

# Contents

[Content Delivery Network documentation](#)

[Overview](#)

[What is Azure Content Delivery Network \(Azure CDN\)?](#)

[Compare product features](#)

[Quickstarts](#)

[Create a profile and endpoint](#)

[Integrate a storage account](#)

[Create a profile by using an Azure Resource Manager template](#)

[Tutorials](#)

[Add Azure CDN to a web app](#)

[Access blobs by using a custom domain over HTTPS](#)

[Add a custom domain](#)

[Configure HTTPS on a custom domain](#)

[Set caching rules](#)

[Enforce HTTPS by using the Standard rules engine](#)

[Samples](#)

[Code samples](#)

[Concepts](#)

[How caching works](#)

[China content delivery](#)

[HTTP/2 support](#)

[Retrieve POP IP's](#)

[Migrate from Standard Verizon to Premium Verizon](#)

[DDoS protection](#)

[How-to guides](#)

[Optimize content](#)

[Large-file optimization](#)

[Media streaming optimization](#)

[Dynamic site acceleration](#)

## Manage

- Create an endpoint
- Manage by using Azure PowerShell
- Restrict access by country or region
- Improve performance by compressing files
- Control caching behavior
  - Control caching behavior by using caching rules
  - Cache content by query strings
    - Standard tier
    - Premium tier
  - Purge cached assets
  - Preload cached assets
- Configure time to live (TTL)
  - Azure web content
  - Azure Blob storage
- Token authentication
- Shared access signature storage support
- Cross-origin resource sharing
- Monitor resources
- Use rules to override behavior
- Use Traffic Manager

## Analyze

- Azure diagnostics logs
- Analytics tools for Azure CDN from Verizon
  - Core reports from Verizon
  - Custom reports from Verizon
- Analytics tools for Premium Azure CDN from Verizon
  - Generate advanced HTTP reports
  - View real-time statistics
  - Analyze edge node performance
  - Get real-time alerts

## Develop

- .NET

[Node.js](#)

[Troubleshoot](#)

[404 status](#)

[File compression](#)

[Allowed certificate authorities](#)

[Reference](#)

[Standard rules engine reference](#)

[Standard rules engine match conditions](#)

[Standard rules engine actions](#)

[Verizon Premium rules engine reference](#)

[Rules engine conditional expressions](#)

[Rules engine match conditions](#)

[Rules engine features](#)

[HTTP variables](#)

[Verizon-specific HTTP headers](#)

[X-EC-Debug headers](#)

[Microsoft-specific HTTP headers](#)

[Azure PowerShell](#)

[.NET](#)

[Java](#)

[REST](#)

[Resources](#)

[Azure CDN POP locations](#)

[POP locations by region](#)

[POP locations by abbreviation](#)

[Azure Content Delivery Network billing](#)

[Azure Roadmap](#)

[MSDN forum](#)

[Pricing](#)

[Pricing calculator](#)

[Service updates](#)

[Stack Overflow](#)

## Videos

# What is a content delivery network on Azure?

1/3/2020 • 3 minutes to read • [Edit Online](#)

A content delivery network (CDN) is a distributed network of servers that can efficiently deliver web content to users. CDNs store cached content on edge servers in point-of-presence (POP) locations that are close to end users, to minimize latency.

