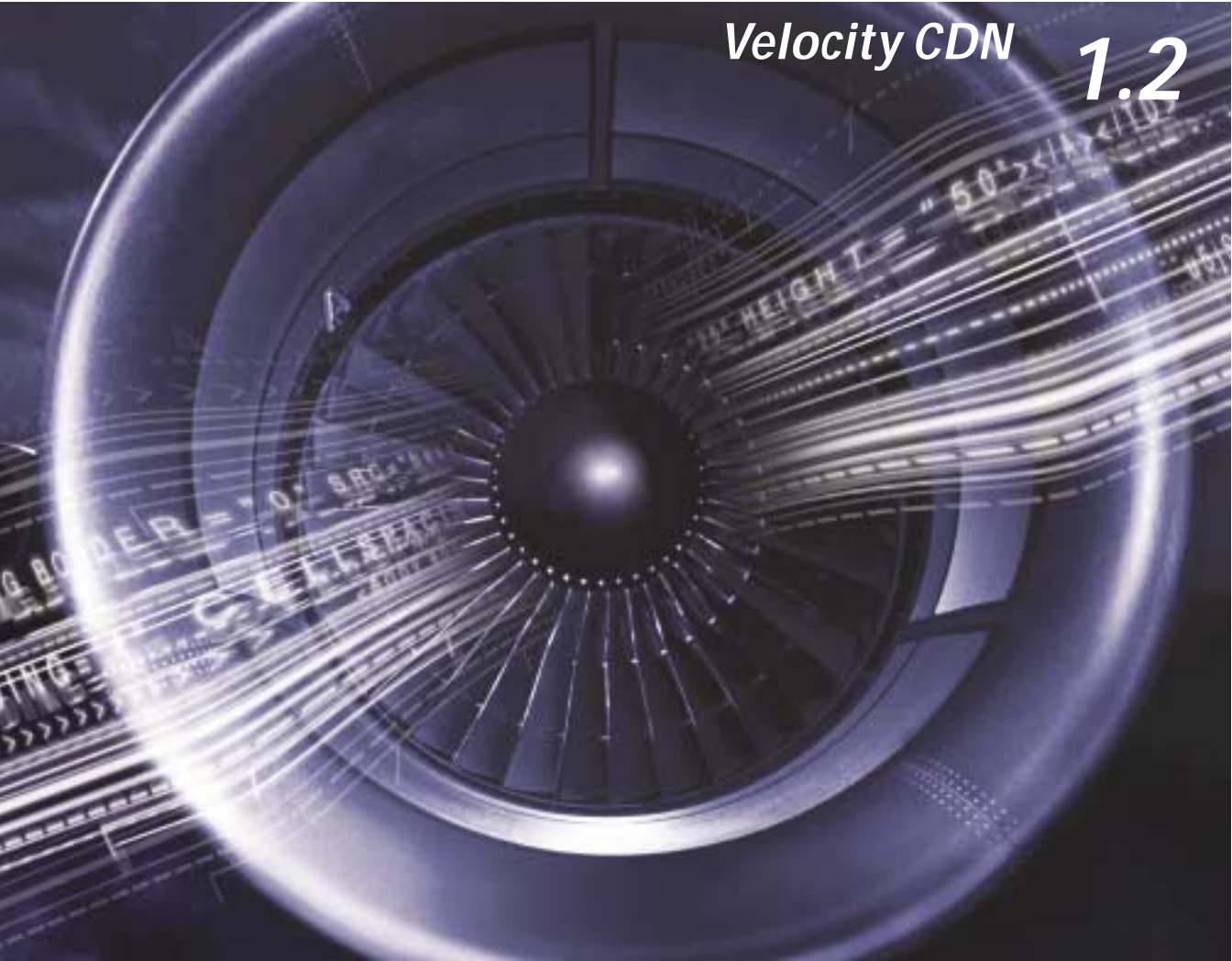




Velocity CDN **1.2**



***Content Controller
Deployment Guide***

Legal Notices

Volera, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Volera, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Volera, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Volera, Inc. reserves the right to make changes to any and all parts of Volera software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Copyright © 1997-2002 Volera, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

U.S. Patent Nos. 5,870,739; 5,873,079; 5,884,304; 6,330,605. U.S. and Foreign Patents Pending.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

Volera, Inc.
2211 North First Street
San Jose, CA 95131-2021
U.S.A.

www.volera.com

Managing Content with Content Controller
April 2002

Online Documentation: To access the online documentation for this and other Volera products, and to get updates, see www.volera.com.

Volera Trademarks

Volera is a trademark of Volera, Inc. in the United States and other countries.

Third-Party Trademarks

Third-party trademarks (indicated by asterisks [*]) are the property of their respective owners.

Contents

	About This Guide	7
1	Content Controller Overview	9
	Content Controller Consists of Two Components	9
	Installing Content Controller	9
	Preparing to Use Content Controller	9
2	Prepopulating, Retaining, and Removing Cached Objects	11
	Understanding Collections and Jobs.	11
	Understanding the Actions You Can Apply Against Collections	13
	Understanding the Processing Order of Collections within Jobs	13
	Understanding URL Masks	14
	How Excelerator Resolves Conflicts Between Multiple Jobs	18
	Prepopulating Content in Cache	19
	Prepopulating HTTP Content.	19
	Prepopulating Streaming Content	20
	Retaining Content in Cache (Pinning)	24
	Preventing Content from Being Cached	25
	Removing Content from Cache	25
3	Creating Content Collections and Jobs	27
	Avoiding Job Configuration Conflicts.	27
	Creating Content Collections.	28
	Creating and Executing Content Jobs	29
4	Ensuring Cache Freshness	33
	Introducing Cache Freshness	33
	Managing Cache Freshness	34
	How Excelerator Checks for Object Freshness	34
	How Excelerator Keeps the Oldest Cached Objects Fresh	34
	How Excelerator Handles the Freshest Objects in Cache	35
	Fine-Tuning Cache Freshness on Network Cache Devices	35
	Using Custom Cache Control Headers	36
	An Overview of How Headers Work	36
	Implementing Custom Cache Control Headers.	37
	An Implementation Example	38

5	Monitoring Content Delivery	41
	Content Monitoring	41
	Understanding How Content Reporting Is Triggered	42
	Understanding How Long Content Statistics Are Displayed	42
	Understanding Monitoring Graphs	43
	Viewing Content Status	43
	Understanding How Failure Reporting Works	43

About This Guide

This guide explains how Content Controller helps you manage content on your network and covers the topics outlined in the following table.

