figure 3, the Control Point is the only component that initiates UPnP actions. The Control Point requests to configure the MediaServer and MediaRenderer so that the desired content flows from the MediaServer to the MediaRenderer (using one of the transfer protocols and data formats that are supported by both the MediaServer and MediaRenderer). The MediaServer and MediaRenderer do invoke any UPnP actions to the Control Point. However, if needed, the MediaServer and/or MediaRenderer may send event notifications to the Control Point in order to inform the Control Point of a change in the MediaServer's/MediaRenderer's internal state.

The MediaServer and MediaRenderer do not control each other via UPnP actions. However, in order to transfer the content, the MediaServer and MediaRenderer use an "out-of-band" (e.g. a non-UPnP) transfer protocol to directly transmit the content. The Control Point is not involved in the actual transfer of the content. It simply configures the MediaServer and MediaRenderer as needed and initiates the transfer of the content. Once the transfer begins, the Control Point "gets out of the way" and is no longer needed to complete the transfer.

However, if desired by the user, the Control Point is capable of controlling the flow of the content by invoking various AVTransport actions such as Stop, Pause, FF, REW, Skip, Scan, etc. Additionally, the Control Point is also able to control the various rendering characteristics on the Renderer device such as Brightness, Contrast, Volume, Balance, etc.

5.1. Media Server

The MediaServer is used to locate content that is available via the home network. MediaServers include a wide variety of devices including VCRs, DVD players, satellite/cable receivers, TV tuners, radio tuners, CD players, audio tape players, MP3 players, PCs, etc. A MediaServer's primary purpose is to allow Control Points to enumerate (i.e. browse or search for) content items that are available for the user to

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

render. The MediaServer contains a ContentDirectoryService, a ConnectionManager Service, and an optional AVTransport Service (depending on the supported transfer protocols).

Some MediaServers are capable of transferring multiple content items at the same time, e.g. a hard-disk-based audio jukebox may be able to simultaneously stream multiple audio files to the network. In order to support this type of MediaServer, the ConnectionManager assigns a unique Connection ID to each "connection" (i.e. each stream) that is made. This ConnectionID allows a third-party Control Points to obtain information about active connections of the MediaServer.

5.1.1. Content Directory Service

This service provides a set of actions that allow the Control Point to enumerate the content that the Server can provide to the home network. The primary action of this service is **Browse()**. This action allows Control Points to obtain detailed information about each Content Item that the Server can provide. This information (i.e. meta-data) includes properties such as its name, artist, date created, size, etc. Additionally, the returned meta-data identifies the transfer protocols and data formats that are supported by the Server for that particular Content Item. The Control Point uses this information to determine if a given MediaRenderer is capable of rendering that content in its available format.

5.1.2. ConnectionManager Service

This service is used to manage the connections associated with a particular device. The primary action of this service (within the context of a MediaServer) is **PrepareForConnection()**. When implemented, this optional action is invoked by the Control Point to give the Server an opportunity to prepare itself for an upcoming transfer. Depending on the specified transfer protocol and data format, this action may return the InstanceID of an AVTransport service that the Control Point can use to control the flow of this content (e.g. Stop, Pause, Seek, etc). As described below, this InstanceID is used to distinguish multiple (virtual) instances of the AVTransport service, each of which is associated with a particular connection to Renderer. Multiple (virtual) instances of the AVTransport service allow the MediaServer to support multiple Renderers at the same time. When the Control Point wants to terminate this connection, it should invoke the MediaServer's **ConnectionComplete()** action (if implemented) to release the connection.

If the **PrepareForConnection()** action is not implemented, the Control Point is only able to support a single Renderer at an given time. In this case, the Control Point should use InstanceID=0.

5.1.3. AVTransport Service

This (optional) service is used by the Control Point to control the "playback" of the content that is associated with the specified AVTransport. This includes the ability to Stop, Pause, Seek, etc. Depending on the supported transfer protocols and/or data formats, a MediaServer may or may not implement this service. If supported, the MediaServer can distinguish between multiple instances of the service by using the InstanceID that is included in each AVTransport action. New instances of the AVTransport service are created via the ConnectionManager's **PrepareForConnection().** A new Instance Id is allocated for each new Service Instance.

5.2. MediaRenderer

The MediaRenderer is used to render (e.g. display and/or listen to) content obtained from the home network. This includes a wide variety of devices including TVs, stereos, speakers, hand-held audio players, music controlled water-fountain, etc. Its main feature is that it allows the Control Point to control how content is rendered (e.g. Brightness, Contrast, Volume, Mute, etc). Additionally, depending on the transfer protocol that is being used to obtain the content from the network, the MediaRenderer may also allow the user to control the flow of the content (e.g. Stop, Pause, Seek, etc). The MediaRenderer includes

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

a Rendering Control Service, a ConnectionManager Service, and an optional AVTransport Service (depending on which transfer protocols are supported).

In order to support rendering devices that are capable of handling multiple content items at the same time (e.g. an audio mixer such as a Karaoke device), the Rendering Control and AVTransport Services contain multiple independent (logical) instances of these services. Each (logical) instance of these services is bound to a particular incoming connection. This allows the Control Point to control each incoming content independently from each other.

Multiple logical instances of these services are distinguished by a unique 'InstanceID' which references the logical instance. Each action invoked by the Control Point contains the Instance ID that identifies the correct instance..

5.2.1. RenderingControlService

This service provides a set of actions that allow the Control Point to control how the Renderer renders a piece of incoming content. This includes rendering characteristics such as Brightness, Contrast, Volume, Mute, etc. The Rendering Control service supports multiple, dynamic instances, which allows a Renderer to "mix together" one or more content items (e.g. a Picture-in-Picture window on a TV or an audio mixer device). New instances of the service are created by the ConnectionManager's **PrepareForConnection()** action.