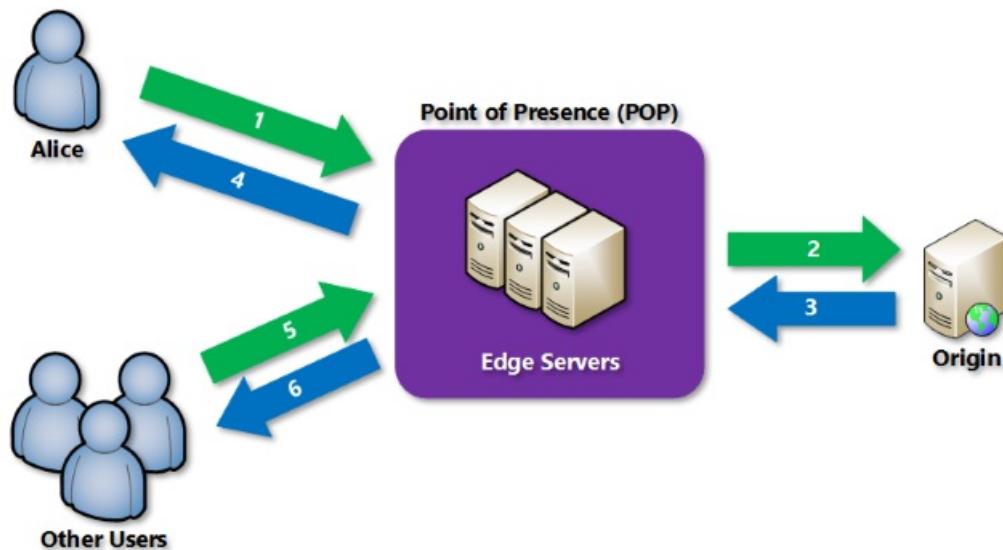
Azure Content Delivery Network (CDN) offers developers a global solution for rapidly delivering high-bandwidth content to users by caching their content at strategically placed physical nodes across the world. Azure CDN can also accelerate dynamic content, which cannot be cached, by leveraging various network optimizations using CDN POPs. For example, route optimization to bypass Border Gateway Protocol (BGP).

The benefits of using Azure CDN to deliver web site assets include:

- Better performance and improved user experience for end users, especially when using applications in which multiple round-trips are required to load content.
- Large scaling to better handle instantaneous high loads, such as the start of a product launch event.
- Distribution of user requests and serving of content directly from edge servers so that less traffic is sent to the origin server.

For a list of current CDN node locations, see [Azure CDN POP locations](#).

## How it works



1. A user (Alice) requests a file (also called an asset) by using a URL with a special domain name, such as <endpoint name>.azureedge.net. This name can be an endpoint hostname or a custom domain. The DNS routes the request to the best performing POP location, which is usually the POP that is geographically closest to the user.
2. If no edge servers in the POP have the file in their cache, the POP requests the file from the origin server. The origin server can be an Azure Web App, Azure Cloud Service, Azure Storage account, or any publicly accessible web server.
3. The origin server returns the file to an edge server in the POP.
4. An edge server in the POP caches the file and returns the file to the original requestor (Alice). The file

remains cached on the edge server in the POP until the time-to-live (TTL) specified by its HTTP headers expires. If the origin server didn't specify a TTL, the default TTL is seven days.

5. Additional users can then request the same file by using the same URL that Alice used, and can also be directed to the same POP.
6. If the TTL for the file hasn't expired, the POP edge server returns the file directly from the cache. This process results in a faster, more responsive user experience.

## Requirements

To use Azure CDN, you must own at least one Azure subscription. You also need to create at least one CDN profile, which is a collection of CDN endpoints. Every CDN endpoint represents a specific configuration of content deliver behavior and access. To organize your CDN endpoints by internet domain, web application, or some other criteria, you can use multiple profiles. Because [Azure CDN pricing](#) is applied at the CDN profile level, you must create multiple CDN profiles if you want to use a mix of pricing tiers. For information about the Azure CDN billing structure, see [Understanding Azure CDN billing](#).

### Limitations

Each Azure subscription has default limits for the following resources:

- The number of CDN profiles that can be created.
- The number of endpoints that can be created in a CDN profile.
- The number of custom domains that can be mapped to an endpoint.

For more information about CDN subscription limits, see [CDN limits](#).

## Azure CDN features

Azure CDN offers the following key features:

- [Dynamic site acceleration](#)
- [CDN caching rules](#)
- [HTTPS custom domain support](#)
- [Azure diagnostics logs](#)
- [File compression](#)
- [Geo-filtering](#)

For a complete list of features that each Azure CDN product supports, see [Compare Azure CDN product features](#).