To	See
Learn about how Content Controller works and how to modify its configuration settings and performance	Chapter 1, “Content Controller Overview,” on page 9
Learn how to prepopulate the cache, retain objects in cache, and remove objects from cache	Chapter 2, “Prepopulating, Retaining, and Removing Cached Objects,” on page 11
Define content collections and create jobs for storing and managing the collections in your content network’s caches	Chapter 3, “Creating Content Collections and Jobs,” on page 27
Manage the freshness of cached objects on your network	Chapter 4, “Ensuring Cache Freshness,” on page 33
Monitor content delivery	Chapter 5, “Monitoring Content Delivery,” on page 41

1

Content Controller Overview

For a high-level overview of Content Controller functionality, see [Content Controller Overview](#) in *Planning Guide*.

Content Controller Consists of Two Components

Content Controller consists of two components: Content Controller Server and Content Controller Agent.

- ♦ **Content Controller Server:** Runs on a Velocity management server with System Controller installed.

Content Controller Server is the management suite component that you interact with. It lets you create content collections that define the content you want to manage and jobs that schedule Excelerator cache devices to perform actions on the content defined in the collections.

- ♦ **Content Controller Agent:** Is an installed component on all Excelerator 2.1 and later cache devices, but it remains idle until activated by Content Controller Server.

Installing Content Controller

Instructions for installing Content Controller are provided in [Getting Started with Content Controller](#).

Preparing to Use Content Controller

Content Controller functionality is available in the Velocity management suite browser-based management tool. For information on using this tool, see [Using](#)

the Browser-Based Management Tool in *System Controller Deployment Guide*.

2

Prepopulating, Retaining, and Removing Cached Objects

Content Controller lets you use content collections and jobs to manage the content on your content network. This chapter explains how Excelerator cache devices process the collections and jobs you will create in **Creating Content Collections and Jobs** to accomplish various content management tasks.

Understanding Collections and Jobs

The Velocity management suite lets you manage the distribution, caching, and delivery of content objects at whatever level suits your purposes.

For example, you can manage a specific media stream, or a group of graphics files, or all the objects on a domain if you need to.

The fundamental object-management mechanism in the Velocity solution is the content collection.

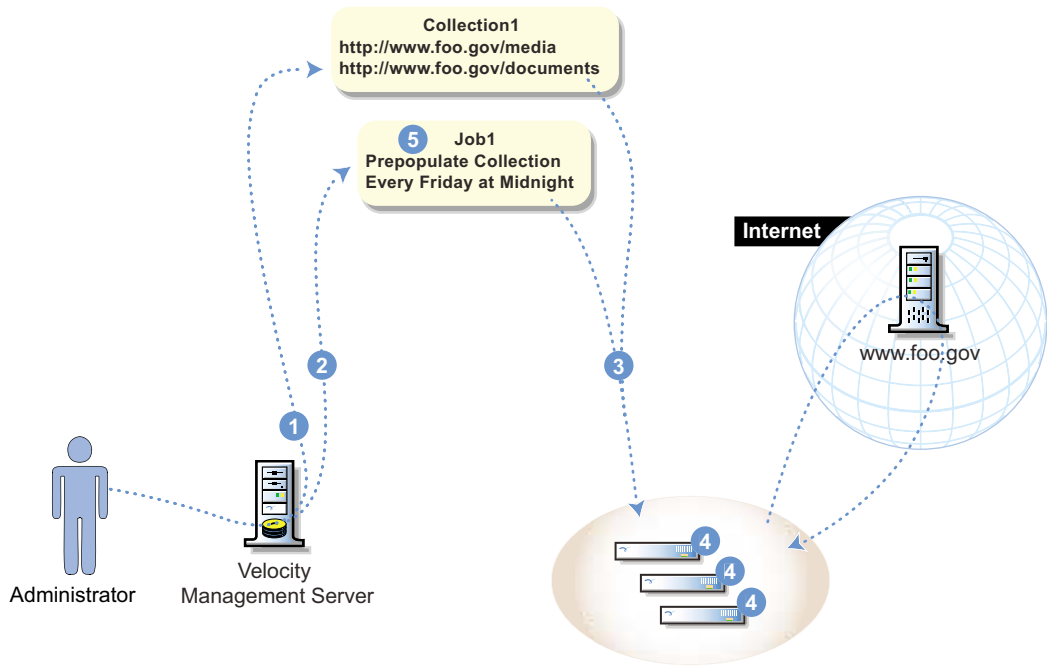
Content collections specify the content you want to manage. Each collection contains a list of URLs and URL masks that point to one or more objects on the network.

You create content collections using Content Controller. But creating collections is only the first part of the management process. After you define the objects you want to manage, you must then specify the actions (caching, prepopulating, retaining in cache, removing from cache, etc.) that you want to apply to each of these collections you have created.

The mechanism for applying actions to the objects in content collections is the content job.

Figure 1 summarizes the interaction of collections, jobs, and device groups.

Figure 1



- 1 The administrator creates collections, which contain URL masks that identify content to be managed .
- 2 The administrator then creates jobs that apply specific actions such as prepopulate, pin, purge, etc. against the collections.
- 3 The collections and jobs are sent to the devices in the device groups designated in the job.
- 4 The devices in the group execute the actions specified for each collection. (In this example the action is to prepopulate content from the Web.)
- 5 Other actions include: designating content to be pinned in cache, purging cached content, or specifically not caching content when it is requested through the device's proxy services.

This following sections explain how Excelerator cache devices process content jobs.