5.2.2. ConnectionManagerService

This service is used to manage the connections associated with a device. Within the context of a MediaRenderer, the primary action of this service is the **GetProtocolInfo()** action. This action allows a Control Point to enumerate the transfer protocols and data formats that are supported by the MediaRenderer. This information is used to predetermine if a MediaRenderer is capable of rendering a specific content item. A MediaRenderer may also implement the optional **PrepareForConnection()** action. This action is invoked by the Control Point to give the Render an opportunity to prepare itself for an upcoming transfer. Additionally, this action assigns a unique ConnectionID that can be used by a 3rd-party Control Point to obtain information about the connections that the MediaRenderer is using. Also, depending on the specified transfer protocol and data format being used, this action may return a unique AVTransport InstanceID that the Control Point can use to control the flow of the content (e.g. Stop, Pause, Seek, etc). (Refer to the AVTransport section below for additional details). Lastly, **PrepareForConnection()** also returns a unique Rendering Control InstanceID which can be used by the Control Point to control the rendering characteristics of the associated content as described above. When the Control Point wants to terminate a connection, it should invoke the Renderer's **ConnectionComplete()** action (if implemented) to release the connection.

5.2.3. AVTransport Service

This (optional) service is used by the Control Point to control the flow of the associated content. This includes the ability to Play, Stop, Pause, Seek, etc. Depending on transfer protocols and/or data formats that are supported, the Renderer may or may not implement this service. In order to support MediaRenderers that can simultaneously handle multiple content items, the AVTransport service may support multiple logical instances of this service. As described above, AVTransport InstanceIDs are allocated by the ConnectionManager's **PrepareForConnection()** action to distinguish between multiple service instances.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

5.3. Control Point

Control Points coordinate the operation of the MediaServer and the MediaRenderer, usually in response to user interaction with the Control Point's UI. The following describes a generic Control Point algorithm that can be used to interact with a wide variety of MediaServer and MediaRenderer implementations.

1. **Discover AV Devices:** Using UPnP's Discovery mechanism, MediaServers and MediaRenderers in the home network are discovered.

2. **Locate Desired Content:** Using the Server's ContentDirectory::Browse() or Search() actions, a desired Content Item is located. The information returned by Browse()/Search() includes the transfer protocols and data formats that the MediaServer supports to transfer the content to the home network.

3. **Get Renderer's Supported Protocols/Formats:** Using the MediaRenderer's ConnectionManager::GetProtocolInfo() action a list of transfer protocols and data formats supported by the MediaRenderer is returned to the Control Point.

4. **Compare/Match Protocols/Formats:** The protocol/format information returned by the ContentDirectory for the desired Content Item is matched with the protocol/format information returned by the MediaRenderer's GetProtocolInfo() action. The Control Point selectsa transfer protocol and data format that are supported by both the MediaServer and MediaRenderer.

5. **Configure Server/Renderer:** The device's ConnectionManager::PrepareForConnection() action (if implemented) informs the MediaServer and MediaRenderer that an outgoing/incoming connection is about to be made using the specified transfer protocol and data format that was previously selected. Depending on the selected transfer protocol, either the MediaServer or MediaRenderer will return an AVTransport InstanceID. This InstanceID is used in conjunction with the device's AVTransport Service (i.e. the device returning the AVTransport InstanceID) to control the flow of the content (e.g. Play, Stop, Pause, Seek, etc). Additionally, the Renderer will return a Rendering Control InstanceID that is used by the Control Point to control the Rendering characteristics of the content.

   Note: Since PrepareForConnection() is an optional action, there may be situations in which either the MediaServer and/or MediaRenderer do not implement PrepareForConnection(). When this occurs and neither MediaServer nor MediaRenderer return an AVTransport InstanceID, the Control Point uses an InstanceID=0 to control the flow of the content. Refer to the ConnectionManager and AVTransport Service Templates for details [?].

6. **Select Desired Content:** Using the AVTransport service (whose InstanceID is returned by either the Server or Renderer), invoke the SetAVTransportURI() action to identify the content item that needs to be transferred.

7. **Start Content Transfer:** Using the AVTransport service, invoke one of the transport control actions as desired by the user (e.g. Play, Stop, Seek, etc).

8. **Adjust Rendering Characteristics:** Using the MediaRenderer's Rendering Control service, invoke any rendering control actions as desired by the user (e.g. adjust brightness, contrast, volume, mute, etc).

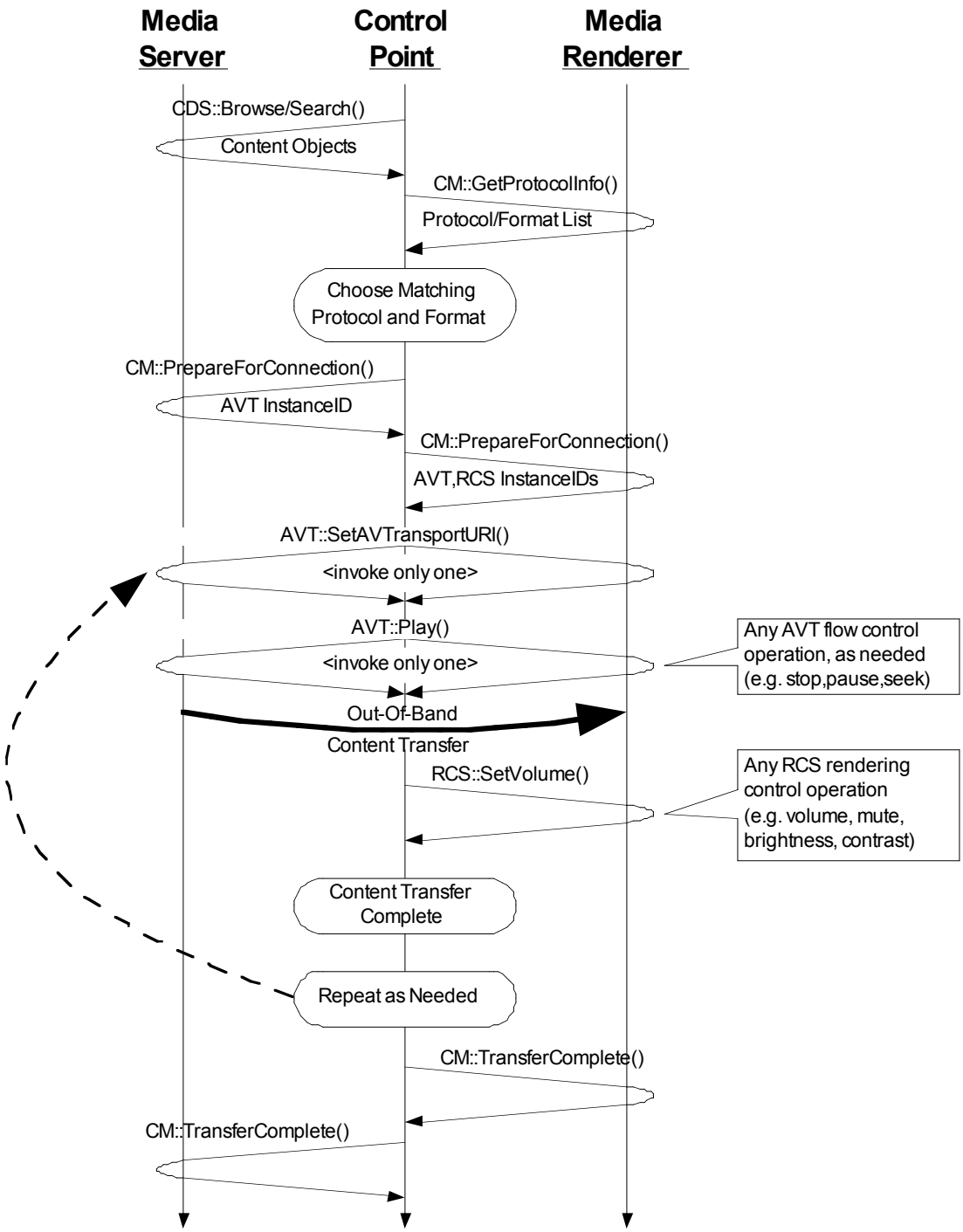
9. **Repeat: Select Next Content:** Using either the AVTransport::SetAVTransportURI() or AVTransport::SetNextAVTRansportURI() actions, identify the next content item that is to be transferred from the same Server to the same Renderer. Repeat as needed.

