MediaVault User Guide

Content Delivery

Table of Contents

Overview

MediaVault is a high performance, server-side authentication service used to assist you in securing your content from unauthorized views. The system uses cryptographic hashing based on the MD5 algorithm. The algorithm is designed so that any changed byte of the hash propagates new changes throughout the remainder of the calculation. The result is a completely different hash that is nearly impossible to break.

MediaVault is scalable because all verification requirements are stored within the URL itself. This small service is easily inserted into the Edgio Networks CDN architecture, allowing local requests at the level of the proxy caches. Bottleneck conditions are significantly reduced due to independence from a central service, such as a verification server.

MediaVault is not a replacement for digital rights management (DRM), and should not be associated with user authentication. However, it can protect content from unauthorized viewing on a broader basis, and it can, more specifically, designate a "start time" and an "end time" for content availability.

This document describes *MediaVault* , which is supported on the following Edgio services:

- Edgio Content Delivery (HTTP and HTTPS)
- Edgio Video Delivery (On Demand Windows Media Live, and On Demand, and MP3)

Specific examples are provided in the following sections:

- *MediaVault* for HTTP
- Cookie-based *MediaVault*
- *MediaVault* for Windows Media Live and On Demand
- *MediaVault* for MP3

Workflow

1. Determine which *MediaVault* query string parameters and values you want to add to your protected URLs.
2. Create a server-side script or application that will convert unprotected URLs into *MediaVault* URLs. For each URL, the script should:
   - For each *MediaVault* parameter you chose (except the "&h=" parameter), calculate the values of the parameter if necessary, and append it to the URL query string.
   - Create a hash value based on the MD5 of the modified URL and the shared secret
   - Append the "&h=" parameter and the hash value to the URL
   - Publish the protected URL in place of the original URL

Cached Query Terms

By default, *MediaVault* instructs the *Content Delivery* service to strip all HTTP and HTTPS query terms before caching an object.

If there are specific query terms that your origin server uses to respond with different content, and which should therefore be retained to differentiate your objects in cache, you can enter them directly in the *Deliver* section of the *Configure* menu in in the content control portal (Step 4 – Advanced Cache > Specific Query String Caching > Keep next query terms) or ask your Account Manager for help.

> **Note:** Query-term stripping is applied only to cached objects, and does not affect requests to origin. All query terms, including *MediaVault* -specific terms, are included in origin requests.

Security

The Security section applies to the following services:

- *MediaVault* for Windows Media
- *MediaVault* for MP3
- *MediaVault* for HTTP

Shared Secret and Hashing

MediaVault provides a secure method of URL-embedded content protection by using a shared secret and a hash computation.

Shared Secret

The shared secret is an alphanumeric string known only to Edgio and the content provider. This secret word is pre-pended to the content URL before the hash is generated, and is unrecoverable from the hash. This prohibits malicious end users from creating their own hashes and protects *MediaVault* parameters from tampering.

Hash Secret

The hash secret is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value.

You must generate your own hash values. MD5 hash generator applications are available free on the internet. There is also an MD5 Hash Generator Tool available in Control that provides a hash generator for you to configure *MediaVault* features.

Two Hash Secrets

MediaVault allows two shared hash secrets to be in place at once.

> **Note:** This feature is ONLY used by *Content Delivery* .

- **Use Case:** This feature is used when you would like two viable production shared secrets, or you are transitioning from one shared secret to another.
- **Usage:** If you want to transition from one hash secret to another without breaking existing URLs, *MediaVault* will try to match the hash using the first hash secret. If the first hash secret fails, then *MediaVault* will try to use the secondary hash secret.

Limitations: There must be an existing hash secret in order to have a secondary hash secret.

Workflow for *MediaVault* hash usage:

1. You can use the target URL with one or more *MediaVault* parameters to create a 128-bit MD5 hash value. This hash is typically expressed as a sequence of 32 hexadecimal digits.
2. Edgio requires you to supply the hash, a secure directory and a shared secret.
3. *MediaVault* then uses the information to create the same hash.
4. If the hash expressions do not match, the consumer request for content is denied.

For optimal security, you should perform URL computation (with appropriate query parameters) outside the media player by using a separate software component for optimum security. For example, generate the final URL within a Content Management System or Web Service and have the media player query the component used for the final secure URL.

Custom URL Tokenization

MediaVault can store and access multiple sets of hash algorithms. Each set can contain separate MD5, SHA1 and SHA256 shared secrets. Individual sets are versioned (using an integer > zero) and are specified by version number using the new va *MediaVault* parameter.

To take advantage of this feature, please contact Edgio Support.

HTTPS URLs

When using HTTPS URLs, the *MediaVault* hash should be created using an HTTPS URL only. The URL may still be requested from a browser using HTTPS.

Control *MediaVault* Features

The *Control* supports *MediaVault* functionality for the following services:

- Windows Media
- MP3
- HTTP

To Generate a Hash:

1. Log into the **Control** (https://control.llnw.com.
2. Click **Content**, click **Secure** then click ***MediaVault*** .
3. Under the **Media Type** heading, select the service you wish to create a hash for.
4. Under the **Shared Secret** heading, enter a shared secret (An alphanumeric value known only to Edgio and the content provider).
5. Under the **Target URL** heading, enter the URL you wish *MediaVault* to protect.
6. Under the **Start Date** heading, select the date you wish your content to become available.
7. Under the **Start Time**heading, enter the time, on the selected start date, you wish your content to become available.

   > **Note:** All times specified are in Mountain Standard Time (MST / GMT-7)

8. Under the **End Date**heading, select the date you wish your content to become unavailable.
9. Under the **End Time** heading, select the time, on the selected end date, you wish your content to become unavailable.

   > **Note:** All times specified are in Mountain Standard Time (MST / GMT-7)

10. Under the **IP Address/Mask** heading, enter the IP address, IP address range or Mask you wish to make your content available to.
    - Individual IP address example: 10.9.12.19 – IP address 10.9.12.19 is allowed. All other requests will be denied.
    - Range of IP addresses example: 1.2.3.0/24 – IP addresses 1.2.3.0 to 1.2.3.255 are allowed. All other requests will be denied.
    - IP mask example: IP=69.28.133.70/24 – IP security with 24 bit mask (255.255.255.0) - valid
    - IP range: 69.28.133.1 - 69.28.133.254. All other requests will be denied.

11. Under the **Referrer URL** heading, enter the URL of the media player that is authorized to play your content. For more information.
12. Under the **Page URL** heading, specify the URL of the authorized page that clients will use to connect to your content.
13. Click the **Generate Output** button and the hash examples for the specified media type display in the MD5 Hash Output section.
14. Use this information as a validation for the hash value created locally.

