



## Understanding the Basics

This chapter provides an overview of some fundamentals that are important to understand before you configure standalone Content Engines.

This chapter contains the following sections:

- [Some Key Terminology, page 2-1](#)
- [Overview of WCCP, page 2-5](#)
- [Understanding Some Basic ACNS Software Caching Concepts, page 2-7](#)
- [Understanding Some Basic ACNS Streaming Media Concepts, page 2-9](#)

## Some Key Terminology

[Table 2-1](#) describes some key terminology that is commonly encountered in caching and streaming.

**Table 2-1**      **Key Terminology**

Term	Definition
<b>Content-related terms</b>	
Content distribution	Method used to serve content in an ANCS 5.x network. With standalone Content Engines, two distribution methods are supported: pull distribution and preloading. See definitions of “preloading” and “pull distribution” in this table.
Content freshness	Freshness of cached content; that is, whether the cached content has been updated on the origin server since it was cached on the Content Engine. When a user requests on-demand content, the Content Engine checks if a cached object is “fresh” (if the content has been updated on the origin server since it cached the content locally) before serving the content to the client. If the content has not been modified since the Content Engine has cached the content, the Content Engine sends the client the cached content. If the content has been modified since it was cached, the Content Engine retrieves a “fresh” copy of the content from the origin server, caches the fresh content, and sends the fresh, cached content to the client.  For information about configuring cache freshness, see the <a href="#">“Configuring HTTP Cache Freshness Settings”</a> section on page 6-8.

Table 2-1 Key Terminology (continued)

Term	Definition
Preloading	Content distribution method in which standalone Content Engines retrieve and store specific content in anticipation of future requests for that content. See the <a href="#">“Configuring Content Preloading for Standalone Content Engines”</a> section on page 3-60. Content Engines that are registered with a Content Distribution Manager support pre-positioning as a content distribution method. Standalone Content Engines do not support pre-positioning.  For information about pre-positioning content, refer to the <i>Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2</i> .
Pull distribution	Content distribution method in which content is retrieved and cached on a Content Engine because of a user request (client-triggered demand). Subsequent requests for the same content are served from the local cache. This type of caching is referred to as “demand-pull caching.”
Request	Communication from a device (client) to retrieve content.
Web object	Object that can be transferred over the web (for example, web pages). Collective term used to refer to text objects and binary objects. A text object is an HTML page. A binary object is a web object other than a text object (for example, GIFs and JPEGs).
<b>Caching-related terms</b>	
Caching	Ability to store web objects for later retrieval.
Cache hit	Requested content (web object) that is in the Content Engine cache.
Cache miss	Requested content (web object) that is not in the Content Engine cache.
Live content	Content stream (typically streaming media) that is being broadcast from an origin server. For a more detailed definition, see <a href="#">Table 1-2</a> .
On-demand content	Content that is acquired, cached, and delivered because of a user request (client-triggered demand). On-demand content is acquired through pull distribution and cached locally on the Content Engine. Pull distribution method is used with the various caching services (for instance, reverse proxy caching, HTTP proxy caching, HTTP transparent proxy caching, RealMedia proxy caching, and RealMedia transparent caching). In contrast, content can be cached on a Content Engine as a result of a preload operation. For a more detailed definition of on-demand content, see <a href="#">Table 1-2</a> .
Preloaded content	Content that is retrieved and stored on a standalone Content Engine because you scheduled a retrieval of specific content (a preload operation) in anticipation of user requests for that content. For more information, see the <a href="#">“Configuring Content Preloading for Standalone Content Engines”</a> section on page 3-60.
<b>Device-related terms</b>	
Client	End user who is using a browser or media player to request web content. Also called “web clients.” See the <a href="#">“Web Clients Supported by Standalone Content Engines”</a> section on page 1-12.
Origin server	Server that contains the content that the client requested. This server acts as the master content repository. In the case of streaming media, the origin server receives live streaming from an encoder. These streams are then forwarded or “split” to Content Engines that act as edge devices that receive content from an origin streaming server. In the case of nonstreaming content, the Content Engine retrieves the requested content from the origin server if the content is not already stored in the local cache.