## Next steps

- To get started with CDN, see [Create an Azure CDN profile and endpoint](#).
- Manage your CDN endpoints through the [Microsoft Azure portal](#) or with [PowerShell](#).
- Learn how to automate Azure CDN with [.NET](#) or [Node.js](#).
- To see Azure CDN in action, watch the [Azure CDN videos](#).
- For information about the latest Azure CDN features, see [Azure CDN blog](#).

# Compare Azure CDN product features

11/20/2019 • 2 minutes to read • [Edit Online](#)

Azure Content Delivery Network (CDN) includes four products: **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Akamai**, **Azure CDN Standard from Verizon**, and **Azure CDN Premium from Verizon**. For information about migrating an **Azure CDN Standard from Verizon** profile to **Azure CDN Premium from Verizon**, see [Migrate an Azure CDN profile from Standard Verizon to Premium Verizon](#). Note that while there is an upgrade path from Standard Verizon to Premium Verizon, there is no conversion mechanism between other products at this time.

The following table compares the features available with each product.

Performance Features and Optimizations	Standard Microsoft	Standard Akamai	Standard Verizon	Premium Verizon
Dynamic site acceleration	Offered via <a href="#">Azure Front Door Service</a>	✓	✓	✓
Dynamic site acceleration - adaptive image compression		✓		
Dynamic site acceleration - object prefetch		✓		
General web delivery optimization	✓	✓, Select this optimization type if your average file size is smaller than 10 MB	✓	✓
Video streaming optimization	via General Web Delivery	✓	via General Web Delivery	via General Web Delivery
Large file optimization	via General Web Delivery	✓, Select this optimization type if your average file size is larger than 10 MB	via General Web Delivery	via General Web Delivery
Change optimization type		✓		
Origin Port	All TCP ports	<a href="#">Allowed origin ports</a>	All TCP ports	All TCP ports
Global server load balancing (GSLB)	✓	✓	✓	✓

Performance Features and Optimizations	Standard Microsoft	Standard Akamai	Standard Verizon	Premium Verizon
Fast purge	✓	✓, Purge all and Wildcard purge are not supported by Azure CDN from Akamai currently	✓	✓
Asset pre-loading			✓	✓
Cache/header settings (using caching rules)	✓ using Standard rules engine	✓	✓	
Customizable, rules based content delivery engine	✓ using Standard rules engine			✓ using rules engine
Cache/header settings	✓ using Standard rules engine			✓ using Premium rules engine
URL redirect/rewrite	✓ using Standard rules engine			✓ using Premium rules engine
Mobile device rules	✓ using Standard rules engine			✓ using Premium rules engine
Query string caching	✓	✓	✓	✓
IPv4/IPv6 dual-stack	✓	✓	✓	✓
HTTP/2 support	✓	✓	✓	✓
Security	Standard Microsoft	Standard Akamai	Standard Verizon	Premium Verizon
HTTPS support with CDN endpoint	✓	✓	✓	✓
Custom domain HTTPS	✓	✓, Requires direct CNAME to enable	✓	✓
Custom domain name support	✓	✓	✓	✓
Geo-filtering	✓	✓	✓	✓
Token authentication				✓
DDOS protection	✓	✓	✓	✓
Bring your own certificate	✓		✓	✓