Media File Extensions

When computing the token of a given URL, the extension of the media file may not be part of the hash. For example, when the media file to be secured is an FLV, MP3 or MP4 then the extension must be omitted from the hash calculation. Whereas, when the media file is any other format (MOV, M4V, F4V), then the extension must be included in the hash calculation.

Example values for generating tokens:

- FLV/a3176/o29/s/path/to/file/filename?e=1233216000
- MP3/a3176/o29/s/path/to/file/filename?e=1233216000
- MP4/a3176/o29/s/path/to/file/filename?e=1233216000
- MOV/a3176/o29/s/path/to/file/filename.mov?e=1233216000
- M4V/a3176/o29/s/path/to/file/filename.m4v?e=1233216000
- F4V/a3176/o29/s/path/to/file/filename.f4v?e=1233216000

MediaVault Parameters

This section provides a matrix and description for all *MediaVault* parameters. Not all parameters are supported for all services.

Usage examples are provided in the following sections:

- *MediaVault* for On Demand
- *MediaVault* for Windows Media Live and On Demand
- *MediaVault* for MP3
- *MediaVault* for HTTP

Supported Parameters by Service

The behavior of the CDN with *MediaVault* is based on the combination of parameters in the URL query string. The *MediaVault* parameters available to Edgio services are:

Service	Start Time (s)	End Time (e)	Time (t) ehash	IP Address /Mask (ip)	Prefix Length (p)	Referrer URL (ru)	Referrer (r)	Page URL (pu)	Rate Limit (ri, rs)	Algorithm Version (va)	Cookie-based (cf, cd, ci)
Content Delivery	Yes	Yes	Yes	Yes, but not wildcards	Yes	No	Yes	No	Yes	Yes	Yes
Windows Media Live and On Demand	Yes	Yes	No	Yes	Yes	No	No	No	No	No	No
MP3	Yes	Yes	No	Yes, but not masks	No	No	No	No	No	No	No

Format

The hash is generated using the following *MediaVault* Secure URL format for the query string:

`http://<published-url>?s=<start-time>&e=<end-time>&p=<length>&ip=<supported-IP-format>&h=<hash>`

where

- **s**=<start-time> The time the request is authorized from (represented as Unix epoch seconds)
- **e**=<end-time> The time the request is authorized to (represented as Unix seconds)
- **ip**=<supported-IP-format> A specific IP address or range of IP addresses
- **p**=<length> A positive integer representing a portion of the URL that will be used for computing the hash
- **ru**=<length> A positive integer representing number of characters
- **r** = <domains> a comma separated list of domains
- **pu**=<length> is a positive integer representing number of characters
- **ri =** <kbytes> An initial data range that will not be rate limited by **rs**
- **rs** = <kbytes/sec> The sustained rate limit to apply after **ri** is reached
- **va**=<version> is an optional hash algorithm version number
- **h**=<hash> is a MD5 hash of the URL

Cookie-Based *MediaVault* Parameters

- **cf** = A Cookie-based *MediaVault* parameter that defines the final expiration of an asset.
- **cd** = A Cookie-based *MediaVault* time delta. Expirations for *MediaVault* cookies are incremented according to this delta for each request.
- **ci** = A Cookie-based *MediaVault* parameter which specifies an initial delta when incrementing expiration values for a given request. After this delta is used one time, the delta from a cd= parameter is used.
- **cp** = Specifies a comma-separated list of paths (that can include wildcards) which are matched against the file path portion of an incoming request to determine if a URL is valid.

With a few exceptions, parameters can be in any order. However, the order must be consistent between when the hash value is calculated and when the *MediaVault* -protected URL is presented to the user. Changing the order of the parameters after the hash has been calculated will result in the hash value being rejected.

- The Start Time, End Time and IP parameters can be in any order.
- The Referrer URL and Page URL parameters must immediately follow the hash secret.
- When the Referrer URL and Page URL features are combined, the Referrer URL must come before the Page URL.

Start Time (s)

Start Time (**s**) represents the time from which the URL is valid. End users can't access the media before the start time. This is used to set a start time for content to go live in the future – without worrying that the audience may see a video before its scheduled time.

Notes:

- If any number in the Start Time set is changed without computing the hash again, the request is denied.
- Alltimes specified in the Control UI are in Mountain Standard Time (MST/GMT-7), but the time listed on a
- URL in the actual s= parameter is UTC.

End Time (e)

End Time (**e**) represents the time after which the URL is invalid. End users can't access the media after the end time. By encoding the end time in a URL, you can set a time-to-live for the content, preventing indefinite access.

Should a request have an end time that is greater than the time a request is received, the request will be considered unauthorized.

> **Note:** When Cookie-Based *MediaVault* is used, the expiration time for a URL is incremented by ci= and cd= parameters (until the cf= value is reached).

Start Time (s) and End Time (e)

MediaVault Start and End Time allow you to specify times that the URL will be valid. A URL can have a time that it starts being valid, a time it ends being valid, or both.

- **Use Case:** You want a link to be valid only during a certain time frame. For example, a link can be set up for a live event that only works during the event.
- **Usage:** To use this feature, add a "s=" and/or "e=" argument to the query string before the "h=" argument. If this parameter is not supplied, the link is valid at all times. If only the start "s=" parameter is supplied, then the link will be valid from that time forward. If only the end "e=" parameter is supplied, then the link will be valid until that time.
- **Limitations:** none
- ***MediaVault* Secure URL Format:** The hash is generated using the following format:

`http://<object-url>?s=<start-time>&e=<end-time>&h=<hash>` where `<start-time>` and `<end-time>` are the number of seconds since the Unix epoch and `<hash>` is an md5 hash of the URL.

Time (t) ehash

MediaVault ehash specifies the expiration time for a URL.

> **Note:** This parameter ONLY applies to *Content Delivery* .

- **Use Case:** You can specify an end date for the link in the hash parameter.
- **Usage:** To use this feature, add a "t=" argument to the query string. The ehash is generated with the same algorithm as a normal hash, but the end time is prepended to it, separated by an underscore. A normal hash secret with an end time (i.e., the "e=" parameter) will have the same result.
- **Limitations:** Your ehash is enabled and you will not be able to use the other options that are available for hash secret.
- ***MediaVault* Secure URL Format:**The hash is generated using the following format:

`http:// //<object-url>?t=<endtime_hash>where <endtime_hash> is the number of seconds since the Unix epoch, followed by an underscore, followed by an md5 hash of the URL.`

IP Address/Range Mask (ip)

The IP Parameter (**ip**) limits the URL to be from a specific IP or a range of IP addresses. If the IP address associated with the request is different or is not within the specified range, the request is denied.