**Table 2-1** Key Terminology (continued)

Term	Definition
Standalone Content Engines	<p>Content Engines that are running ACNS 5.x software and are intentionally not registered with a Content Distribution Manager (if there is one) so that they can be managed as standalone devices. Standalone Content Engines can be configured or managed using the following: the Setup utility, the Content Engine CLI, or the Content Engine GUI. Multiple standalone Content Engines can be deployed (for example, you can deploy clusters of standalone Content Engines.)</p> <p>This guide describes how to configure, manage, and monitor standalone Content Engines as caching and streaming engines.</p> <p>For information about Content Engines that are registered with a Content Distribution Manager, refer to the <i>Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2</i>.</p>
<b>Deployment-related terms</b> Centrally managed deployments	<p>Deployments that consist of one or more of the following ACNS 5.x network devices: Content Distribution Managers, Content Engines that are registered with a Content Distribution Manager, and Content Routers that are running ACNS 5.x software.</p> <p>For information about configuring, maintaining, and monitoring such deployments, refer to the <i>Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2</i>.</p>
Locally managed deployments	<p>Deployments that consist of one or more standalone Content Engines that are running ACNS 5.x software and are configured as caching and streaming engines. These are the focus of this guide.</p>
PAC file	<p>File (proxy automatic configuration) that is used to point client browsers to a specific Content Engine as their proxy server. This file resides on a server in your intranet, and can be loaded at startup by a browser on a client desktop. A PAC file is written in JavaScript. Use a PAC file if you want to use direct proxy routing to direct request from certain clients to a specific Content Engine. See the <a href="#">“Using PAC Files to Point Client Browsers Directly to a Standalone Content Engine”</a> section on page 3-37.</p>
<b>Routing-related terms</b> Direct proxy routing	<p>Type of routing in which client browsers or media players are configured to explicitly point to a specific forward proxy server (the Content Engine). Because the web clients are aware that their content requests are being directed to the Content Engine (forward proxy server), the clients can point to this Content Engine for nontransparent caching services (for example, client acceleration by quickly accessing cached content on this Content Engine). See <a href="#">Table 1-5</a> for a list of supported caching and streaming services available when direct proxy routing is used.</p> <p>For information about configuring clients to use this routing method, see the <a href="#">“Configuring Client Browsers and Media Players for Direct Proxy Routing”</a> section on page 3-36.</p>
Transparent redirection	<p>Type of routing in which Web Cache Communication Protocol (WCCP) router or Layer 4 switch transparently intercepts client requests and redirects the request to the Content Engine. The web clients are not aware that a WCCP router or a Layer 4 switch is redirecting their requests to the Content Engine (the transparent proxy server). See <a href="#">Table 1-6</a> for a list of supported caching and streaming services when transparent redirection is used.</p> <p>For information about configuring this routing method, see <a href="#">Chapter 5, “Configuring Transparent Redirection for Standalone Content Engines.”</a></p>

Table 2-1 Key Terminology (continued)

Term	Definition
<b>Proxy server-related terms</b>	
Forward proxy server	Content Engine that receives content requests directly from the web clients because direct proxy routing is being used to direct client requests to the Content Engine. See the <a href="#">“Overview of Nontransparent Forward Proxy Caching”</a> section on page 4-3.
Reverse proxy server	Content Engine that is running the reverse proxy service for web server acceleration. Acts as a proxy for a small number of web servers (web server farms) by locally caching the requested content from these specific web servers. See the <a href="#">“Overview of Transparent Reverse Proxy Caching”</a> section on page 4-10.
Transparent proxy server	Content Engine that is used for client acceleration. Acts as a proxy for web clients by caching requested content whenever possible in its local cache. Used if transparent redirection (WCCP routing or Layer 4 switching) is the routing method used to direct client requests to the Content Engine. The web clients are not aware that their requests are being intercepted by a WCCP router or a Layer 4 switch and being redirected to the Content Engine (the transparent proxy server). See the <a href="#">“Overview of Transparent Forward Proxy Caching”</a> section on page 4-6.
<b>Service-related terms</b>	
Proxy cache services	Caching services that use a Content Engine as a proxy server for the following types of acceleration: <ul style="list-style-type: none"> <li>Client acceleration for HTTP, HTTPS, FTP, WMT, and RTSP requests (through a Content Engine that is functioning as a forward proxy server or transparent proxy server)</li> <li>Web server acceleration for HTTP or FTP-over-HTTP requests (through a Content Engine that is functioning as a reverse proxy server)</li> </ul>
RTSP streaming and caching services	Supported RTSP streaming and caching services (for example, RealMedia proxy caching, RealMedia transparent caching, and RealProxy live splitting). For more information, see <a href="#">Chapter 8, “Configuring RTSP Media Services on Standalone Content Engines.”</a>
Sanitized logging	Logging of web resource requests that does not include (or deliberately hides) the identity of the client. For more information, see the <a href="#">“Sanitizing Transaction Logs”</a> section on page 20-34.
Transaction	Completed request (whether successful or failed) for a web resource by a client. Information about transactions is recorded in the ACNS software transaction logs. For more information, see the <a href="#">“Monitoring Transactions with Standalone Content Engines”</a> section on page 20-18.
WCCP	Cisco Web Cache Communication Protocol (WCCP). Protocol used by routers and Content Engines to support transparent redirection of client requests for content to Content Engines that are functioning as transparent proxy servers. For more information, see the <a href="#">“Overview of WCCP”</a> section on page 2-5.
WMT streaming and caching services	Supported WMT streaming and caching services (for example WMT proxy caching, WMT transparent caching, and WMT live splitting). For more information, see <a href="#">Chapter 7, “Configuring WMT Streaming Media Services on Standalone Content Engines.”</a>