Understanding the Actions You Can Apply Against Collections

NOTE: Keep in mind that although actions in jobs are applied against collections, the actions actually happen on an individual object basis.

A single job can contain multiple collections, each of which can have any of the following actions associated with it:

- ◆ **Normal:** The system watches for objects covered by the collection's URL masks. If the objects are retrieved in response to user requests, they are cached. However, the objects are not pinned in cache. Cache devices track and report monitoring statistics for each collection with monitoring enabled.
- ◆ **Prepopulate:** The system downloads all objects covered by the collection's URL masks at the time and/or interval scheduled, but it doesn't pin the objects in cache. Cache devices track and report monitoring statistics for each collection with monitoring enabled.
- ◆ **Pin:** The system watches for objects covered by the collection's URL masks. If the objects are retrieved in response to user requests, they are pinned (retained) in cache for the period specified. Cache devices track and report monitoring statistics for each collection with monitoring enabled.
- ◆ **Prepopulate and Pin:** The system downloads all objects covered by the collection's URL masks and pins the objects after they are downloaded. Cache devices track and report monitoring statistics for each collection with monitoring enabled.
- ◆ **Bypass:** The system watches for objects covered by the collection's URL masks. If the objects are retrieved in response to user requests, the system passes them through and doesn't cache them.
- ◆ **Purge:** At the start of a job processing cycle and/or when the cycle completes, the system remove all objects from cache that are covered by the collection's URL masks.

Understanding the Processing Order of Collections within Jobs

If a cache device receives a job that has multiple collections with different actions, it will process each collection in order by action type as follows:

1. **Purge:** Collections with purge actions are processed first in the order listed.

NOTE: In contrast with other action types, purging only occurs at the beginning of each cycle for processing the job's collections (as specified by the Process Collections Frequency setting) and at the end of the cycle (if so specified).

This lets you ensure two things: First, that any previously cached content covered by the collection is removed from cache unless caching of the content is covered by another job (see [“How Excelerator Resolves Conflicts Between Multiple Jobs” on page 18](#)). Second, it ensures that once designated objects have been purged, they can be pinned, prepopulated, and so on as specified for other collections within the job.

2. **Pin or Prepopulate and Pin:** Collections with either of these actions are processed next in the order listed.
3. **Normal or Prepopulate:** Collections with either of these actions are processed next in the order listed. A normal action will never replace a pin action put in place for a given object by a previously processed collection. However, a prepopulate action could cause objects that were previously targeted for pinning to also be prepopulated.
4. **Bypass:** Collections with this action type are processed last. Objects that were targeted by an earlier pin, prepopulate, or normal action will not be targeted for bypassing cache.

Understanding URL Masks

URL masks are the basic building blocks of content management. It is therefore critical that you understand the rules for creating them and how the system interprets them.

About Wild Cards in URL Masks

Only the asterisk (*) wild card is allowed in URL masks.

Excelerator interprets everything between an asterisk and the next delimiter to the right (a forward slash [/], a period[.], or a colon [:]) as a wild card. This effectively allows only one asterisk between delimiters.

How Excelerator Matches Objects Against URL Masks

Often URL masks can affect many different objects stored on a cache device. Conversely, a given object might be covered by multiple URL masks.

It is therefore critical that you understand the process Excelerator follows in matching objects against URL masks before you begin creating collections.

There are three basic points to keep in mind:

1. Excelerator evaluates each object separately as the object passes through the system and follows a specific processing order in determining whether the object is covered by a URL mask.
2. Once a matching mask is found, Excelerator applies the action associated with the mask's collection and object processing stops.
3. The explanations in **Table 1** refer to the actions that Excelerator applies to objects requested by users. For information on prepopulation, see **“Prepopulating Content in Cache” on page 19**.

Table 1 introduces four categories of URL masks you can use when creating collections, and it explains the rules of order that Excelerator cache devices follow in matching objects and masks when determining whether normal caching, pinning, purging, or bypassing is required.

Table 1

Type	Examples	Notes
Hostname	<p>http://www.foo.gov/documents/picture.gif</p> <p>http://www.foo.gov/documents/</p> <p>http://www.foo.gov/</p> <p>foo.gov/documents/</p> <p>foo.gov/</p> <p>*.foo.gov/</p>	<p>Exceleator checks objects against the hostname masks in a given collection first. (Collection processing order is discussed in “Understanding the Processing Order of Collections within Jobs” on page 13.)</p> <p>Exceleator also processes the most specific hostname masks in the collection first. (The hostname mask examples in the middle column flow from most specific to least specific.)</p> <p>Although hostname masks can include the protocol or scheme, the DNS name, the path, and the filename, only the DNS name (hostname) is required.</p> <p>All DNS name portions must be indicated, if only by an asterisk wildcard (see the sixth mask).</p> <p>If a scheme is not indicated, the system automatically inserts http:// when processing a hostname mask.</p> <p>Exceleator interprets hostnames literally. For example, the sixth entry would cover www.foo.gov, ww1.foo.gov, army.foo.gov, etc., but the fourth and fifth entries would cover only foo.gov because Exceleator assumes that a scheme immediately precedes the hostname.</p> <p>IMPORTANT: Prepopulation actions only work with hostname URLs and require a complete, resolvable DNS hostname. Therefore, the sixth mask will not work for a prepopulation action. For more information, see “Prepopulating Content in Cache” on page 19.</p>