Multiple machines behind a firewall will look like one IP to the server and can all use the link with that IP. Both IPv4 and IPv6 addresses are supported.

The argument to the IP parameter is <supported-IP-format>, represented by one of the following:

- 192.169.1.25 (specific IP address)
- 192.168.1.0/24 (range of IP addresses)
- 2607:f4e8:120:901::0/64 (range of IPv6 addresses)

If this parameter is not supplied, all IP addresses will be valid.

IP Address

The IP address allows a URL (or media file) to only be accessed by client(s) originating from a specific IP address such as 1.2.3.4.

IP Address example: This link will only be authorized if the IP the request came from is 10.9.12.19. All other requests will be denied:

`http://test.llnwd.net/wm9/md5/file.wmv?ip=10.9.12.19&h=d051d13c78d51a67508734cd6dc5d694`

IP Address Range

IP address range allows access from a range of logical addresses (usually within the address space assigned to the organization.) For example, the address 1.2.3.0/24 indicates a range (in the 4th octet) up to the 24th binary placeholder (00000001.00000010.00000011.00000000). This range would include addresses from 1.2.3.0 to 1.2.3.255. For assistance creating a range, connect to http://www.subnet-calculator.com and enter the IP addresses associated with the content.

IP Range example: This link will only be authorized if the IP the request it came from is within the 10.9.12.0/24 range. All other requests will be denied:

```
http://test.llnwd.net/wm9/md5/file.wmv?ip=10.9.12.0/24&h=d051d13c78d51a67508734cd6dc5d694
```

IP Mask

The IP Mask is an optional part of the IP parameter, indicating the number of bits to use in network identification. This feature was added to *MediaVault* to authorize requests based on the network ID when content requestors connecting via dynamic routes using NAT have a different IP address for each request. When this occurs, the requestor isn't authorized and can't access the content.

Testing may be necessary to identify networks that require masking and the optimal mask to apply. For example, a URL should be generated using a network mask when several authorized requestors cannot view the content or when requestors originating from networks known to use load balancing routers. Mask should be applied sparingly and set to the maximum number of bits that will allow the requestor access to the content. Masks lower than 8 bits should never be necessary.

Prefix Length (p)

Prefix Length (**p**) specifies a limited number of characters from the URL to calculate the hash. The hash calculation is done only on the portion of the full URL that matches the number of characters specified by the length parameter. This is typically used to remove a file name from the URL before the hash calculation is made, allowing access to all files in a directory. When the prefix length parameter is used with HTTPS, the MD5 calculation is done only on the portion of the full URL that matches the actual character count specified by the length parameter.

- **Use Case:** You want to generate a hash for a directory, not for specific files.
- **Usage:** To use this feature, add a "p=" argument to the query string before the "h=" argument. If this parameter is not specified, then the full URL will be used for the hash calculation. A hash secret (or primary hash secret and secondary hash secret).
- **Limitations:** If the value of the "p=" argument is greater than the length of the URL then the whole path will be used.
- ***MediaVault* Secure URL Format:** The hash is generated using the following format:

```
http://<object-url>?p=<P-value>&h=<hash>where <P-value> is a positive integer and <hash> is an md5 hash of the URL.
```

Referrer URL (ru)

The referrer is the URL of a previous “item” that led to the current request. In other words, the referrer is part of the HTTP request sent by the browser program to the web server. The Referrer URL in *MediaVault* is your media player (SWF) URL. The Referrer URL feature compares the referrer on the external request with the referrer passed in with the hash. If the referrer came from a non-designated source, the request is denied.

The **ru** parameter limits the number of characters in the Referrer URL that are used to generate the hash. "ru=1" means that the entire Referrer URL is used.

Referrer example: If the referrer is `http://www.yoursite.com/mediaplayer.swf`, and the Referrer URL parameter is 24 in the target URL, the media file or URL can only be accessed by client(s) originating from `http://www.yoursite.com/`, even though the referrer is `http://www.yoursite.com/mediaplayer.swf`. This is because the 'ru=' parameter is set to 24, which takes only the first 24 characters of the reported referrer (http://www.yoursite.com/).

Note: Unlike the s=, e=, or ip= fields, the substring representing the number of characters (N) specified by the ru= field must be included in the hash computation immediately following the secret.

Referrer (r)

When connecting to a server, the client may provide the location from which the request initiated in a Referrer header. The Referrer feature within *MediaVault* provides a method for you to restrict access to only authorized referring URLs.

- **Use Case:** You want the links to be protected so they are only served from a specific page.
- **Usage:** To use this feature, add a "r=" argument to the query string before the "h=" argument. If this parameter is not supplied, all referrers will be valid. A hash secret (primary hash secret and secondary hash secret).
- **Limitations:** Some links from some browsers may not include referrer data in the HTTP Header, which is required for this feature. Most normal web page links will send the referrer data, but some helper applications will not know what the referrer is and will not send the header. Referrer data can also be spoofed fairly easily by including the header in the request; thus it is not the most secure protection.
- ***MediaVault* Secure URL Format:** The hash is generated using the following format:

`http://<object-url>?r=<referrer>&h=<hash>`

where <referrer> must exactly match the value of the referrer data in the HTTP header and <hash> is an md5 hash of the URL.

Page URL (pu)

The Page URL feature compares the page URL on the external request with the Page URL passed in with the hash. If the Page URL does not match, the request is denied.

The **pu** parameter limits the number of characters in the Page URL that are used to generate the hash. "pu=1" means that the entire Page URL is used.

Page URL example: If the Page URL is `http://www.yoursite.com/page.html` and pu=33, the media file or URL can only be accessed by client(s) originating from `http://www.yoursite.com/page.html`, since the 'pu=' parameter is set to the length of the page url. If 'pu=24' instead, client(s) can originate from any pages on `http://www.yoursite.com/`. This allows an entire directory to be protected by the same hash.

Referrer URL (ru) & Page URL (pu) Combined

Referrer URL and Page URL features can be combined. However, when computing the hash of the URL with both the Referrer URL and Page URL checks, the Referrer URL must come before the Page URL.

Rate Limit (ri, rs)

- **ri** - Optional initial range that will not be rate limited (kbytes)
- **rs** - Sustained rate limit to apply after **ri** is reached (kbytes/sec)

Use Case

Rate Limit lets you specify a maximum data transfer speed, with an optional initial transfer of full-speed data. For example, you may want to rate limit video streams overall, but provide video players with an initial, full-speed burst of data or metadata to improve the user experience.

The **ri** parameter lets you set the initial amount of data that will not be rate limited, while the **rs** parameter lets you set the sustained rate limit to apply after the data specified by **ri** has been transferred.