Table 2-1 Key Terminology (continued)

Term	Definition
<b>Streaming-related terms</b>	
RealMedia (RM) format	Format of RealNetworks streaming media files. Standalone Content Engines can serve RealMedia content (.rm files) to RealMedia players if RealProxy is enabled on the Content Engine.  For more information, see <a href="#">Chapter 8, “Configuring RTSP Media Services on Standalone Content Engines.”</a>
SMIL	Synchronized Multimedia Integration Language. An open standard markup language that is similar to HTML. This simple but powerful markup language allows you to coordinate multiple clips. SMIL also allows you to define how, when, and where you want the multiple clips to be played. The SMIL file establishes the overall time line of the presentation.
RealMedia players	RealNetworks media players such as RealPlayer and RealOne player. If a standalone Content Engine receives a request from a RealMedia player (either through direct proxy routing or transparent interception), the Content Engine uses the RealNetworks RTSP protocol to serve the requested RealMedia content to the RealMedia player. If the requested RealMedia content is not already stored in its local cache, the Content Engine retrieves the requested content from the origin streaming server (RealNetworks server).  For more information, see <a href="#">Chapter 8, “Configuring RTSP Media Services on Standalone Content Engines.”</a>
Stream splitting	Process in which a single live stream from the origin server is split, or copied, into multiple streams by a splitter to serve to clients that requested the stream. This process is especially common when the stream has to traverse a mixture of multicast and non-multicast-enabled networks. The splitter can receive incoming streams by unicast or multicast, and can serve streams by unicast or multicast. Stream splitting can be used for live streams or scheduled rebroadcasts.  For information about stream splitting with WMT, see the <a href="#">“Configuring Standalone Content Engines to Deliver WMT Live Streams”</a> section on page 7-29.

## Overview of WCCP

Cisco developed the Web Cache Communication Protocol (WCCP) within Cisco IOS software to enable routers or switches to transparently redirect packets to network caches. In environments with standalone Content Engines, WCCP acts as a communication mechanism between a router and the standalone Content Engines.



### Note

WCCP is only licensed on the Content Engine (caching device) and not on the redirecting router. WCCP does not interfere with normal router or switch operations. The router must be running a version of Cisco IOS software that supports WCCP Version 1 or Version 2.

When caching support is enabled on the router and WCCP support is enabled on the Content Engines, the devices can communicate and deliver the WCCP features and services for which they are configured. To use a WCCP-enabled router, an IP address must be configured on the interface connected to the Internet, and the interface must be visible to the Content Engine on the network.

There are two versions of WCCP, Version 1 and Version 2.

The following is a summary of the capabilities and limitations of WCCP Version 1:

- Only a single WCCP-enabled router (home router) is supported.
- Only a single WCCP service (the web-cache service [service 0]) is supported.
- Traffic redirection is only supported on port 80.
- Each home router can support the standard web-cache service (service 0) for up to 32 Content Engines.
- There is no bypass support (for example, static bypass, error bypass, and authentication bypass are not supported).
- There is no support for generic routing encapsulation (GRE) on return.

WCCP Version 2 supports a broad set of services and features (including bypass support) and more than one WCCP-enabled router. (See [Table 5-2](#) for a list of WCCP Version 2 features and services.) Consequently, we recommend that you use WCCP Version 2.

The following sequence of events details the interaction between Content Engines and routers that have been configured to run WCCP Version 2:

1. Each Content Engine is configured with a router list. (See the [“Defining Router Lists on Standalone Content Engines”](#) section on page 5-10.)
2. Each Content Engine announces its presence and a list of all routers with which it has established communications. The routers reply with their view (list) of Content Engines in the group.

Routers and Content Engines become aware of one another and form a WCCP service group using a management protocol. The Content Engines also send periodic “Here I am” messages to the routers that allow the routers to rediscover the Content Engines. To properly depict the view, the protocol needs to include the list of routers in the service group as part of its messages.

3. Once the view is consistent across all the Content Engines in the Content Engine cluster, one Content Engine is designated the lead. When there is a cluster of Content Engines, the one seen by all routers and the one that has the lowest IP address becomes the lead Content Engine.

The role of this lead Content Engine is to determine how traffic should be allocated across the Content Engines in the Content Engine cluster. The lead Content Engine sets the policy that the WCCP-enabled routers must adhere to when redirecting packets to the Content Engines in this cluster. The assignment information is passed to the entire service group from the designated Content Engine so that the routers in the service group can redirect the packets properly and the Content Engines in the service group can better manage their load.