Performance features and optimizations	Standard Microsoft	Standard Akamai	Standard Verizon	Premium Verizon
<b>Analytics and reporting</b>	<b>Standard Microsoft</b>	<b>Standard Akamai</b>	<b>Standard Verizon</b>	<b>Premium Verizon</b>
Azure diagnostic logs	✓	✓	✓	✓
Core reports from Verizon			✓	✓
Custom reports from Verizon			✓	✓
Advanced HTTP reports				✓
Real-time stats				✓
Edge node performance				✓
Real-time alerts				✓
<b>Ease of use</b>	<b>Standard Microsoft</b>	<b>Standard Akamai</b>	<b>Standard Verizon</b>	<b>Premium Verizon</b>
Easy integration with Azure services, such as <a href="#">Storage</a> , <a href="#">Web Apps</a> , and <a href="#">Media Services</a>	✓	✓	✓	✓
Management via <a href="#">REST API</a> , <a href="#">.NET</a> , <a href="#">Node.js</a> , or <a href="#">PowerShell</a>	✓	✓	✓	✓
Compression MIME types	Default only	Configurable	Configurable	Configurable
Compression encodings	gzip, brotli	gzip	gzip, deflate, bzip2, brotli	gzip, deflate, bzip2, brotli

# Quickstart: Create an Azure CDN profile and endpoint

7/5/2019 • 3 minutes to read • [Edit Online](#)

In this quickstart, you enable Azure Content Delivery Network (CDN) by creating a new CDN profile and CDN endpoint. After you have created a profile and an endpoint, you can start delivering content to your customers.

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Prerequisites

For the purposes of this quickstart, you must have created a storage account named *mystorageacct123*, which you use for the origin hostname. For more information, see [Integrate an Azure storage account with Azure CDN](#).

## Log in to the Azure portal

Log in to the [Azure portal](#) with your Azure account.

## Create a new CDN profile

A CDN profile is a container for CDN endpoints and specifies a pricing tier.

1. In the Azure portal, in the upper left, select **Create a resource**.

The **New** pane appears.

2. Select **Web + Mobile**, then select **CDN**.

The screenshot shows the Microsoft Azure portal interface. In the top left, there's a 'Create a resource' button with a red box around it. The main area is titled 'New' and shows the 'Azure Marketplace'. Under 'Featured', there are several service categories: 'Web App' (with a globe icon), 'Mobile App' (with a smartphone icon), 'Logic App' (with a person icon), 'Web App for Containers' (with a cloud icon), 'CDN' (with a cloud icon, highlighted by a red box), and 'Media Services' (with a camera icon). On the left sidebar, under 'All services', various Azure services are listed: Dashboard, All resources, Resource groups, App Services, Function Apps, SQL databases, Azure Cosmos DB, Virtual machines, Load balancers, Storage accounts, and Virtual networks.

The **CDN profile** pane appears.

- For the CDN profile settings, use the values specified in the following table:

SETTING	VALUE
<b>Name</b>	Enter <i>my-cdn-profile-123</i> for your profile name. This name must be globally unique; if it is already in use, you may enter a different name.
<b>Subscription</b>	Select an Azure subscription from the drop-down list.
<b>Resource group</b>	Select <b>Create new</b> and enter <i>my-resource-group-123</i> for your resource group name. If it is already in use, you may enter a different name or you can select <b>Use existing</b> and select <b>my-resource-group-123</b> from the drop-down list.
<b>Resource group location</b>	Select <b>Central US</b> from the drop-down list.
<b>Pricing tier</b>	Select <b>Standard Verizon</b> from the drop-down list.
<b>Create a new CDN endpoint now</b>	Leave unselected.

Home > New > CDN profile

**CDN profile**

\* Name  
my-cdn-profile-123

\* Subscription  
<subscription name>

\* Resource group  
 Create new  Use existing  
my-resource-group-123

\* Resource group location ⓘ  
Central US

\* Pricing tier (View full pricing details)  
Standard Verizon

Create a new CDN endpoint now

Pin to dashboard

**Create**   [Automation options](#)

4. Select **Pin to dashboard** to save the profile to your dashboard after it is created.

5. Select **Create** to create the profile.

For **Azure CDN Standard from Microsoft** profiles only, profile completion usually completes in two hours.

## Create a new CDN endpoint

After you've created a CDN profile, you can use it to create an endpoint.