Type	Examples	Notes
Path	<p>/documents/picture.gif</p> <p>/documents/picture.gif/</p> <p>/documents/</p>	<p>If no hostname entries match an object, Excelerator begins comparing the objects against the path masks in the collection.</p> <p>Excelerator assumes that the first forward slash in the path mask immediately follows some hostname.</p> <p>For example, the first entry would apply only to a file named PICTURE.GIF that is located in a DOCUMENTS directory at the root of any host.</p> <p>The forward slash in the second entry causes Excelerator to assume that PICTURE.GIF is a directory.</p> <p>For example, a Pin action associated with this entry would apply to any matched objects that have a URL directory path that starts with a DOCUMENTS directory followed by a subdirectory named PICTURE.GIF.</p> <p>The third entry would apply to any matched objects that contain a DOCUMENTS directory at the start of their URL paths.</p>
Filename	<p>/picture.gif</p> <p>/widget.js</p> <p>/default.htm</p>	<p>After all path masks in a collection have all been compared against an object, Excelerator processes filename masks.</p> <p>For example, if requested objects named PICTURE.GIF, WIDGET.JS, and DEFAULT.HTM have not been covered by one of the hostname or path entries above, the files will have the pin type rule for their respective filename mask applied to them.</p> <p>If the first entry carries a pin type rule of Bypass, all PICTURE.GIF files that didn't match previously processed hostname or path masks would not be cached.</p>

Type	Examples	Notes
File Extension	/* .gif /* .js /* .htm	File extension entries are processed last. These are simply filename entries with the root of the filename replaced by an asterisk, which makes them less specific than complete filenames. For example, if the examples shown all had pin types of Bypass, then only those .GIF, .JS, and .HTM files that had been cached and pinned because of hostname, path, or filename masks would be stored in cache. All other files with the named extensions would not be cached.

Exceleator Uses IP Addresses After Resolving URL Masks

As stated earlier, you should include fully qualified DNS or hostnames in URL masks whenever possible.

Exceleator resolves DNS names to their respective IP addresses and uses those addresses when pinning objects. This has interesting implications for pinning.

For example, if you use the DNS name `www.foo.gov` as the URL mask and you know that the DNS name `foo.gov` resolves to the same IP address, you don't need to include a `foo.gov` URL mask in the collection. Since both URLs resolve to the same IP address, Exceleator will treat objects for both DNS names the same.

On the other hand, if `www.foo.gov` and `foo.gov` resolve to different IP addresses, separate URL masks would be required to cover both sites.

How Exceleator Resolves Conflicts Between Multiple Jobs

Since a single cache device can have multiple jobs applied against it, it logically follows that cache devices might receive conflicting action requests from different jobs for the same objects (URL masks).

To resolve conflicts between jobs and minimize bandwidth consumption, Exceleator devices use the following order of precedence when they encounter conflicting action requests for the same objects (URL masks).

Table 2

Original Job Action Directive for Object	Resolution if a Conflicting Directive Is Received
Pin or Prepopulate and Pin	These actions take first precedence and are treated as equals by the system. Since prepopulation occurs at the beginning of each processing cycle, there is no net system impact if these actions replace each other for a given URL mask.
Normal or Prepopulate	These actions are retained unless the new action is to mark the object as pinned. They take second precedence and are treated as equals since there is no net effect when they replace each other for a given URL mask.
Bypass or Purge	These actions are only applied if no other actions are in place for a given URL mask. They are treated as equals and are not distinguished since they both have the same effect.

Prepopulating Content in Cache

Prepopulating HTTP Content

Prepopulation of objects to cache requires that Excelerator actively retrieve objects from the Web. As the information in [Table 1](#) points out, only URLs that include a hostname can be used for prepopulation purposes.

When creating URL masks for prepopulation purposes, keep the following in mind:

- ◆ Only hostname URL masks are valid for prepopulation actions, and the masks must include a complete DNS name.
- ◆ Requests to DNS and subsequently to the origin server are built around the hostname mask exactly as it appears in the collection.
- ◆ If the hostname mask doesn't include a scheme, Excelerator includes `http://` in the request.
- ◆ If the hostname mask ends with a filename (no forward slash) then Excelerator initially requests only that file.

If the file returned contains additional links, Excelerator uses the settings specified for the mask, such as Follow Link Levels, Follow Links to

Other Sites, Max Objects, etc. to determine how many additional requests to issue to the origin server and other sites when applicable.

- ◆ If the mask ends with a forward slash, Excelerator assumes the mask ends with a directory path and that the origin server will return one or more objects just as it would in response to a browser or other Web request.

If the objects returned contain additional links, Excelerator uses the settings specified for the mask, such as Follow Link Levels, Follow Links to Other Sites, Max Objects, etc. to determine how many additional requests to issue to the origin server and other sites when applicable.

Prepopulating Streaming Content

The information contained in [“Prepopulating HTTP Content” on page 19](#) also applies to prepopulating streaming content.

General Guidelines for Prepopulating Streams

IMPORTANT: To optimize bandwidth usage and contain download costs, we recommend you prepopulate streams during low-traffic hours when upstream rates are usually lower.

The following points apply to prepopulating streaming content.

- ◆ A forward proxy service for the media type must exist on each device and be active when prepopulation begins.

For Windows Media, an HTTP Forward Proxy Service on the same IP address is also required

- ◆ Prepopulation of media streams can consume multiple connections per stream. If a stream contains multiple tracks at different bit rates, the prepopulation will require one connection for the audio track and one for each bit rate encoded in the stream.

NOTE: Content Controller is only aware of one connection per stream.

- ◆ Slow-links download support lets you download high-bit-rate streams over a slow connection. This is automatic and allows serving of high-quality streams on the local content network once the stream is downloaded.

NOTE: QuickTime streams are not compatible with slow-link downloads.

- ◆ Prepopulating streams takes at least as long as playing the stream and can take much longer if you are prepopulating over a slow link.

For example, a stream encoded for 700 Kbps that is 2 minutes long could take about one hour to prepopulate over a 56 K connection.

- ◆ Downloading streams can easily saturate network bandwidth. We recommend, therefore, that you create multiple jobs with small collections and schedule them to download when rates are low and at different times.
- ◆ Downloading from the Web usually requires that you specify at least 2 link levels for stream collections since the first two levels (level 0 and level 1) are usually text files that only contain links to the stream.

Prepopulating Apple QuickTime Content

Most job actions (pinning, purging, bypass and normal management) work the same for Apple QuickTime streaming content as they do for HTTP content.

Prepopulation of Apple QuickTime streams requires specific planning and configuration.