When you use the Setup utility to configure HTTP, WMT, or RealMedia transparent caching through WCCP redirection, WCCP Version 2 is used. In order to use WCCP Version 1 for HTTP transparent caching (the web-cache service), you must configure this WCCP service through the Content Engine CLI. (See the [“Scenario 1—Configuring the Web-Cache Service with WCCP Version 1”](#) section on page 5-36.) For information about configuring transparent redirection with WCCP, see [Chapter 5](#), [“Configuring Transparent Redirection for Standalone Content Engines.”](#)

## Overview of WCCP Services

Some of the WCCP services that routers can supply have a well-known set of criteria and a predefined service identifier (for example, the web-cache service has zero “0” as its identifier [service 0]). Other examples of such services include the reverse-proxy service (service 99), the https-cache service (service 70), and the rtsp service (service 80).

Other services that are not well known can be supported by defining a dynamic (user-defined) WCCP service (services 90 to 97) using WCCP Version 2. As part of defining a dynamic WCCP service, you specify a set of criteria (for example, the application type [HTTP caching, HTTPS caching, or streaming], hash parameters, password, and weight), and assign this service a service identifier (service 90 to 97) on the Content Engine. Although you can specify the criteria for a user-defined service on the Content Engine, the WCCP Version 2-enabled routers must support the particular service identifier (service 90 to 97). For more information about configuring user-defined WCCP Version 2 services, see the [“Configuring Standalone Content Engines to Support User-Defined WCCP Services” section on page 5-13](#).

WCCP Version 1 only supports one WCCP service (the web-cache service) and a single router (home router). Only WCCP Version 2 supports user-defined WCCP services (services 90 to 97). Consequently, we recommend that you use WCCP Version 2 because it supports a wider set of features and services (see [Table 5-2](#)) as well as multiple routers.

See [Table B-3](#) for a list of supported WCCP services. For information about using the Content Engine CLI to configure WCCP services for standalone Content Engines, see [Chapter 5, “Configuring Transparent Redirection for Standalone Content Engines.”](#)

## Understanding Some Basic ACNS Software Caching Concepts

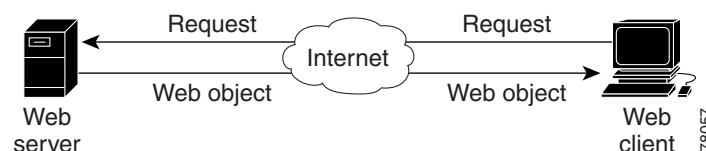
This section describes the following fundamentals about ACNS software caching:

- [Introduction to Caching](#)
- [Cacheable Content](#)
- [Network Protocols and Caching](#)

### Introduction to Caching

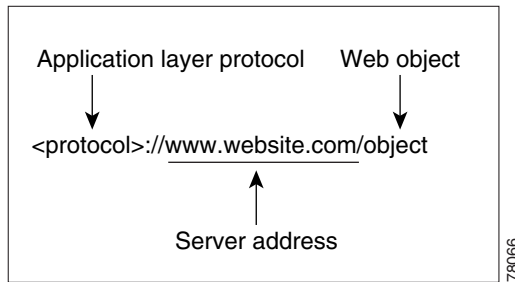
Caching refers to the ability to store web objects such as web pages for later retrieval. Typically a web client requests an object from a web server using a web browser. (See [Figure 2-1](#).)

**Figure 2-1 Basic Web Request**



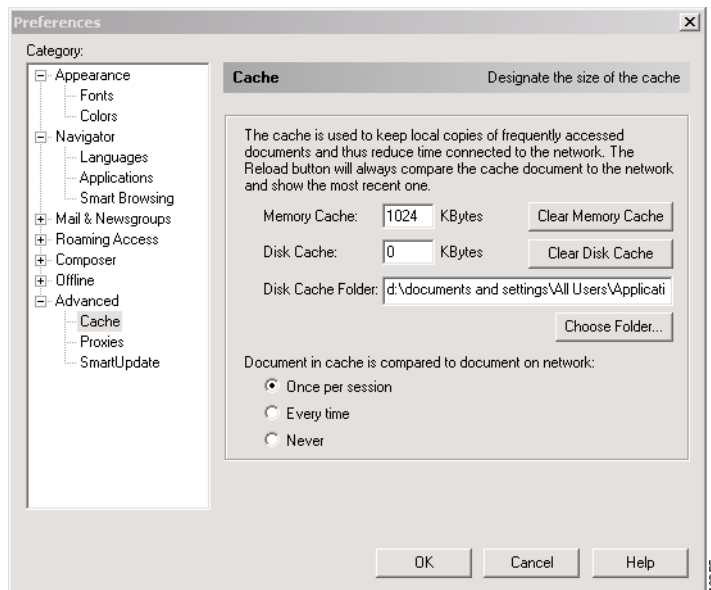
Every web object has a uniquely defined address (URL) that allows the web browser to retrieve web objects, assuming that the server address exists in the DNS space. The basic components of a URL address are shown in [Figure 2-2](#).