10. **Cleanup Server/Renderer:** When the session is terminated and MediaServer and MediaRenderer are no longer needed in the context of the session, the MediaServer's and MediaRenderer's ConnectionMgr::ConnectionComplete() action is invoked to close the MediaServer's connection.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

Based on the interaction sequence shown above, the following diagram chronologically illustrates the typical interaction sequence between the three Control Point and the MediaServer and MediaRenderer.

Playback: General Interaction Diagram



© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

6. Example Playback Scenarios

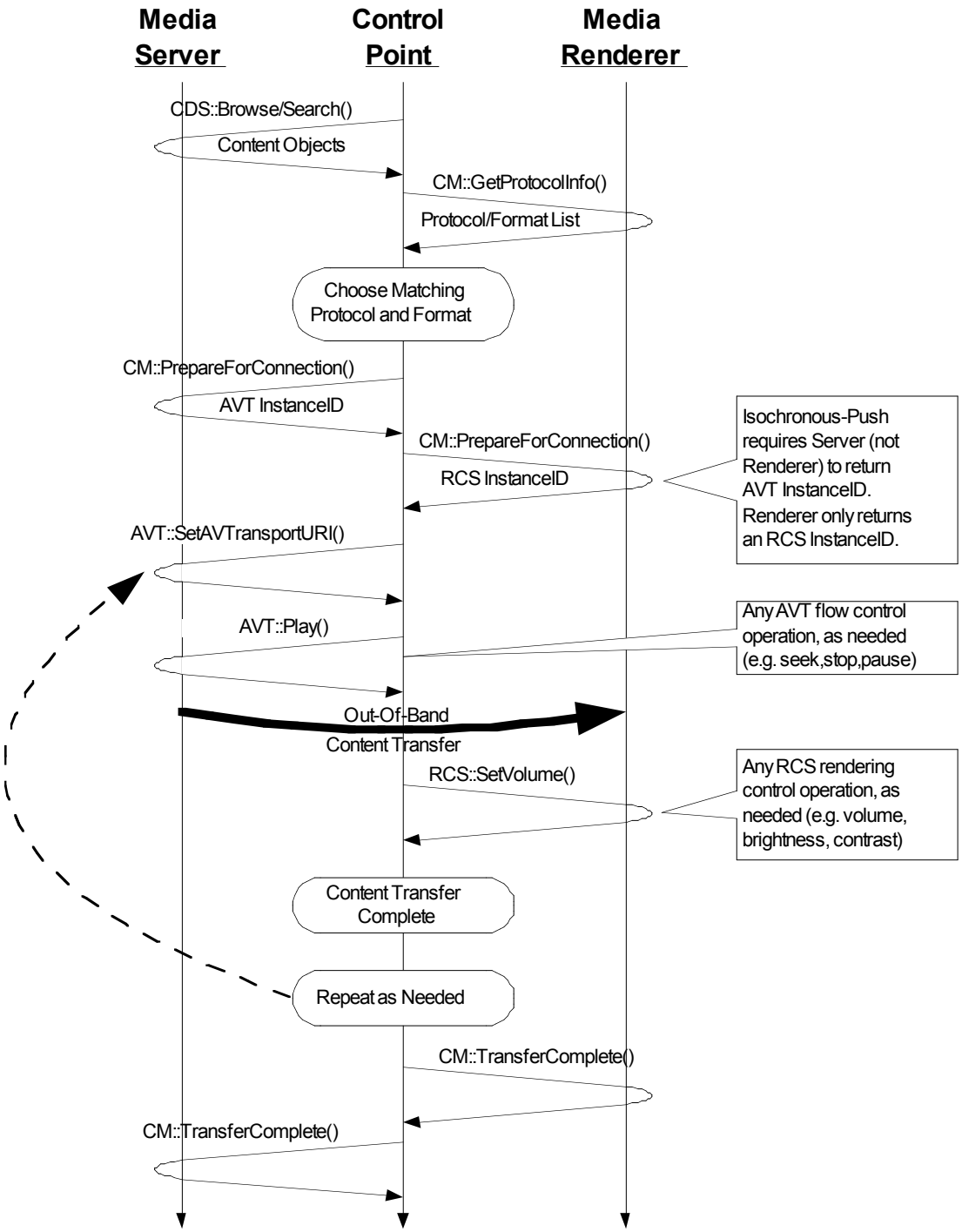
As described above, the AV Architecture is designed to support arbitrary transfer protocols and data formats. However, in some cases, certain devices are intentionally designed to support a single transfer protocol and/or data format only. For example, a manufacturer may want to deliver a product that targets a particular price-point and/or market segment. In these cases, some AV devices may combine one or more logical entities into a single physical device.