To include Apple QuickTime streams in collections, if the job action associated with the collection is Prepopulate or Prepopulate and Pin, you must do the following:

1. Create and enable an Apple QuickTime forward proxy service in System Controller.
2. Specify the RTSP protocol moniker in the URL masks you specify for the collection in Content Controller.

For example, the URL mask for prepopulating the stream *film.mov* into cache from a media server located at *www.media.com* would be:

```
rtsp://www.media.com/film.mov
```

IMPORTANT: We recommend you limit the number of URL masks in the collection when prepopulating Apple QuickTime streams for the following reason.

The cache device will attempt to simultaneously prepopulate all of the streams in the collection. If your cache device exhibits performance problems due to excessive bandwidth consumption or high CPU utilization, you will need to limit the size of the collection.

Prepopulating Windows Media Content

Most job actions (pinning, purging, bypass and normal management) work the same for Windows Media streaming content as they do for HTTP content.

Prepopulation of Windows Media streams requires specific planning and configuration as explained in the following sections.

Prepopulating MMS Streams

NOTE: Prepopulation requires that the origin media server have HTTP support enabled. Otherwise streams cannot be prepopulated.

To include MMS streams in collections, if the job action associated with the collection is Prepopulate or Prepopulate and Pin, you must do the following:

1. Create and enable an HTTP forward proxy service in System Controller.
2. Create and enable an MMS forward proxy service in System Controller.
3. Specify the MMS protocol moniker in the URL masks you specify for the collection in Content Controller.

For example, the URL mask for prepopulating the stream *film.asf* into cache from a media server located at *www.media.com* would be:

```
mms://www.media.com/film.asf
```

IMPORTANT: We recommend you limit the number of URL masks in the collection when prepopulating MMS streams for the following reason.

The cache device will attempt to simultaneously prepopulate all of the streams in the collection. If your cache device exhibits performance problems due to excessive bandwidth consumption or high CPU utilization, you should consider reducing the streams in the collection or creating a play list for the streams (see [Prepopulating Play Lists \(.ASX Files\)](#) below). The latter course of action lets you limit the number of streams through the Max. Concurrent Connections field.

Prepopulating Play Lists (.ASX Files)

NOTE: Prepopulation requires that the origin media server have HTTP support enabled. Otherwise streams cannot be prepopulated.

To include play lists in collections, if the job action associated with the collection is Prepopulate or Prepopulate and Pin, you must do the following:

1. Create and enable an HTTP forward proxy service in System Controller.
2. Create and enable an MMS forward proxy service in System Controller.
3. Specify the HTTP protocol moniker in the URL masks you specify for the collection in Content Controller.

For example, the URL mask for prepopulating the play list *films.asx* into cache from a Web server located at *www.media.com* would be:

`http://www.media.com/films.asx`

4. When specifying the object attributes for the play list, set Follow Links to at least 2 and Follow Links to Other Hosts to Yes.

IMPORTANT: The cache device will attempt to simultaneously prepopulate up to the number of streams set for the Max. Concurrent Connections field. If your cache device exhibits performance problems due to excessive bandwidth consumption or high CPU utilization, you can reduce this setting.

Prepopulating Real Media Content

Most job actions (pinning, purging, bypass and normal management) work the same for Real Proxy streaming content as they do for HTTP content.

Prepopulation of Real Proxy streams requires specific planning and configuration as explained in the following sections.

Prepopulating Real Proxy Streams

To include Real Proxy streams in collections, if the job action associated with the collection is Prepopulate or Prepopulate and Pin, you must do the following:

1. Create and enable an Real Media forward proxy service in System Controller.
2. Specify the RTSP protocol moniker in the URL masks you specify for the collection in Content Controller.

For example, the URL mask for prepopulating the stream *film.rm* into cache from a media server located at *www.media.com* would be:

`rtsp://www.media.com/film.rm`

IMPORTANT: We recommend you limit the number of URL masks in the collection when prepopulating Real Proxy streams for the following reason.

The cache device will attempt to simultaneously prepopulate all of the streams in the collection. If your cache device exhibits performance problems due to excessive bandwidth consumption or high CPU utilization, you should consider reducing the streams in the collection or creating a play list for the streams (see [Prepopulating Real Proxy Play Lists \(.RAM/SMI\)](#) below). The latter course of action lets you limit the number of streams through the Max. Concurrent Connections field.

Prepopulating Real Proxy Play Lists (.RAM/.SMI)

To include play lists in collections, if the job action associated with the collection is Prepopulate or Prepopulate and Pin, you must do the following:

1. Create and enable a Real forward proxy service in System Controller.
2. Specify the appropriate protocol moniker in the URL masks you specify for the collection in Content Controller.

For example, the URL mask for prepopulating the play list *films.ram* into cache from a Web server located at *www.media.com* would be:

```
http://www.media.com/films.ram
```

For example, the URL mask for prepopulating the play list *films.smi* into cache from a Web server located at *www.media.com* would be:

```
rtsp://www.media.com/films.smi
```

Or

```
http://www.media.com/films.smi
```

3. When specifying the object attributes for the play list, set Follow Links to at least 2 and Follow Links to Other Hosts to Yes.

IMPORTANT: The cache device will attempt to simultaneously prepopulate up to the number of streams set for the Max. Concurrent Connections field. If your cache device exhibits performance problems due to excessive bandwidth consumption or high CPU utilization, you can reduce this setting.

Retaining Content in Cache (Pinning)

Excellerator cache devices use a variety of metrics to remove or purge content and free up disk space. Sometimes this results in vital content being purged because it has been posted for longer than the maximum time, or because it hasn't been requested the minimum number of times.

Content Controller lets you override the usual purge routines for content that you need to keep in cache. For example, some content demands high availability for quality of service reasons, even though it is accessed less frequently than other content in cache.

By using a combination of pinning and the principles explained in [Chapter 4, "Ensuring Cache Freshness,"](#) on page 33, you can guarantee that important content stays put and stays fresh.

Preventing Content from Being Cached

You can prevent content from being cached by creating a collection of URL masks that target the content and then applying a Bypass action to the collection.