**Figure 2-2 Basic Components of URL Address**



To implement caching using web browsers, browsers have a local cache that the user can set up to help retain recently viewed pages for easier access during the web viewing process. (See [Figure 2-3](#).) For instance, clicking the **Back** button on a browser takes advantage of local caching, because this recently viewed page is rapidly displayed.

**Figure 2-3 Local Browser Cache Configuration**



Despite the inherent advantages obtained using a local cache, a Content Engine is often used to provide specialized caching features to many users.

The ability of the Content Engine to store information on a large scale for later retrieval has significant advantages for the user and Internet traffic as a whole:

1. It reduces network congestion by keeping web objects close to the user instead of accessing the same content through the network.
2. It reduces the time it takes to display a web page, because the page is stored locally or close to the user.
3. It reduces the load on the origin web server, because this server does not have to redistribute content that has recently been acquired.

In this scenario, a user attempts to retrieve a web object from a web server. Because the local browser does not have the page stored in its cache, the browser sends the request to the origin web server using the Content Engine as its proxy for the web request. In this deployment, the Content Engine serves as a proxy to the web client, and it first tries to satisfy the web request from its cache of stored objects.

## Cacheable Content

Cacheable content is typically static application data and can be associated with a file type and file extension, as described in [Table 2-2](#).

**Table 2-2** File Type and Extensions of Cacheable Content

File Type	File Extension
Graphic files	gif, jpeg, bmp
Compressed files	zip, gz, tar
Document files	text, txt, pdf
Multimedia files	avi, mov, mpeg, wav

## Network Protocols and Caching

The interaction between a web browser and a web server uses the existing standard application-layer Internet protocols such as HTTP, Microsoft Media Server (MMS) protocol, and RTSP. The Content Engine has to be able to serve web objects to the web client using all of these web access protocols. See [Table B-1](#) for a descriptive list of the network protocols that a Content Engine, which is running the ACNS software (Release 5.2 or later), can use to serve content to web clients.

## Understanding Some Basic ACNS Streaming Media Concepts

This section describes how streaming media services are supported on a Content Engine in a locally managed deployment. It discusses the following topics:

- [Understanding Some Basic ACNS Streaming Media Concepts](#)
- [Understanding IP Multicasting Fundamentals](#)

For a list of supported streaming media protocols, see [Table B-2](#). For information about how to configure media services on standalone Content Engines, see:

- [Chapter 7, “Configuring WMT Streaming Media Services on Standalone Content Engines”](#)
- [Chapter 8, “Configuring RTSP Media Services on Standalone Content Engines”](#)

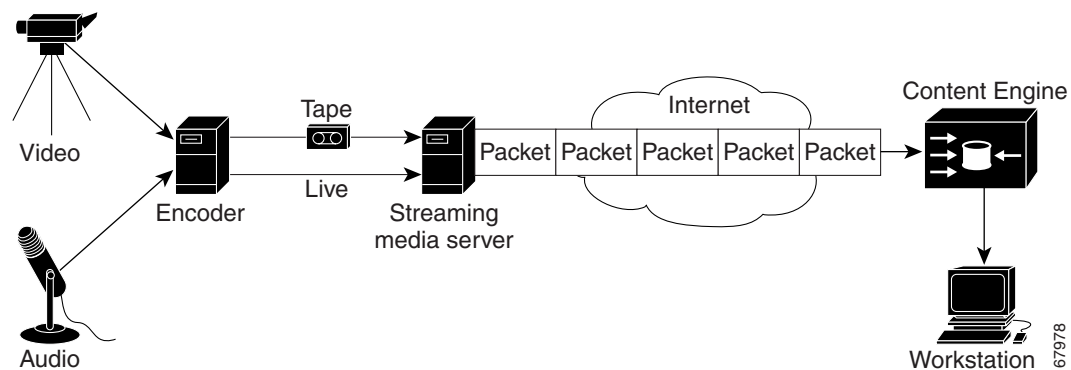
For information about how to configure media services on Content Engines that are registered with a Content Distribution Manager, refer to the *Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2*.

## Understanding Streaming Media Solutions

Streaming is a technology that allows content to be accessed or viewed before all the media packets have been received, whereas with caching, the content must be received in its entirety before it can be accessed. Streaming media can be delivered as live content or on-demand content, such as video on demand (VOD).

Figure 2-4 shows streaming media files, which are files that are sent to the user and played on the user's media player as the files are received from the network. Streaming media files avoid a waiting period for viewing these files because they are immediately available as a continuous stream of data packets. This eliminates the need to store large media files for viewing purposes or the need to allocate storage space for these files before playing them.

**Figure 2-4 Streaming Media Model with Content Engine in Direct Proxy Mode**



In Figure 2-4, both audio and video are being recorded from an event to be distributed later either as a VOD or live to a network of users. The encoder software and hardware compress the signal into streamable files, which are then sent to a media file server. The streaming media server in turn delivers the media files on a live or on-demand basis to the Content Engine, which then relays the requested content to users who are running a particular media software (for example, the Windows Media player) on their personal computers or other electronic devices.