The following sub-sections illustrate the flexibility of the generic Device Interaction Model algorithm. Each of the following interaction diagrams are variations of the generic diagram with various steps omitted. These omitted steps are not included because the particular scenario does not require them.

6.1. Isochronous-Push Transfer Protocols (IEC61883 / IEEE1394)

When using an isochronous transfer protocol (e.g.IEC61883/ IEEE1394), the underlying transfer mechanism provides real-time content transfer between the MediaServer and MediaRenderer. This ensures that individual packets of content are transferred within a certain (relatively small) period of time. This real-time behavior allows the MediaRenderer to provide the user with smooth-flowing rendering of the content without implementing a read-ahead buffer. In this environment, the flow of the content is controlled by the MediaServer. The MediaRenderer immediately renders the content that it receives from the MediaServer. Refer to the diagram below for details.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.



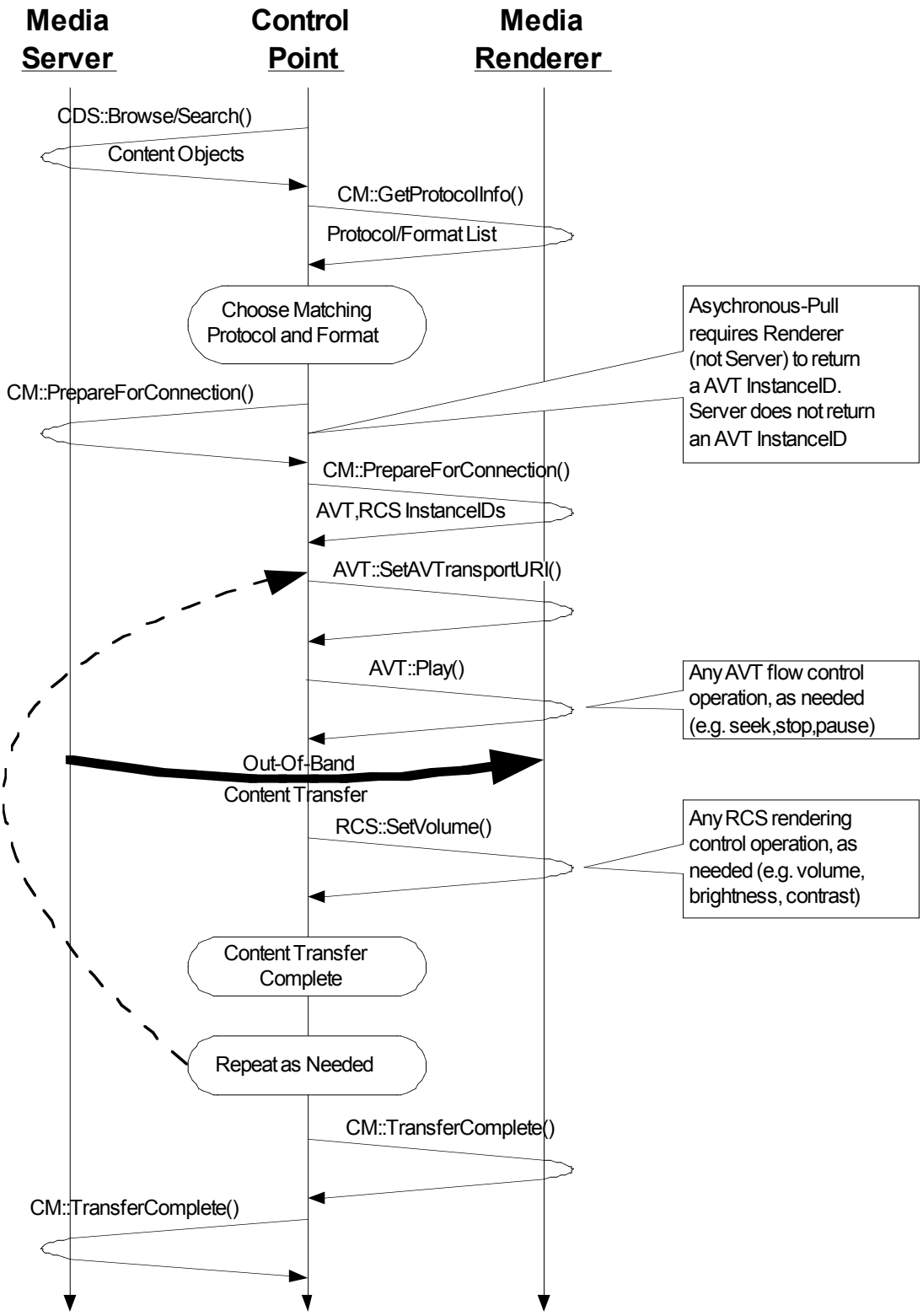
6.2. Asynchronous-Pull Transfer Protocols (e.g. HTTP GET)

In this example, the transfer protocols that are used do not provide real-time guarantees. The arrival of a particular packet of content is unpredictable relative to the previous packets. Unless corrected, this causes the content to be rendered with certain undesirable anomalies (e.g. detectable latencies, jitter, etc.). In

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

order to compensate for these types of transfer mechanisms, a Renderer device typically implements a read-ahead storage buffer in which the Renderer reads-ahead of the current output and places the data into a buffer until the contents are needed. This allows the MediaRenderer to smooth out any rendering anomalies that might otherwise exist. Since the MediaRenderer must control the flow of the content, it is obligated to provide the instance of the AVTransport service that will be used.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