- **Usage:** To use this feature, add the "rs=" argument, and optionally the "ri=" argument, to the query string before the "h=" argument. If the **ri** parameter is not added, then requests are satisfied at full speed.
- ***MediaVault* Secure URL Format:** The hash is generated using the following format: `http://<object-url>?ri=<unlimited Kbytes>&rs=<Rate Limit after ri is reached>&h=<hash>`
  where <hash> is an md5 hash of the URL.

Note: The `ri` parameter has a maximum value of `20000` KB, and the rs parameter has a maximum value of `10000` KB/s. If larger values are provided, they will be reset to these maximums.

Algorithm Version (va)

If you are using multiple hash algorithm sets, you can tell *MediaVault* which hash algorithm set to use with the **va** parameter. The parameter value must be an integer greater than zero.

For more information on using hash algorithm sets, see Custom URL Tokenization.

Cookie-Based (cf, cd, ci, cp)

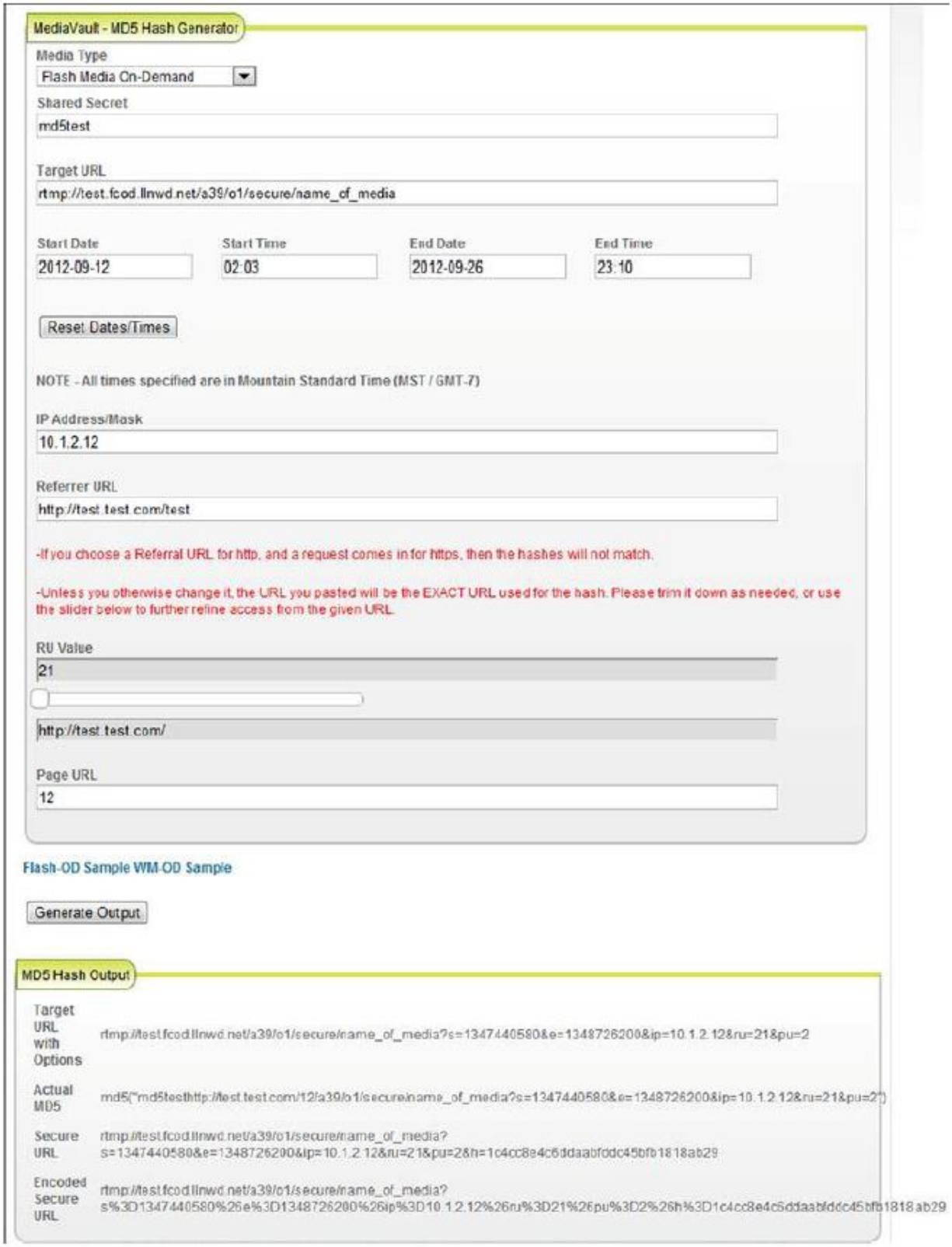
- **cf** - Final expiration. After the time specified in cf= is reached, content is unauthorized.
  Note: Expiration times, via the "e" parameter will be incremented in cookies until "cf" is reached.
- **cd** - Delta (in seconds) between the second request and all subsequent requests. In other words, the expiration time ("e" parameter) will be increased by this delta for the life of the content.
- **ci** - Initial delta (optional). The time delta used between first and second requests. When this is not present, "cd" is used in its place.
  Note: This parameter is never present in a cookie.
- **cp**: Paths. A comma-separated list of paths (that can include wildcards) which are matched against the file path portion of an incoming request to determine if a URL is valid.

Parameter Order

With a few exceptions, parameters can be in any order. However, the order must be consistent between when the hash value is calculated and when the *MediaVault* -protected URL is presented to the user. Changing the order of the parameters after the hash has been calculated will result in the hash value being rejected by the Edgio Delivery Network.

- The Start Time, End Time and IP parameters can be in any order.
- The Referrer URL and Page URL parameters must immediately follow the hash secret.
- When the Referrer URL and Page URL features are combined, the Referrer URL must come before the Page URL.

Example:



MediaVault also supports allowing some query terms to be appended to the end of the secure URL without being included in the hash calculation. This may be useful for certain use cases, such as when a video player adds arbitrary query terms to the end of the request. These query terms must appear after the hash value (h=) in the secure URL.