Cisco ACNS software supports several types of streaming media solutions, including solutions from Microsoft, RealNetworks, and Apple Computer. See Table 1-3 for a list of supported streaming media solutions.

Client requests for media files can be directed to the Content Engine through the following routing methods:

- With direct proxy routing, the client media players are configured to point directly to the Content Engine.
- With transparent redirection, a WCCP router or Layer 4 switch transparently intercepts a client's request for media files and transparently redirects the request to the Content Engine.

In both types of routing, the Content Engine always obtains the live content from the external streaming media server (Encoder > streaming media server > Content Engine) or encoder (Encoder > Content Engine); the Content Engine is not the originator of the live content.

Whereas browsers mainly use HTTP to fetch contents, streaming media players mainly use various streaming protocols to fetch and display contents. For example, RealMedia players use the RealNetworks RTSP protocol to stream audio and video contents, the Windows Media player uses the Microsoft MMS protocol to stream audio and video contents, and QuickTime players use the IETF standard RTSP protocol to stream multimedia contents. The media player on the client desktop can be started directly by the end user, or it can be automatically launched by the client browser. See [Table 1-4](#) for a list of supported web clients.

## How Client Browsers Spawn Media Players

It is important to understand the ways in which media players are launched, because media-index files and relative links impose restrictions on the content distribution software, typically requiring that the directory structure be maintained on the Content Engine.

The two primary ways for the client browser to start a streaming media player on a client desktop are:

- Through Java script embedded in the web page.
- For the browser to start a streaming media player through a MIME-type association. An MIME-type association is a browser capability that enables the browser to invoke a particular application when it encounters an object with a particular MIME-type suffix. A set of default association rules covers the common object types on the Internet. Users can edit, add, or delete these MIME-type association rules in their browsers. For example, through MIME-type association, the client browser launches the Adobe Acrobat reader when it encounters a \*.pdf file, and it launches the Windows Media Player when it encounters an \*.asf or \*.asx file.

Client browsers can launch media players on a client desktop in a variety of situations. For example, if the HTML page contains a link to a video that is encoded in ASF format (see below), while processing this HTML page, the client browser launches the Windows Media player on the client desktop in order to retrieve the file wmt\_vod/welcome1.asf from the server named stream-ce50.abc.com. The MMS protocol is used to retrieve the .asf file for the Windows Media player.

```
HTML>
HEAD>
<TITLE>Example Page with a Video/TITLE>
</HEAD>
<BODY>
<A HREF="mms://stream-ce50.abc.com/wmt_vod/welcome1.asf">TEST CLIP</A>
</BODY>
</HTML>
```

A more common example of a how a client browser launches the media player on the client desktop is when the video and presentation material is packaged in a media-index file such as an .asx file or a Synchronized Multimedia Integration Language (SMIL) file. The .asx file is used by WMT streaming, and the SMIL file is used by the RealNetworks RealMedia streaming. The browser is typically configured so that the moment it retrieves an .asx file, it spawns the Windows Media Player and passes the .asx file to the player. Similarly, when the client browser retrieves an SMIL file, it spawns the RealMedia player on the client desktop and passes the SMIL file to the RealMedia player on the client desktop.

**Note**

In ACNS 5.2 software, support for SMIL files for RealProxy was added.

Media-index files can contain either relative links to media files or absolute links to media files. For example, the following is an .asx file that contains absolute links to .asf files:

```
<!--A simple playlist with entries to be played in sequence.-->
<ASX version = "3.0">
  <Title>The Show Title</Title>
  <Entry>
    <Ref href = "mms://adventure-works.com/Path1/title1.wma" />
  </Entry>
  <Entry>
    <Ref href = "mms://abc-studio.com/Path2/title2.wma" />
  </Entry>
  <Entry>
    <Ref href = "mms://fox-networks.com/Path3/title3.wma" />
  </Entry>
</ASX>
```

The .asx file instructs the player to retrieve three videos in the specified sequence. Note that the three videos are at different websites. The full URL for the video is listed in the .asx file.


**Note**


---

RealMedia files have an .rm extension, which indicates that the file is in a RealMedia format.

---

Note that in the case of a relative link, the full URL is constructed by replacing the last path component in the SMIL file's URL path with the relative link.

Most media players support synchronized media, that is, audio and video that are synchronized with a slide presentation. The media are typically put in the same directory, and are referenced using relative links.

## How a Client Media Player Issues a Request

Once the media player has been launched, it is given a media file to play and the player follows procedures similar to those of the browser.