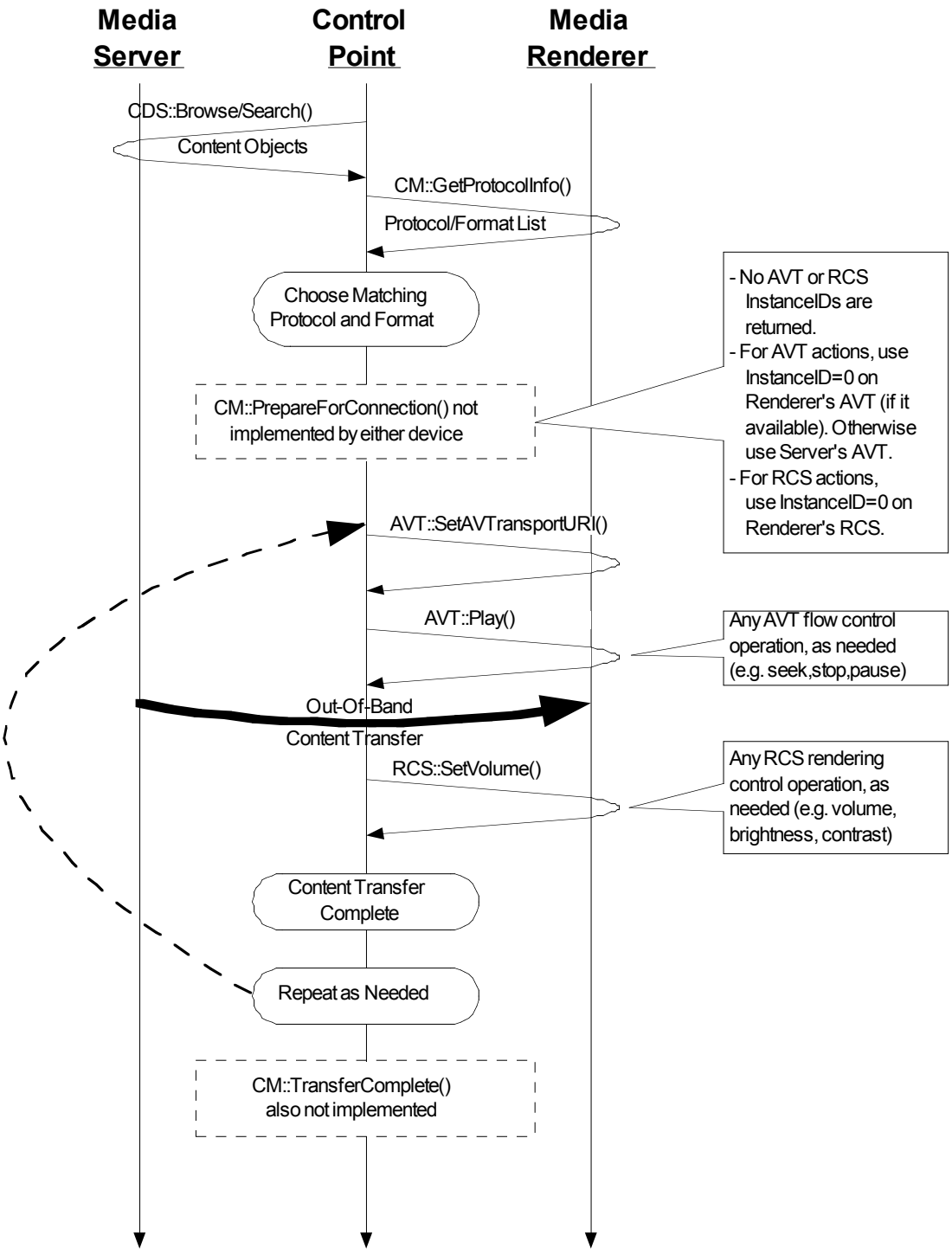


© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

6.3. No CM::PrepareForConnection() Action

In some circumstances, vendors may choose to not implement the PrepareForConnection() action, which (among other tasks) provides a mechanism for the Control Point to obtain the InstanceID of the AVTransport and Rendering Control Service to use for controlling the flow and rendering characteristics of the content. When the PrepareForConnection() action is not implemented, the Control Point must "fall-back" and assume an InstanceID=0. The following diagram illustrates how the general Device Interaction Model gracefully scales to handle this situation.