Keep in mind that if other collections cover any of the content you don't want cached, and if these other collections have any overriding actions such as prepopulate, pin, or normal associated with them, the Bypass action will have no effect. For more information, see [“Understanding the Processing Order of Collections within Jobs” on page 13](#).

Removing Content from Cache

You can remove content from cache by creating a collection of URL masks that target the content and then applying a Purge action to the collection in a job.

Keep in mind that purge actions occur only if no other jobs applied to the device have collections with actions that would cause the same content to be prepopulated, pinned, or cached normally.

3

Creating Content Collections and Jobs

The power of Velocity™ Management Suite content management comes from creating content collections and including the collections into content jobs. Content jobs target individual cache devices through their associated device groups. By creating jobs and scheduling them to run on the network's cache devices you can completely control what content is available, when it is available, and where it is located on your network.

Avoiding Job Configuration Conflicts

As you develop a content distribution strategy for your network, it is vital that you approach collection and job creation methodically. Otherwise, your content network implementation will not run as efficiently as it might, nor will it provide the service that your customers need and expect.

Additionally, you could duplicate content requests in multiple jobs. This could, in turn, cause double-billing problems.

IMPORTANT: Content Controller Server and Content Controller Agent work with each other to identify and remove duplicate object references within a single job to protect against the possibility of double billing through Content Accountant.

However, this protection does not cover duplicate object references across multiple jobs. Therefore, collections for trending and billing need to be split into separate jobs to protect against double billing.

For help with planning your collections and jobs, see [Planning Your Content Network](#) in *Planning Guide*.

Creating Content Collections

NOTE: We recommend that you plan your content collections using the [Velocity Management Suite Planning Worksheets \(../pdf/worksheets.pdf\)](#) and the instructions in [Planning Your Content Network](#) in *Planning Guide*.

If you haven't already done so, we also recommend you review the information in [Chapter 2, "Prepopulating, Retaining, and Removing Cached Objects,"](#) on page 11 before continuing with the collection creation process.

To create a content collection, complete the following steps:

- 1 Access the Velocity management server using a compatible browser.
For help accessing the management tool, see [Using the Browser-Based Management Tool](#) in *System Controller Deployment Guide*.
- 2 Click Content > Collection List > Create New Collection.
- 3 If you created a [worksheet \(../pdf/worksheets.pdf\)](#) for this collection, refer to your notes as you complete the remaining steps.
- 4 If you are using one or more private CDNs, choose the CDN to which this collection will be assigned. Otherwise, skip to [Step 5](#).
For more information about private CDNs, see [Scoping the Need for Multiple CDNs](#) in *Planning Guide*.
- 5 Type a name for the collection.
A collection name must include from one to 16 alphanumeric characters. The name might give an indication as to the content of a collection or the target end users for collection content.
- 6 If desired, type a description of the collection and any notes for future reference.
- 7 Click the Continue to Step 2 button.
- 8 If you have created a text file containing a list of URL masks to import into the collection, click Importing from File > browse to the file > click Import File > continue with [Step 10](#).
Otherwise, skip to [Step 9](#).
For more information on URL masks, see ["How Excelerator Matches Objects Against URL Masks"](#) on page 14.
- 9 Click Entering Manually > type the URL masks for the collection's objects (or copy the masks to the clipboard and paste them in).

For more information, see [“How Excelerator Matches Objects Against URL Masks”](#) on page 14.

- 10 If the same attributes (link levels, etc.) apply to the entire collection, you can specify the attributes before continuing.

For more information on the attributes associated with each collection, click Help.

- 11 Click OK & Return to Specify Objects.
- 12 If the URL masks (objects) in the collection have different attributes associated with them, click each mask and set its attributes.
- 13 Click the Continue to Step 3 button.
- 14 If you want to monitor collection activity, enable Content Collection Monitoring > specify a monitoring frequency > type the IP address of the Velocity management server where the management suite is installed and running.

IMPORTANT: Statistics become available only when there is collection activity to report and after twice the reporting frequency has elapsed. The server waits during the first period after there is collection activity before starting to collect information, then it collects the data during the next period and reports when the period ends.

- 15 If you are using Content Accountant to aggregate and report content statistics, enable Content Collection Reporting and specify the settings required by Content Accountant.

For more information on the attributes associated with each collection, click Help.

- 16 Click the Continue to Step 4 button.
- 17 Review the configuration and click the appropriate Edit button or click the item link to make any required changes.
- 18 Click Add Collection Now.

Creating and Executing Content Jobs

IMPORTANT: Jobs must include at least one content collection and one device group. You can create collections while creating a job or beforehand, but you must create at least one device group and assign it to the job's CDN before creating the job. Otherwise, the system will not let you complete the job creation process.

To create a content job, complete the following steps:

- 1 Access the Velocity management server using a compatible browser.
For help accessing the management tool, see [Using the Browser-Based Management Tool](#) in *System Controller Deployment Guide*.
- 2 Click Content > Job List > Create New Job.
- 3 If you created a [worksheet \(../pdf/worksheets.pdf\)](#) for this job, refer to your notes as you complete the remaining steps.
- 4 If you are using one or more private CDNs, choose the CDN to which this job will be assigned. Otherwise, skip to [Step 5](#).
For more information about private CDNs, see [Scoping the Need for Multiple CDNs](#) in *Planning Guide*.
- 5 Type a name for the job.
A job name must include from one to 16 alphanumeric characters. The name might give an indication as to the collections the job uses, the job's purpose or some other string to uniquely identify the job.
- 6 If desired, type a description of the job and any notes for future reference.
- 7 Click the Continue to Step 2 button.
- 8 Check one or more collections you want the job to use > click the Continue to Step 3 button.
- 9 For each collection you have included in the job, select an Action and specify whether you want URL download failures reported for the collection.
For more information about the actions you can apply, click help or see [“Understanding the Actions You Can Apply Against Collections”](#) on [page 13](#).
- 10 Click the Continue to Step 4 button.
- 11 Enter the job schedule information.
Click Help for more information.
- 12 Click the Continue to Step 5 button.
- 13 Check one or more device groups to which you want the collection actions applied.
- 14 Click the Continue to Step 6 button.
- 15 Review the configuration and click the appropriate Edit button or click the item link to make any required changes.