PHP Example

```
<?PHP
	$url = $_POST["url"];
	$secret = $_POST["md5"];

	if (!$url && !$secret) {
		$url = "http://limelighturl/file.ext";
		$secret = "Secret";
	}
?>
<html>
	<head>
		<title>MD5 Hash Sample Page</title>
	</head>
	<body>
		<form method="post">
			URL: <input type="text" name="url" value="<?PHP print $url;?>" size=80>
			<br>
			Secret: <input type="text" name="md5" value="<?PHP print $secret;?>">
			<br>
			<br>
			<input type="submit">
		</form>

		<?PHP
			if (ereg("[&?]h=", $url)) {
				print "<p>Oops! You should not include the h= portion of the url.
calculate it for you!</p>\n";
				exit();
			}

			$end = time() + 24*60*60; //24 hours in seconds
			$md5 = md5("$secret$url");

			if (strpos($url, '?') > 0) {
				$auth_url = "$url&h=$md5";
			} else {
				$auth_url = "$url?h=$md5";
			}

			print "<table border>\n";
			print "<tr><td>Secret:</td><td>$secret</td></tr>\n";
			print "<tr><td>Url with options:</td><td>$url</td></tr>\n";
			print "<tr><td>MD5 Hash Input:</td><td>$secret$url</td></tr>\n";
			print "<tr><td>Generated MD5 Hash:</td><td>$md5</td></tr>\n";
```

```
                        print "<tr><td>Authenticated URL</td><td><a

                        href=\"$auth_url\">$auth_url</a></td></tr>\n";

                        print "</table>\n";

                        if (strpos($url, 'e=') <= 0) {
                                print "If you want to have an expiring url that expires after a da
following into the URL box:\n";
                                print "<pre>\n";

                                if (strpos($url, '?'))
                                        print "$url&e=".(time()+24*60*60)."\n";
                                else
                                        print "$url?e=".(time()+24*60*60)."\n";

                                print "</pre>\n";
                        }

                        if (strpos($url, 'rs=') <= 0) {
                                print "If you want to have a rate limited url, enter the following
(where rs can equal any number greater than your minimum limit):\n";
                                print "<pre>\n";

                                if (strpos($url, '?'))
                                        print "$url&rs=85\n";
                                else
                                        print "$url?rs=85\n";

                                print "</pre>\n";
                        }

                        if (strpos($url, 'ri=') <= 0) {
                                print "If you want to allow the user to download a number of bytes
ratelimited, enter the following into the URL box (where ri equals the value in bytes
to not be rate limited ):\n";
                                print "<pre>\n";

                                if (strpos($url, '?'))
                                        print "$url&ri=1000\n";
                                else
                                        print "$url?ri=1000\n";

                                print "</pre>\n";
                        }
                ?>
```

```
        </body>
</html>
```

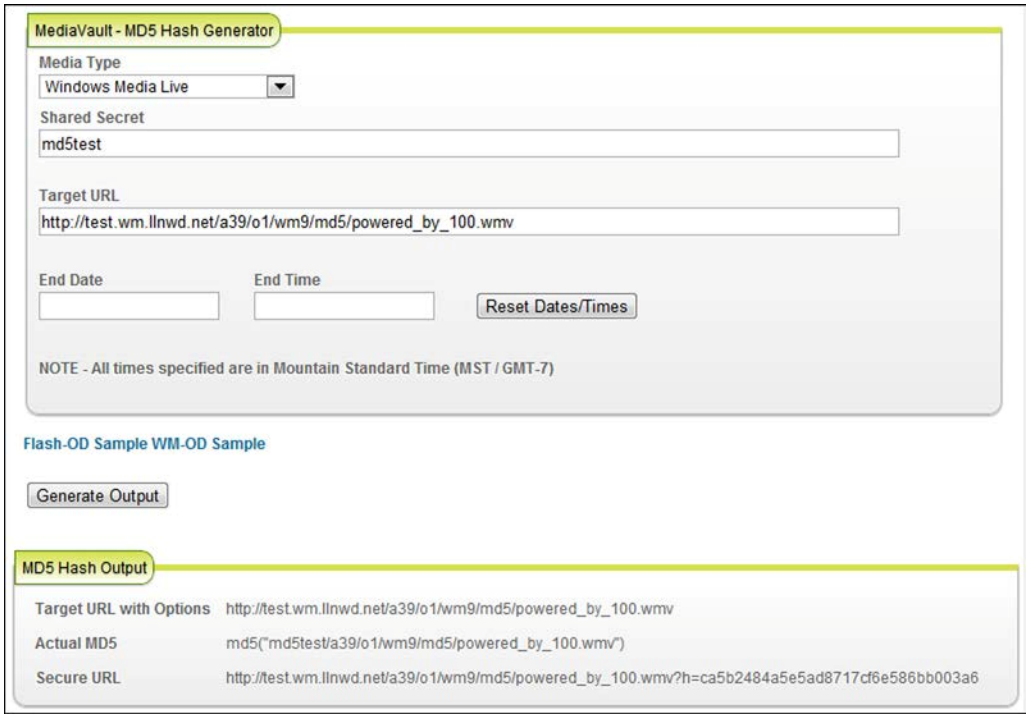
MediaVault for Windows Media Live and On Demand

Windows Media Live

The following parameters may be used with *MediaVault* for Windows Media Live to generate a hash value:

End Time (e)

End Time Example

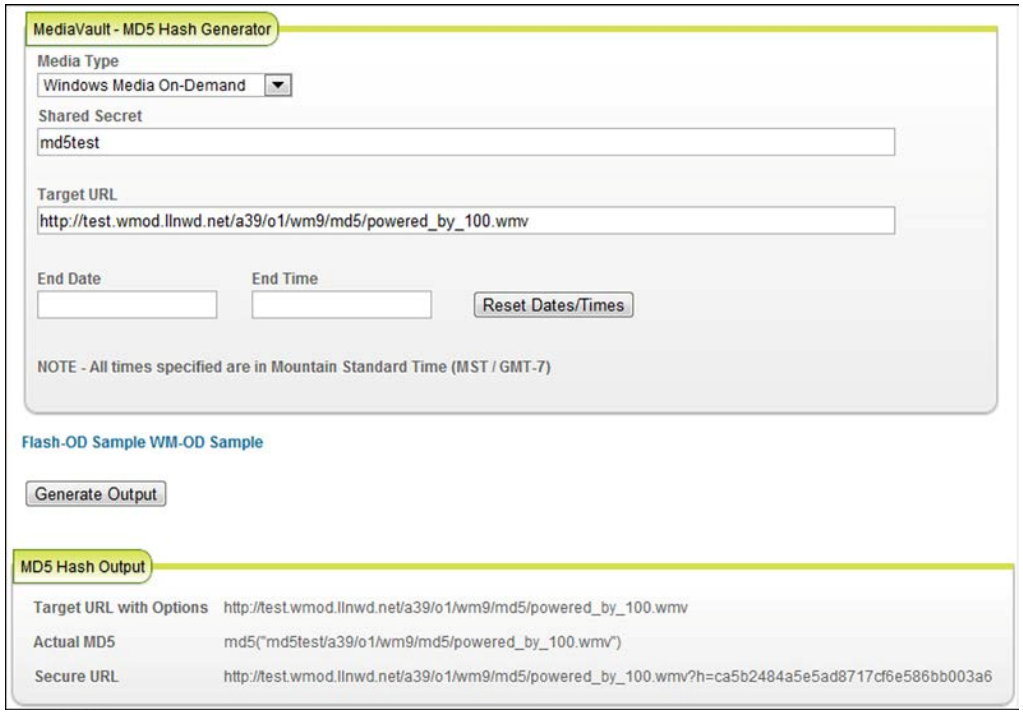


Windows Media On Demand

The following parameters may be used with *MediaVault* for Windows Media On-Demand to generate a hash value:

End Time (e)

End Time Example



ASP Server Script Example

```
<% @LANGUAGE=VBSCRIPT %>
<% Dim sUrl,strdate,sHash %>
<HTML>
<SCRIPT LANGUAGE="JavaScript" RUNAT="Server" SRC="md5.js">
</SCRIPT>
<HEAD>
<%
Function FmHash(sDir,sObject,strdate) Dim iDate,sSecret
sUrl = ""
iDate = DateDiff("s", "01/01/1970 00:00:00", strdate)
sUrl = sDir&sObject&"?e="&iDate sSecret = Request.Form("secret") sHash = LCase
(hex_md5(sSecret&sUrl)) sUrl = sUrl&chr(38)&"h="&sHash
FmHash = sUrl
End Function
%>
</HEAD>
```

```
<BODY>
<FORM action="indie.asp" method="post">
<INPUT type="hidden" name="postBack" id="postBack" value="true">
<br>
<INPUT type="text" value="<%= Request.Form("secret")%>" name="secret">
<INPUT type="submit" name="update" value="Update">
</FORM>
<%
If Request.Form("postback") Then
Dim sFile, sServer, sPath, dtToday
sServer = "mms://indieweb.wmod.llnwd.net" sPath = "/a371/o2/S/"
sFile = "TEST.wmv"
dtToday = DateSerial(Year(Now), Month(Now), Day(Now)) Response.Write("Expired
Yesterday: <A href='" & sServer & FmHash(sPath,
sFile, DateAdd("d", -1, dtToday)) & "'>TEST.wmv</A>") Response.Write("<BR>")
Response.Write("Expires Tomorrow: <A href='" & sServer & "" & FmHash
(sPath,sFile, DateAdd("d", 1, dtToday)) & "'>TEST.wmv</A>")
End If
%>
</BODY>
</HTML>
```

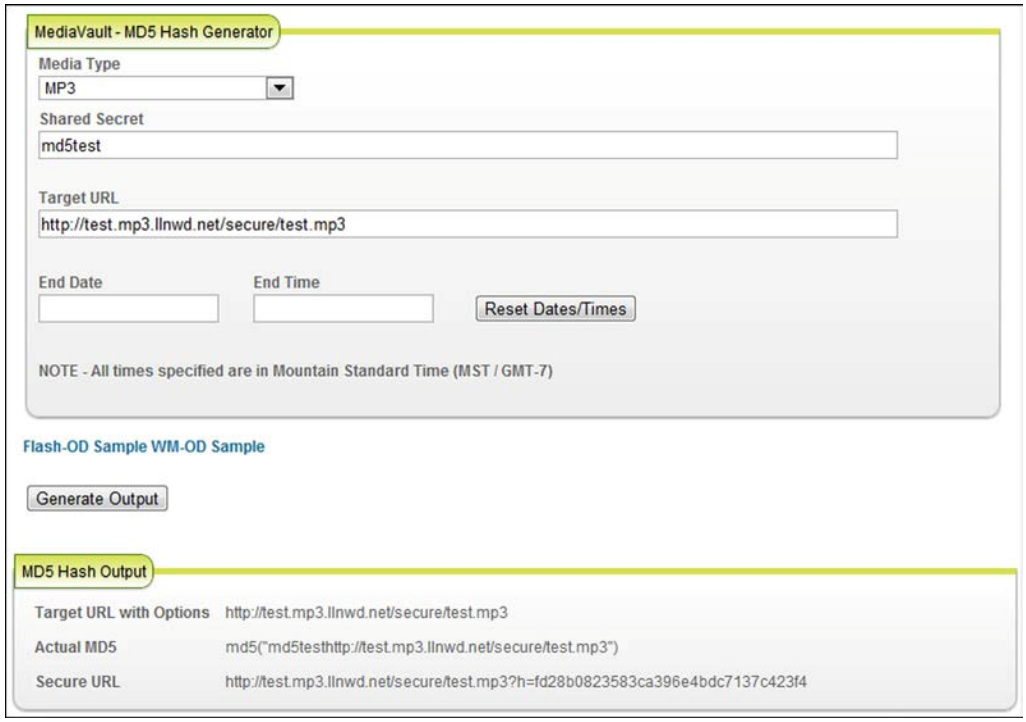
MediaVault for MP3

MediaVault for MP3 Parameters

The following parameters may be used with *MediaVault* for MP3 to generate a hash value:

End Time (e)