1. The media player looks up the IP address of the streaming server through the DNS resolution server that is configured on the client desktop. If a proxy is configured for the media player, the media player does not perform this DNS lookup; the player requests that the proxy perform the look up for the IP address. (Just as you can configure a client browser to point directly to a Content Engine, you can also configure client media players to point directly to a Content Engine. For more information, see:
  - [Pointing Windows Media Players Directly to a Standalone Content Engine](#),
  - [Pointing RealMedia Players Directly to a Standalone Content Engine](#).)
2. The media player tries to establish a connection to the media server on the default port (port 1755 for WMT, and port 554 for RTSP) or a specified port. In the case of a proxy, it will establish a connection to the proxy on the specified port.

3. Once the connection is established between the media player and the media server or the proxy, the media player exchanges information with the media server or the proxy to establish the sessions.

Typically, the first packet that the media player sends to the server or the proxy contains the IP address of the intended media server or the domain of the server. This is different from HTTP, where the first packet contains not only the domain information but also the object information. In the case of both WMT and RTSP streaming protocols, the first packet carries server information only. The object information (pathname) is not available until the second or the third packet exchange between the client media player and the server. When the WMT protocol is used, the object information is obtained in the fourth packet exchange. When RTSP-over-RTP is used, the object information is obtained in the third packet exchange.

### RTSP Redirect Method

Similar to HTTP, the RTSP protocol also has a “redirect” message. The server can send a redirect message to the media player with a new URL, and the player plays the new URL.

### WMT Redirect Method

The current version of the WMT streaming protocol does not provide a “redirect method.” Instead, a more commonly used method is for the server to use the .asx file to publish videos, and, if redirection is needed, for the server to rewrite the .asx file (changing the references in the file). As explained previously (“[How Client Browsers Spawn Media Players](#)”) the .asx file directs the media player to retrieve various content objects, so rewriting the references in the file effectively redirects the media player to different URLs.

## Understanding Caching Policies in Streaming Media Caching

In contrast to HTTP caching, caching policies in streaming media caching are much simpler, because streaming media are mostly large static content.

With WMT caching, you use the Content Engine GUI or CLI (for example, the **wmt cache max-obj-size** global configuration command) to specify the caching policies for a standalone Content Engine. All responses are cacheable, including partial responses. All WMT requests result in communication between the Content Engine and the origin server, whether the Content Engine establishes the streaming control session with the server for the client, even if the request is a cache hit. By establishing the streaming control session, the Content Engine can verify that its cached content is fresh, and the client can access the content. Because streaming objects are typically very large, the overhead of establishing the control session with the server is negligible and does not reduce the bandwidth savings from the cache hits.

With RealProxy caching (RealMedia proxy caching and RealMedia transparent caching), you specify the RealProxy caching policies through the RealNetworks RealSystem Administrator GUI. For more information on this topic, see the “[Configuring RealProxy with the RealSystem Administrator GUI](#)” section on page 8-20.

## Understanding IP Multicasting Fundamentals

Multicast delivery enables the distribution of streaming media by allowing different receiving devices on the IP multicast to receive a single stream of media content from the Content Engine simultaneously. This can save significant network bandwidth consumption, because a single stream is sent to many devices, rather than sending a single stream to a single device every time that this stream is requested.

This multicast delivery feature is enabled by setting up a multicast address on the Content Engine to which different devices, configured to receive content from the same channel, can subscribe. The delivering device sends content to the multicast address set up at the Content Engine, from which it becomes available to all subscribed receiving devices.

To take advantage of multicasting, all devices (including Content Engines, routers, and clients) must be multicast-enabled. For this reason, multicasting is mostly used in local networks where routers can be configured for multicasting. Multicast delivery over the Internet can only be accomplished when all the devices that participate in the multicast have been enabled for multicasting.

The Internet Assigned Numbers Authority (IANA) controls the assignment of IP multicast addresses. The IANA has assigned the IPv4 Class D address space to be used for IP multicast. Therefore, all IP multicast group addresses fall in the range from 224.0.0.0 through 239.255.255.255. However, some combinations of source and group address should not be routed for multicasting purposes. See the [“Unusable Multicast Address Assignments”](#) section on page B-12 for a list of the unusable multicast address ranges and the reasons they should not be used.

### Insecure Services

Applications that use multicast addresses in the 224.0.1.2/32, 224.0.1.22/32, and 224.0.2.2/32 ranges have been demonstrated to be vulnerable to exploitation, which has led to serious security problems.

### Copying Files Between Servers and Clients

Some applications are used to copy files between servers and clients and to otherwise maintain groups of personal computers. These applications are intended to be used on a local subnet or within an administrative domain, but the default addresses used by the software are not within the administrative addresses used by the administratively scoped addresses listed in [Table B-8 on page B-13](#).

### Limited Scope Addresses

Limited scope addresses are also called administratively scoped addresses. These addresses are described in RFC 2365, *Administratively Scoped IP Multicast*, to be restricted to a local group or organization. Companies, universities, or other organizations can use limited scope addresses to have local multicast applications that will not be forwarded outside their domain. Routers typically are configured with filters to prevent multicast traffic in this address range from flowing outside an autonomous system (AS) or any user-defined domain. Within an autonomous system or domain, the limited scope address range can be further subdivided so that local multicast boundaries can be defined. This subdivision is called address scoping and allows for address reuse between these smaller domains.