- 16** Click Add Job Now or Add/Enable Job Now if you want the job to become active immediately.

4

Ensuring Cache Freshness

The information in this chapter actually involves the use of System Controller to configure proxy service and other settings.

It is included in this guide with Content Controller information because it is so closely tied to content management. However, only administrators with System Controller access will be able to do most of the tasks described in this chapter.

Introducing Cache Freshness

When first introduced to Web content caching, many network administrators assume that the Excelerator object cache is basically the same as a browser's cache, which all users access when they click the Back button. The logical extension from this assumption is the fear that Excelerator will serve stale content that doesn't accurately reflect the fresh content on the origin Web server.

Actually, most time-sensitive Web content is flagged by Webmasters in such a way that it cannot become stale unless a caching system ignores the Webmaster's settings. And in fact, the Excelerator honors all flags that affect cache freshness, including Time to Expire, Don't Cache, and Must Revalidate directives.

In addition, the Excelerator can be fine-tuned for cache freshness in the following ways:

- ◆ Accelerated checking of objects that have longer than desirable Time to Expire headers.
- ◆ Delayed checking of objects that have shorter than desirable Time to Expire headers.

- ◆ Checking objects for freshness that do not include Time to Expire headers.

Managing Cache Freshness

Cache freshness is a primary concern of most content network administrators. The following sections briefly explain how your cache device ensures fresh content for network users and the options you have for adjusting this feature.

How Excelerator Checks for Object Freshness

Although the following explanation is an over-simplification, it lays the foundation for the specific examples that follow this section.

Each Excelerator cache device has timers that it applies to every cached object.

Each time an object is cached or revalidated, the cache device starts a timer for that object. As long as the timer is running, the cache device will vend the object from cache. After the time has expired and when the cache device receives a request for the object, it will issue an IF-MODIFIED-SINCE request to the origin Web server.

If the object has changed, Excelerator retrieves the updated object into cache and serves it to the requesting browser before restarting the timer.

If the object has not changed, the cache device vends the object from cache and resets the timer, and the countdown for vending the object from cache begins again.

If a browser forces a refresh of the object, Excelerator honors the browser request, retrieves and caches the object regardless of whether it has changed, and restarts the timer.

How Excelerator Keeps the Oldest Cached Objects Fresh

More than 80% of all Web objects have either no Time to Expire directives or they are set to stay cached for as long as weeks or even months.

Since many of these objects actually change fairly frequently, the cache device has two timers for ensuring their freshness. You can configure these timers in System Controller.

HTTP Maximum: This timer overrides an object's Time to Expire settings if it is longer than the timer's value.

The default timer value is six hours. This means that Excelerator will not vend an object that has been in cache longer than six hours without first checking whether it should be refreshed.

HTTP Default: Excelerator applies this timer to objects that don't have Time to Expire settings.

The default timer value is two hours. This means that Excelerator will not vend an object that has no Time to Expire setting that has been in cache longer than two hours without first checking whether it should be refreshed.

How Excelerator Handles the Freshest Objects in Cache

Most Webmasters ensure that their time-sensitive objects have appropriate Time to Expire directives. Late-breaking news stories and photographs, for example, might stay in cache for only a few minutes before expiring.

By default, the Excelerator simply honors the Webmasters' instructions and revalidates the objects in cache as directed.

However, some cache device installations, such as those connected through a modem might need to limit how often these objects are refreshed. The cache device has a third timer for this purpose, also accessible in System Controller.

HTTP Minimum: This timer sets the minimum number of hours or minutes Excelerator will serve HTTP data from cache before revalidating it against content on the origin Web server. No requested object will be revalidated sooner than specified by this value.

The default value for this timer is 0, meaning that Excelerator honors the Time to Expire directive for each object (assuming, of course, it is not longer than the HTTP Maximum timer).

If the timer is set to a value other than 0, it then overrides any object's Time to Expire directive that is shorter than the value set.

Fine-Tuning Cache Freshness on Network Cache Devices

The default timer settings explained in the previous sections are tuned for most cache device installations. However, as you read about them you might think

of special requirements you have that could be met better if the default settings were adjusted.

Perhaps you are accelerating content that doesn't contain Time to Expire directives but changes frequently and needs to be refreshed more often than every two hours. You can adjust the HTTP Default timer in System Controller so that Accelerator refreshes the objects more frequently.

Perhaps you have severe Internet bandwidth restrictions and an environment with users who don't require object freshness checks every six hours. You can adjust the HTTP Maximum timer in System Controller to a different setting that meets your requirements and conserves bandwidth.

To access the cache freshness settings for a device in System Controller, do the following:

- 1 In the browser-based tool, click System > click Device List > click a device name.
- 2 Click Configuration > Cache Freshness (in the Cache Settings section).

If you choose to adjust the timer values, avoid settings that result in object refreshing more often than is necessary. Otherwise you could easily negate the bandwidth and response-time benefits of having the cache device on your network.

Using Custom Cache Control Headers

In addition to fine tuning cache freshness using the system's global HTTP timers as explained in [Fine-Tuning Cache Freshness on Network Cache Devices](#), you can configure each proxy service to recognize custom headers in HTTP packets. Your Web server can then use these headers for transmitting caching instructions that only the configured cache devices will recognize and follow.

An Overview of How Headers Work

Only the accelerator service containing the custom header definition follows the cache policies specified in the custom headers.

All other caches, including the non-configured caches, requesting browsers, and external proxy caches (transparent caches, client accelerators, etc.) do not recognize the custom headers and therefore follow only the cache policies specified by the standard cache control headers.

This means that you have the following options for configuring your Web server:

- ◆ You can specify that browsers and/or external caches cannot cache the objects, but the cache device can.

This lets you offload request-processing from the origin Web server while still requiring that users return to the site each time they request an object.