End Time Example



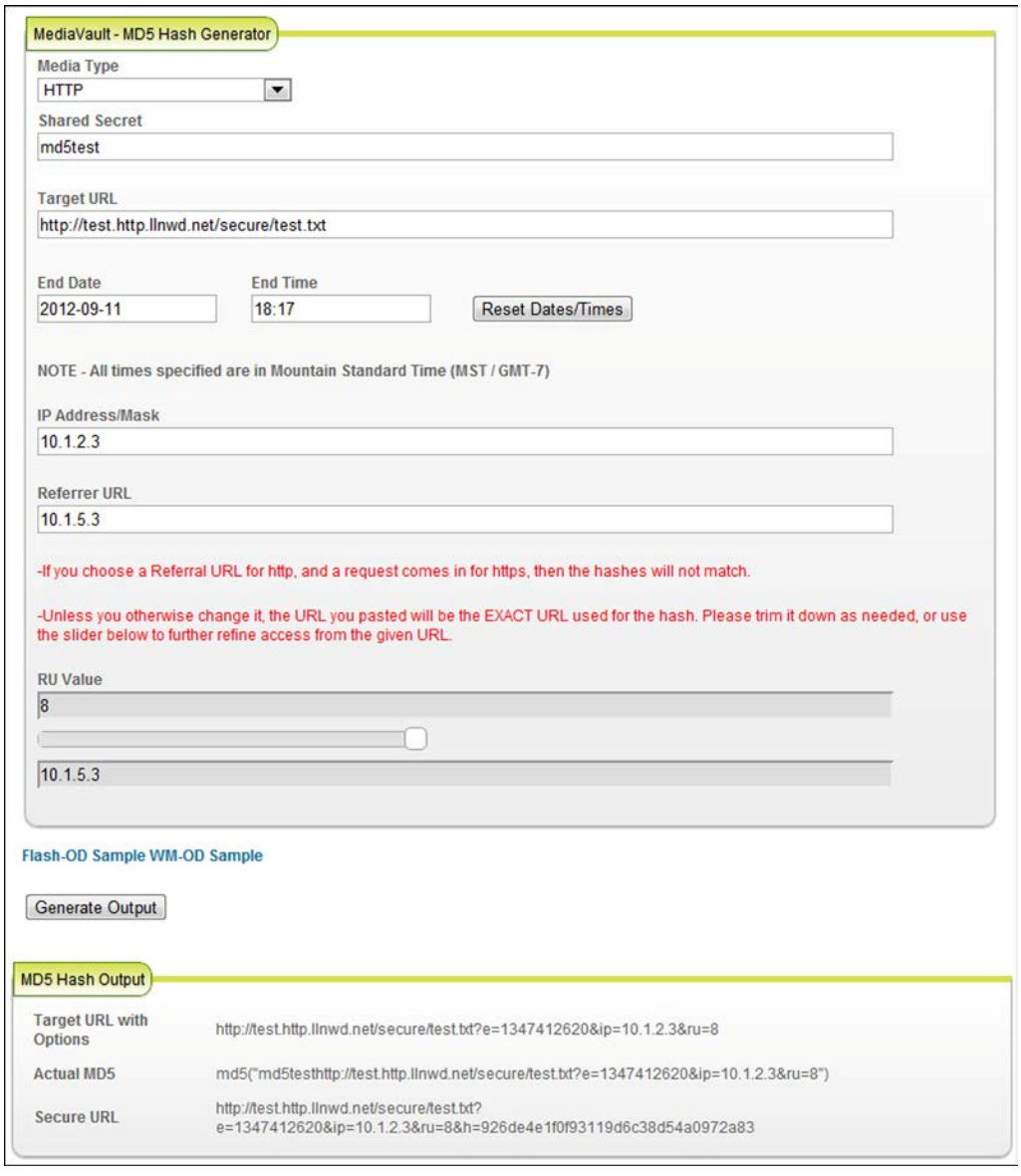
MediaVault for HTTP

MediaVault for HTTP Parameters

The following parameters may be used with *MediaVault* for HTTP to generate a hash value:

- End Time (e)
- IP Address/ (Range)Mask (ip)
- Referrer URL (ru)

End Time, IP Address and Referrer URL Example



.exe Files

When using *MediaVault*the file extension is stripped and the resulting file cannot be executed on your desktop (or wherever you save the file to). If this occurs, append the extension with a query string value at the end of the URL i.e., &file=.exe.

Example:

Non-executable File (URL with the file extension stripped from the end of the URL):

```
http://securehelixproducts.real.com/encoder/Producer13_1_1_
Setup.exe?e=1275930304&h=56ac8562b77f7f75e944cab4037143d9
```

Executable File (URL with the file extension (&file=.exe) appended to the end of the URL):

```
http://securehelixproducts.real.com/encoder/Producer13_1_1_
Setup.exe?e=1275930304&h=56ac8562b77f7f75e944cab4037143d9&file=.exe
```

Path-based

This feature allows you to use MediaVault to secure media streams via path-based tokenized support. Use this method to embed security tokens in the path for streaming media as an alternative to cross-site cookies to prevent browsers from interfering with the functionality.

For this feature, there are three rewrite options: `mv_path_based`, `mv_path_token`, and `mv_path_delim`.

Parameters

Rewrite Option	Description
`mv_path_based`	Enables path-based processing for the rewrite. While path-based is enabled, queryterm-based URLs will be accepted.
`mv_path_token`	Optional. Only needed to use a special token string. Default: token= Use these characters with the `mv_path_token` rewrite option: ['a-z'] ['A-Z'] ['0-9'] ['/'] ['='] A string length between 4 to 12 characters is recommended. Using a slash or multiple slashes as the only character in a token is invalid.
`mv_path_delim`	Optional. Only needed to use a special delimiter. Default: ~ Do not use these characters for the mv_path_delim rewrite option: [':'] ['/'] ['?'] ['#'] ['['] [']'] ['@'] ['!'] ['$'] ['&'] ['\''] ['('] [')'] ['*'] ['+'] [','] [';'] ['=']

Example

cdn_rewrite <published-url> <source-url> acct_id <ID> hashsecret <secret> mv_path_based [other options ...]

- After matching the rewrite, if the `mv_path_based` flag is set, EdgePrism will convert the request URL into the normal form.
- The MediaVault rewrite options (queryterm aliases, prefix, etc.) will behave as usual.
- The normal URL is used to construct the private/ origin request URL options to strip queryterms to work as expected.
- This option is not compatible with other MediaVault options that are doing path string matching or with `fallbackurl`.

Cookie-based

Edgio now provides a way of working with *MediaVault* content authorization parameters in the form of HTTP cookies. This new scheme places the parameters within HTTP cookies after the initial request. In addition, *MediaVault* provides for the management of sequences of requests by adding incremental expiration to the cookies.

> **Note:** This feature does not apply to any other services other than HTTP.

Parameters

The following parameters are available in *MediaVault* URLs and cookies when the `hashsecret_cookie` setting is enabled.

> **Note:** For customers with Configure access to *Content Delivery*, this setting will be visible in Control in the configuration for the associated *Content Delivery*Account, under Additional Options.

Parameters:

- **cf**: Final expiration. This is defined as the final time that content will be authorized. Expiration times, via the "e" parameter will be incremented in cookies until "cf" is reached.
- **cd**: Delta (in seconds) between the second request and all subsequent requests. In other words, the expiration time ("e" parameter) will be increased by this delta for the life of the content.
- **ci**: Initial delta (optional). The time allowed between the start of manifest file delivery and the next request.
- **cp**: Paths. A comma-separated list of paths (that can include wildcards) which are matched against the file path portion of an incoming request to determine if a URL is valid.

When this is not present, "cd" is used in its place. This parameter is never present in a cookie.

The following parameters may also be used with cookie-based *MediaVault* parameters, and will and maintain the same function as their URL-based counterparts.