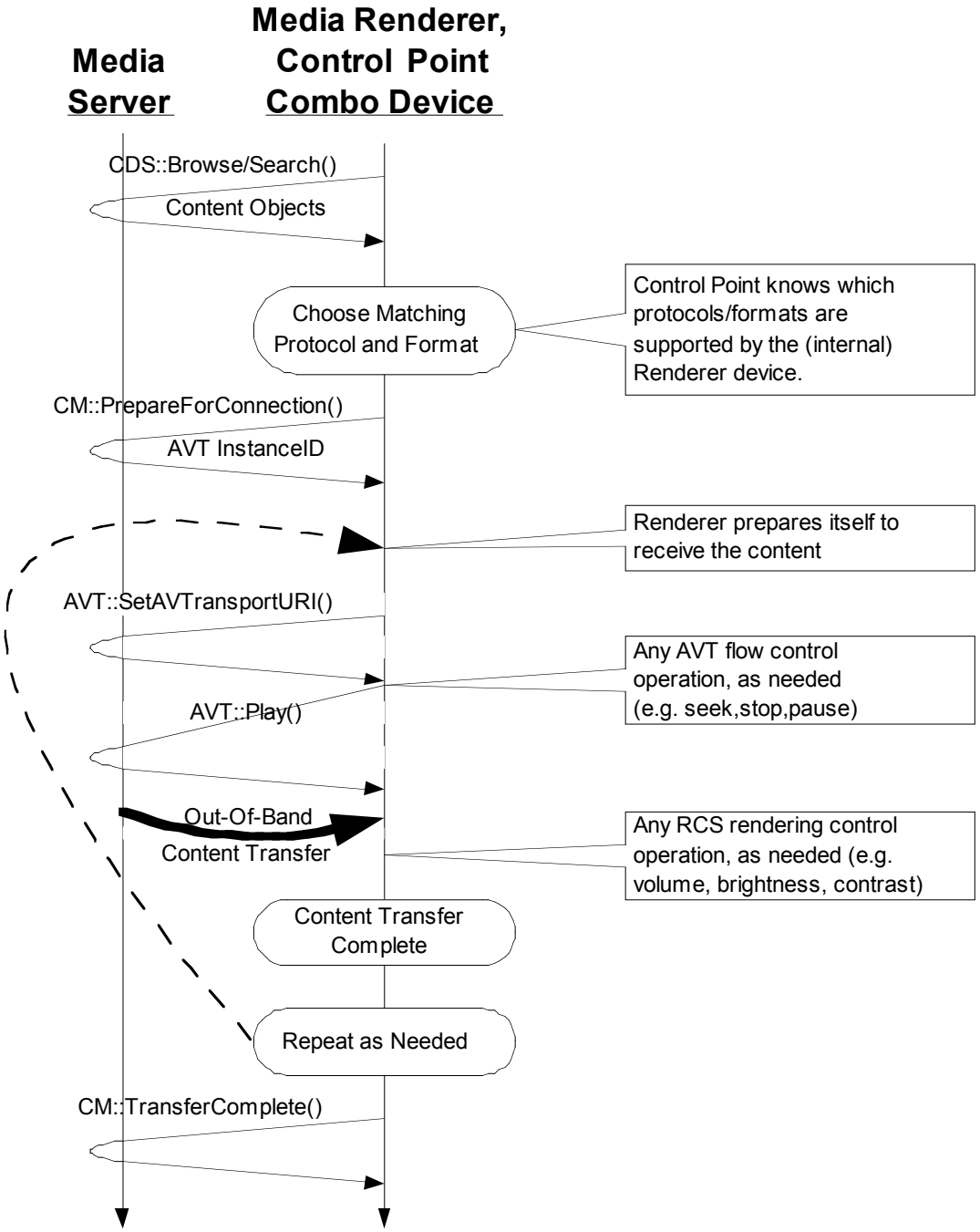
© 2000 Contributing Members of the UPnP Forum. All rights Reserved.



© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

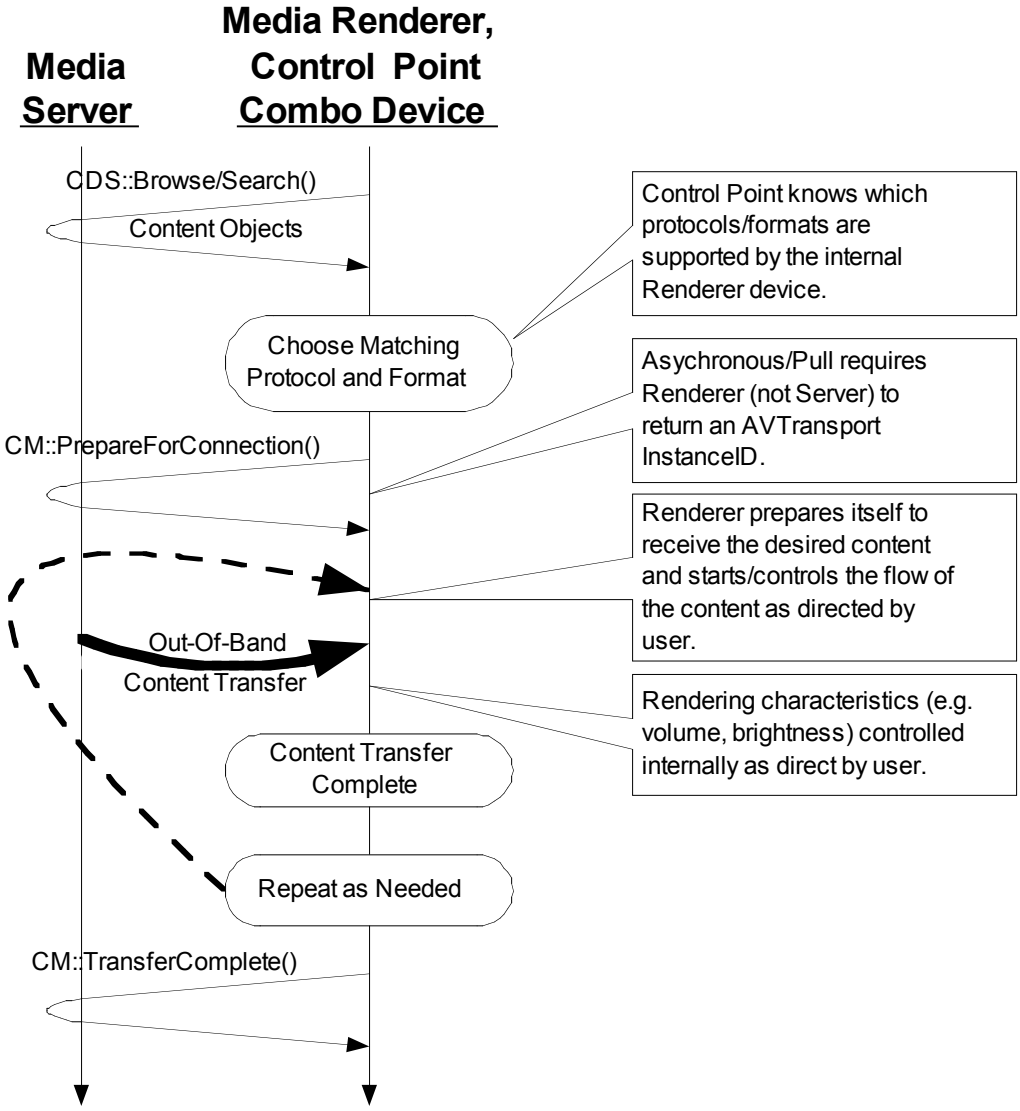
6.4. Renderer Combo Device using Isochronous-Push (e.g. IEEE-1394)

The following example illustrates how the general Device Interaction Algorithm is used to handle devices that also include integrated Control Point functionality (e.g. a TV).