### Source-Specific Multicast Addresses

Addresses in the 232.0.0.0/8 range are reserved for source-specific multicast (SSM). SSM is an extension of the Protocol-Independent Multicast (PIM) protocol that allows for an efficient data delivery mechanism in one-to-many communications.

## GLOP Addresses

RFC 2770, *GLOP Addressing in 233/8*, proposes that the 233.0.0.0/8 address range be reserved for statically defined addresses by organizations that already have an AS number reserved. This practice is called GLOP addressing. The AS number of the domain is embedded into the second and third octets of the 233.0.0.0/8 address range. For example, the AS number 62010 is written in hexadecimal format as F23A. Separating the two octets F2 and 3A results in 242 and 58 in decimal format. These values result in a subnet of 233.242.58.0/24 that would be globally reserved for AS 62010 to use.

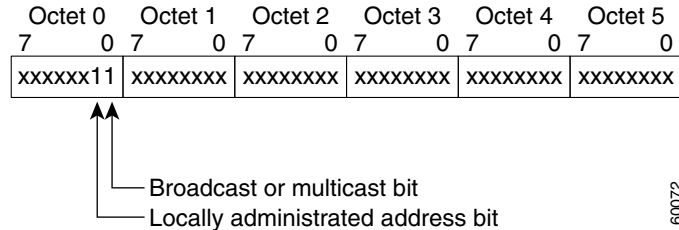
## Layer 2 Multicast Addresses

Historically, network interface cards (NICs) on a LAN segment could receive only packets destined for their burned-in MAC address or the broadcast MAC address. In IP multicast, several hosts need to be able to receive a single data stream with a common destination MAC address. Some means had to be devised so that multiple hosts could receive the same packet and still be able to differentiate between several multicast groups.

One method to accomplish this is to map IP multicast Class D addresses directly to a MAC address. Today, using this method, NICs can receive packets destined to many different MAC addresses—their own unicast, broadcast, and a range of multicast addresses.

The IEEE LAN specifications made provisions for the transmission of broadcast and multicast packets. In the 802.3 standard, bit 0 of the first octet is used to indicate a broadcast or multicast frame. [Figure 2-5](#) shows the location of the broadcast or multicast bit in an Ethernet frame.

**Figure 2-5 IEEE 802.3 MAC Address Format**



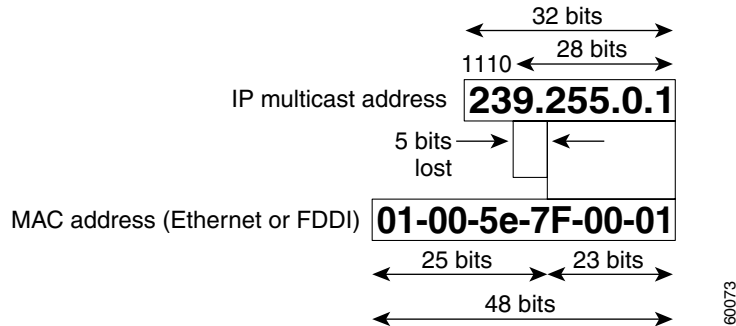
This bit indicates that the frame is destined for a group of hosts or all hosts on the network (in the case of the broadcast address 0xFFFF.FFFF.FFFF).

IP multicast makes use of this capability by sending IP packets to a group of hosts on a LAN segment.

### Ethernet MAC Address Mapping

The IANA owns a block of Ethernet MAC addresses that start with 01:00:5E in hexadecimal format. Half of this block is allocated for multicast addresses. The range from 0100.5e00.0000 through 0100.5e7f.ffff is the available range of Ethernet MAC addresses for IP multicast.

This allocation allows for 23 bits in the Ethernet address to correspond to the IP multicast group address. The mapping places the lower 23 bits of the IP multicast group address into these available 23 bits in the Ethernet address. (See [Figure 2-6](#).)

**Figure 2-6** IP Multicast to Ethernet or FDDI MAC Address Mapping

Because the upper five bits of the IP multicast address are dropped in this mapping, the resulting address is not unique. In fact, 32 different multicast group IDs map to the same Ethernet address. (See [Figure 2-7](#).) Network administrators should consider this fact when assigning IP multicast addresses. For example, 224.1.1.1 and 225.1.1.1 map to the same multicast MAC address on a Layer 2 switch. If one user subscribed to Group A (as designated by 224.1.1.1) and the other users subscribed to Group B (as designated by 225.1.1.1), they would both receive both A and B streams. This situation limits the effectiveness of this multicast deployment.

**Figure 2-7** MAC Address Ambiguities