- **e**: Expiration time. Should a request have an expiration time that is greater than the time a request is received, the request will be considered unauthorized. When cookies are generated, expiration time is either the sum of current_time + ci or current_time + cd. If ci is present, the first expiration time provided in a cookie will be the former.
- **p**: Prefix length: Cookie-based *MediaVault* hashes are computed on both portions of the request URL as well as the cookie string. This prefix specifies how much of the URL will be used for computing the MD5 digest. Should the "grab_range_from_url" rewrite option be specified and no prefix parameter be present, an implicit prefix up
- to but not including the first slash in a URL with the string "/range/" will be used. This prefix is also used when generating the Path= component of a *MediaVault* cookie.
- **ri**: Initial rate limit (bytes/sec)
- **rs**: Rate limit (kbytes/sec)
- **h**: MD5 hex digest of the URL and *MediaVault* parameters contained in the cookie. This works in the same way as current *MediaVault* hashes, except part of the MD5 is computed on the URL (up to the prefix if present) and the rest via the Cookie header sent from a client.

Example

The following sequence of events could take place between a client and a CDN server:

1) The client requests a manifest. The following URL is requested at time 1000000000.

```
http://vault.cx.rd.example.com/x/blah.m3u8?p=30&ci=20&cd=5&e=1000000010&cf=1000021610&h=4c66b14d48e9a42b8b10129aa0ae2799
```

This translates to a manifest file request with:

- **p: URL prefix of 30 characters (http://vault.cx.rd.example.com/x/).**
- **ci**: Initial delta of 20 seconds.
- **cd**: Delta of 5 seconds. This delta will be used in subsequent requests because an initial delta is present.
- **e**: Expiration time of 1000000010. The manifest request URL is valid until this time.
- **cf**: Final expiration time. Any requests related to this manifest must be received by the CDN server by this time.
- **h**: URL hash with secret, in this case:

```
md5("secret" + "http://vault.cx.rd.example.com/x/" +
"?p=30&ci=20&cd=5&e=1000000010&cf=1000021610")
```

2) Given that the request is received at a time earlier than "e", it is considered valid, and a manifest is returned along with a Set-Cookie header:

```
Set-Cookie:_llnw_
hashsecret=p=30&e=1000000020&cd=5&cf=1000021610&h=ed772eae4d592989bc53aed593a0dbcb;
Path=/x/; HttpOnly
```

This translates to:

- **p**: URL prefix of 30 characters (as above).
- **e**: A subsequent request must occur by time 1000000020. This is the arrival time (1000000000) + "ci",
- an initial delta of 20.
- **cd**: Delta of 5 seconds, this will be used on calculation of expiration in the next request.
- **cf**: Final expiration time (as above).
- **h**: MD5 hash:

```
md5("secret" + "http://published.url/x/" + "=p=23&e=1000000020&dcd=5&fcf=1000021610")
```

> **Note:** The "ci" parameter is not included in the cookie because it is only used when generating the expiration in the first one.

3) A subsequent client request would be without *MediaVault* parameters in the URL, but in a cookie. The following request arrives at time 1000000010:

http://example.cx.rd.example.com/x/video.ts/range/2000000-4000000

With the following cookie header (which was generated in step 2):

```
Cookie:_llnw_
hashsecret=p=30&e=1000000020&cd=5&cf=1000021610&h=ed772eae4d592989bc53aed593a0dbcb
```

4) Because the arrival time of the request is less than the expiration time specified in the cookie, *MediaVault* returns the content, and sets a subsequent cookie. The content is not considered expired because the new expiration time is still smaller than the final expiration.

```
Set-Cookie:_llnw_
hashsecret=p=30&e=1000000015&cd=5&cf=1000021610&h=9c44e95f38c41033147c367147cc5f2c;
Path=/x/; HttpOnly
```

This translates to:

- cd: Delta of 5 seconds, used to calculate "e" in this request.
- e: The end time is "cd" (5 seconds) added to the time the request was received, 1000000010, a total of 1000000015.
- h: The new hash:

```
md5("secret" + "http://vault.cx.rd.example.com/x/" +
"=p=30&e=1000000015&cd=5&cf=1000021610")
```

(Other parameters remain the same as above)

5) The client and *MediaVault* continue this process until content is no longer desired or time passes what is specified in the final expiration parameter.

Notes

- In the case of HTTPS requests, the cookie will have a "Secure" parameter appended to it.
- The Expiration field of the cookie will be 120 seconds after the expiration time listed in the "e" parameter of the authorization string.
- In the absence of a prefix (p=0), for cookie based requests, the entire URL along with the authorization string (up to the MD5 hash) will be used for computing hashes. The exception to this case is as noted above when a range is obtained from the URL path.
- The path field in the Set-Cookie header will be truncated by the prefix provided when valid. Should a prefix fall in the middle of a path, the path will be truncated to the leftmost slash relative to the prefix.
- It is not permitted to have a "ci" parameter without a "cd" parameter.
- In the case that both a URL and a cookie have *MediaVault* parameters, the parameters in the URL will be attempted first. Should authorization fail due to expiration (a "soft" failure), the cookie parameters will be attempted. If authorization fails on a URL due to a bad hash computation, a cookie will not be attempted if present.

Rewrite Options

Selective URL Handling

`hashsecret_queryterm_list <val_only> <order> <exclude> <qt_list>`

This option allows you to specify which queryterms are to be included in the authentication. The queryterms in the list will be used in the authentication if they are present in the URL. The list does not indicate required queryterms that must be present in the URL. The option has flags to control its behavior.

<val_only>	flag	Determines if we are using just the queryterm value. -0 (zero) means we use the queryterm name and value in the calculation (like we do today). -1 (one) means we just use the queryterm values.
<order>	flag	Determines the order that queryterms are used in the hash calculation. - 0 (zero) means that we'll use the queryterms in the order we find them in the request URL. -1 (one) means we'll use the queryterms in the order they are in the <qt_list> of this option. Note: this flag is not currently implemented (queryterms will be used in the order they are found in the request URL). This flag should be set to zero.
<exclude>	flag	Indicates whether this list of queryterms are to be included or excluded from the hash calculation. -0 (zero) means that the queryterms are to be included in the hash calculation; queryterms not in the list are ignored. -1 (one) means that the queryterms are to be excluded from the hash calculation; only queryterms not in this list will be included in the calculation. Note: this flag is not currently implemented (the queryterms in the list will be included in the hash calculation). This flag should be set to zero.
<qt_list>	comma separated list of queryterms	To be used in the calculation. It is important to note that these are the actual queryterm names as they will appear in the request URL.

Example 1

```
hashsecret_queryterm_list 0 0 0 qt1,qt2
calculated as: <url>?qt1=qt1_val&qt2=qt2_val
```

Example 2:

```
hashsecret_queryterm_list 1 0 0 qt1,qt2
calculated as: <url><qt1_val><qt2_val>
```

Note that when doing values only, the values are packed together without a '&' between and the trailing '?' is not used.