- ◆ You can also specify separate cache times for browsers, external caches, and the accelerator service you are defining.

Implementing Custom Cache Control Headers

To implement custom cache control headers, you must do the following:

1. Enter a header string in the Custom Cache Control Header dialog box, for example, MYCACHE.
2. Configure the Web server to send an HTTP header containing the defined string and the time in seconds that the object should be retained in cache, for example, MYCACHE: 60.

If the number is non-zero, Exceleator treats the reply as if it had the following headers:

```
Cache-Control: public  
Cache-Control: max-age= number
```

If the number is zero (0), Exceleator treats the reply as if it had the following header:

```
Cache-Control: no-cache
```

3. Ensure that the Web server continues to send standard HTTP cache-control headers so that browsers and external caches follow the caching policies you intend them to.

For example, you could do the following:

- ◆ Use an Expires or Cache-Control: Max-Age header to specify that browsers should cache an object for two minutes
- ◆ Use a Cache-Control: Private header to prevent external caches from caching the object at all
- ◆ Use a custom cache control header, such as MYCACHE: 1800, to indicate that the accelerator should cache the object for 30 minutes.

Custom Cache Control Headers override the following standard HTTP cache-control headers on the cache device but do not affect how browsers and external caches respond to them:

```
Cache-Control: no-store  
Cache-Control: no-cache  
Cache-Control: max-age= number  
Cache-Control: private  
Cache-Control: public  
Pragma: no-cache  
Expires: date
```

An Implementation Example

For example, you might do the following:

1. While configuring a Web server accelerator service, you enable Custom Cache Control Headers > click Header Options > insert a string in the Custom Cache Control Header list with the value of FOOTTL.

The cache device will now recognize FOOTTL as a custom cache control header on objects requested through the service you are configuring.

2. You then configure the accelerated Web server to send FOOTTL: 600 in the headers of objects you want to be cached at the cache device.

The cache device will recognize this header as overriding the standard HTTP cache-control headers listed above when objects are requested through the accelerator service you are configuring.

3. Finally, you ensure that the Web server continues to send the following standard HTTP cache-control headers:

- ◆ Cache-Control: Max-Age headers that cause browsers to cache objects for no longer than two minutes
- ◆ Cache-Control: Private headers that cause external caches to not cache the objects.

When your Web server sends an object with the FOOTTL header in response to a cache device request made through the accelerator service, your cache device recognizes the custom header and caches the object for 10 minutes. Requesting browsers cache the object for only two minutes. And external caches do not cache the object.

Thus, the cache device off loads a processing burden from the Web server by caching the frequently requested objects for 10 minutes (the value you specified in Step 2). Browsers, on the other hand, must always access the cache device to get the objects if their previous requests are older than two minutes. And the objects in the cache device's cache are kept fresh due to their relatively brief time-to-live value.

5

Monitoring Content Delivery

In addition to monitoring the effectiveness and status of caching servers and groups, you will usually want to monitor the status of specific content as well.

The monitoring functionality in Content Controller lets you determine the content delivery performance for a content collection, how long certain content should remain in the cache, details of who is accessing the content, when they are accessing it and from what location, and other critical business decisions.

Content Controller can also track the prepopulation status of collection objects as follows:

Content Monitoring

You must do the following for content monitoring to function.

- ◆ Create a content collection.
- ◆ Enable monitoring for the collection.
- ◆ Include the collection in a job.
- ◆ When specifying a job action, check the Report URL Download Failure Details check box.

NOTE: For more information on download failure notification, see [“Understanding How Failure Reporting Works”](#) on page 43.

- ◆ Enable the job.
- ◆ Wait to monitor status until the job action starts and the first time period specified by the monitoring frequency value has elapsed.

If monitoring is enabled for a collection, an icon appears in the Monitoring column for each Collection Name.

Understanding How Content Reporting Is Triggered

When you enable monitoring for a collection, the cache devices assigned to the job report monitoring statistics to the collection's monitoring server. The cache devices only report when there is activity for an object in the collection.

Triggers vary depending on the object type (HTTP, MMS, or RTSP).

Examples:

- ◆ For stream monitoring to be activated, a player must request a streaming object in a job's collection, the object must be successfully retrieved, and the player must then be closed.
- ◆ HTTP prepopulation requests must begin downloading.
- ◆ Stream prepopulation requests must successfully download at least one stream.

Understanding How Long Content Statistics Are Displayed

Statistics are displayed for the period of time defined by the Refresh Frequency specified for the collection. If the frequency is 30 seconds, you will need to access the monitoring during that time period to see activity that occurred before the time period started.

If playing start time is S , the total playing time is P , and reporting time frequency is F , the maximum time required for the statistics to show up in the monitoring screen is represented by the formula $S + P + F$.

For example, if a stream start time is 1:00 p.m., the movie is 2 hours long, and the reporting frequency is 15 minutes, the statistics will show up no later than 3:15 p.m. This assumes that the start time and reporting frequency happen to be synchronized, which is unlikely in an actual situation.

The statistics will then display in the monitoring screen for the period of time defined by the reporting frequency. In this example that would be for 15 minutes.

Understanding Monitoring Graphs

Graph displays are time-delayed by 20 minutes. However, they provide a historical view of object delivery while statistics only represent a snapshot of the activity for the time period defined by the Refresh Frequency.

The graph data is the average during the reporting time interval.

Viewing Content Status

To see the content distribution status, do the following:

- 1 Click Content Controller > Job List.
- 2 In the Status column of the job that contains the collections in question, click Enabled.
NOTE: Only enabled jobs can be checked.
- 3 In the Status column of the device you want to query, click the Active link.
- 4 In Status of Collections at Device, click the target collection name.

Understanding How Failure Reporting Works

As explained earlier, you must check the Report URL Download Failure Details check box if you want to monitor content.

When a URL download failure occurs, the cache device attempts to send failure notification only one time. If the management server cannot be reached, the failure report is lost. Clicking Refresh from Device will update all statistics except the failure notification.