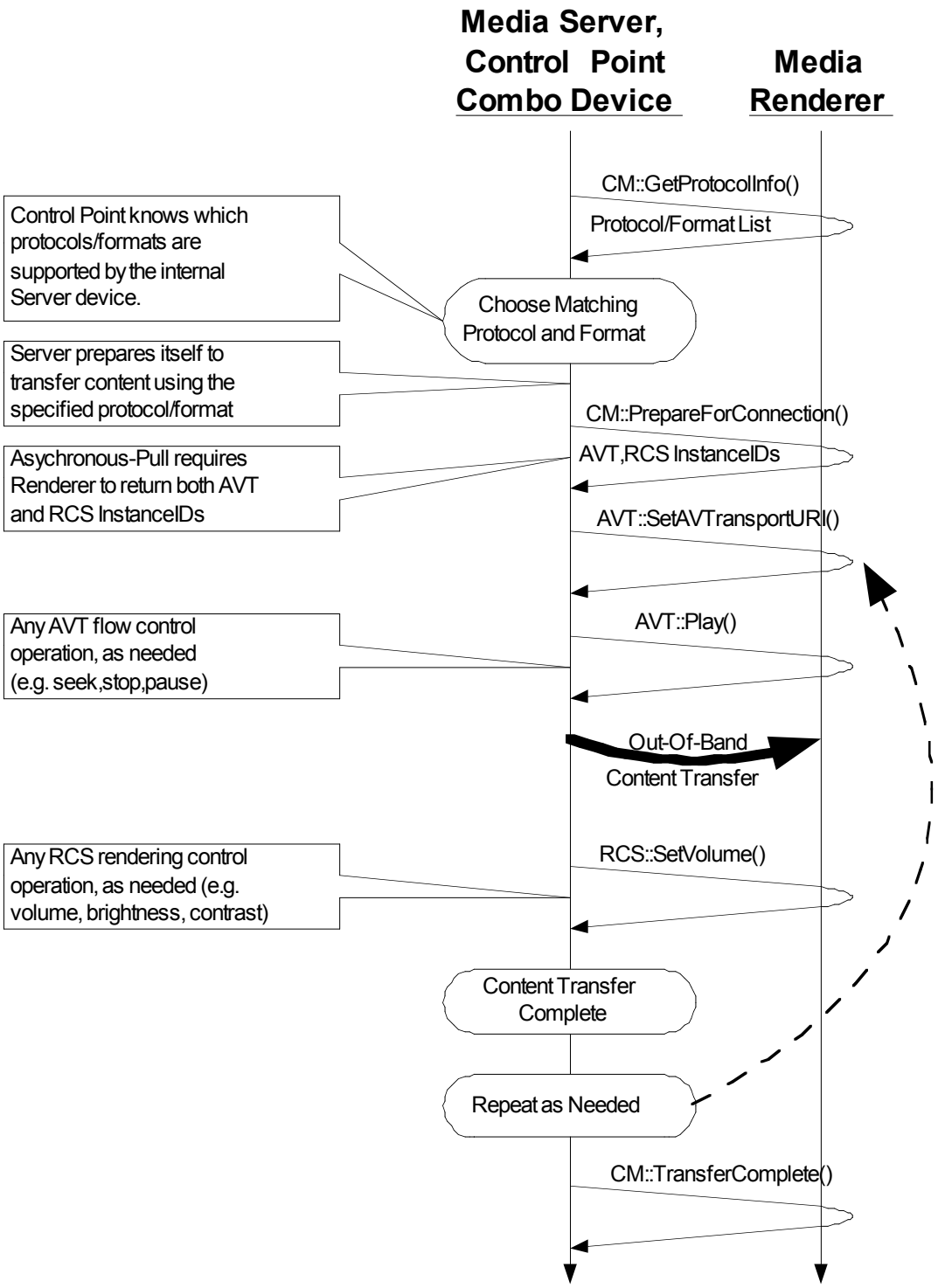
© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

6.5. Render Combo Device using Asynchronous-Push (e.g. HTTP GET)



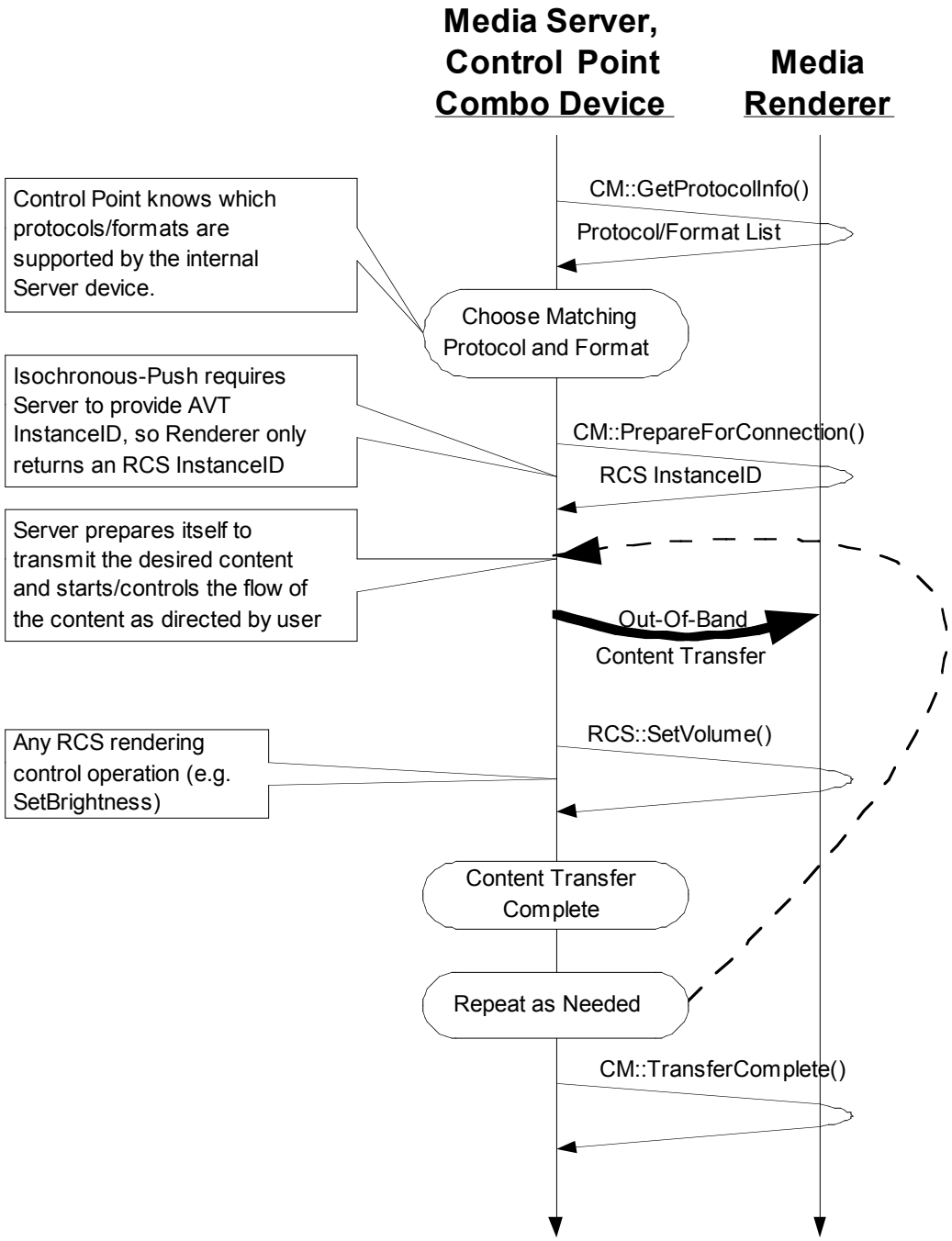
6.6. Server Combo Device using Asynchronous-Push (e.g. HTTP GET)

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.



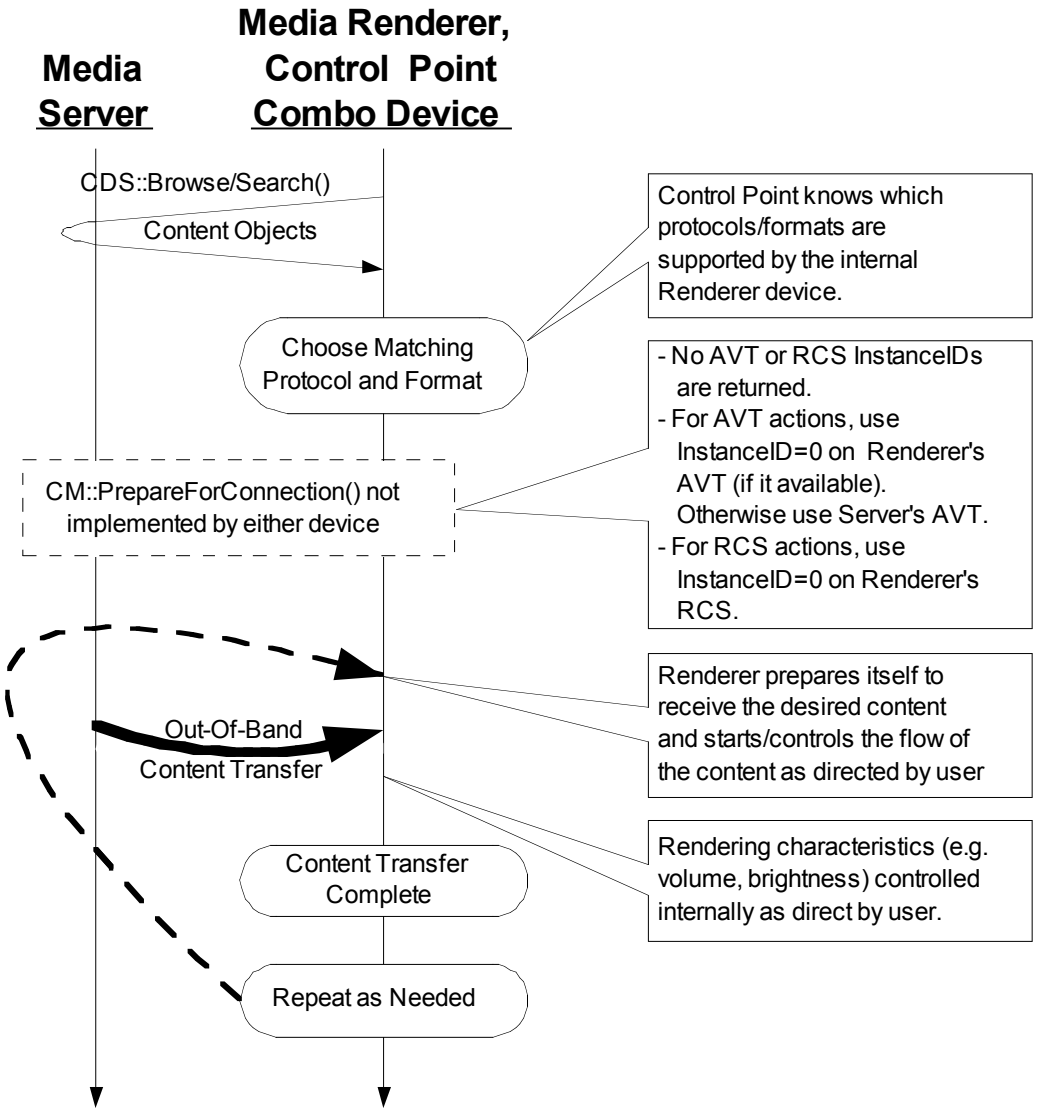
© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

6.7. Server Combo Device using Isochronous-Push (e.g. IEEE-1394)



© 2000 Contributing Members of the UPnP Forum. All rights Reserved.

6.8. Simplest Interaction Model Supported



7. Recording Architecture

The UPnP AV Architecture defines a rudimentary Recording capability. A Record action is defined within the AVTranport Service(). As content is being transferred from the MediaServer to the MediaRenderer, a Control Point may issue the 'Record' action. This results in the device 'recording' that content to some type of unspecified storage. The details of the Record feature depend completely on the recording device and can range dramatically from device to device.

© 2000 Contributing Members of the UPnP Forum. All rights Reserved.