1. In the Azure portal, select in your dashboard the CDN profile that you created. If you can't find it, select **All services**, then select **CDN profiles**. In the **CDN profiles** page, select the profile that you want to use.

The CDN profile page appears.

2. Select **Endpoint**.

Essentials	
Resource group (change) <a href="#">my-resource-group-123</a>	Pricing tier Standard Verizon
Status Active	
Location Central US	
Subscription name (change) <subscription name>	
Subscription ID <subscription ID>	

The **Add an endpoint** pane appears.

3. For the endpoint settings, use the values specified in the following table:

SETTING	VALUE
<b>Name</b>	Enter <i>my-endpoint-123</i> for your endpoint hostname. This name must be globally unique; if it is already in use, you may enter a different name. This name is used to access your cached resources at the domain < <i>endpoint name</i> >.azureedge.net.
<b>Origin type</b>	Select <b>Storage</b> .
<b>Origin hostname</b>	Enter <i>mystorageacct123.blob.core.windows.net</i> for your hostname. This name must be globally unique; if it is already in use, you may enter a different name.
<b>Origin path</b>	Leave blank.
<b>Origin host header</b>	Leave the default generated value.
<b>Protocol</b>	Leave the default <b>HTTP</b> and <b>HTTPS</b> options selected.
<b>Origin port</b>	Leave the default port values.
<b>Optimized for</b>	Leave the default selection, <b>General web delivery</b> .

Add an endpoint □ X

Allows configuring content delivery behavior an...

\* Name  .azureedge.net

\* Origin type

\* Origin hostname

Origin path

Origin host header

Protocol   Origin port

Optimized for

Add Automation options

4. Select **Add** to create the new endpoint.

After the endpoint is created, it appears in the list of endpoints for the profile.

The screenshot shows the Azure portal interface for managing a CDN profile. At the top, there are navigation links: Endpoint, Manage, Purge, Move, and Delete. A success message is displayed: "Successfully created endpoint 'my-endpoint-123'". Below this, the "Essentials" section provides details about the endpoint, including its resource group ("my-resource-group-123"), pricing tier ("Standard Verizon"), status ("Active"), location ("Central US"), subscription name ("

Because it takes time for the registration to propagate, the endpoint isn't immediately available for use:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes within 90 minutes.

## Clean up resources

In the preceding steps, you created a CDN profile and an endpoint in a resource group. Save these resources if you want to go to [Next steps](#) and learn how to add a custom domain to your endpoint. However, if you don't expect to use these resources in the future, you can delete them by deleting the resource group, thus avoiding additional charges:

1. From the left-hand menu in the Azure portal, select **Resource groups** and then select **my-resource-group-123**.
2. On the **Resource group** page, select **Delete resource group**, enter *my-resource-group-123* in the text box, then select **Delete**.

This action will delete the resource group, profile, and endpoint that you created in this quickstart.

## Next steps

To learn about adding a custom domain to your CDN endpoint, see the following tutorial:

[Tutorial: Add a custom domain to your Azure CDN endpoint](#)

# Quickstart: Integrate an Azure storage account with Azure CDN

12/23/2019 • 5 minutes to read • [Edit Online](#)

In this quickstart, you enable [Azure Content Delivery Network \(CDN\)](#) to cache content from Azure storage. Azure CDN offers developers a global solution for delivering high-bandwidth content. It can cache blobs and static content of compute instances at physical nodes in the United States, Europe, Asia, Australia, and South America.

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Log in to the Azure portal

Log in to the [Azure portal](#) with your Azure account.

## Create a storage account

Use the following procedure to create a new storage account for an Azure subscription. A storage account gives access to Azure Storage services. The storage account represents the highest level of the namespace for accessing each of the Azure Storage service components: Azure Blob, Queue, and Table storage. For more information, see [Introduction to Microsoft Azure Storage](#).

To create a storage account, you must be either the service administrator or a coadministrator for the associated subscription.

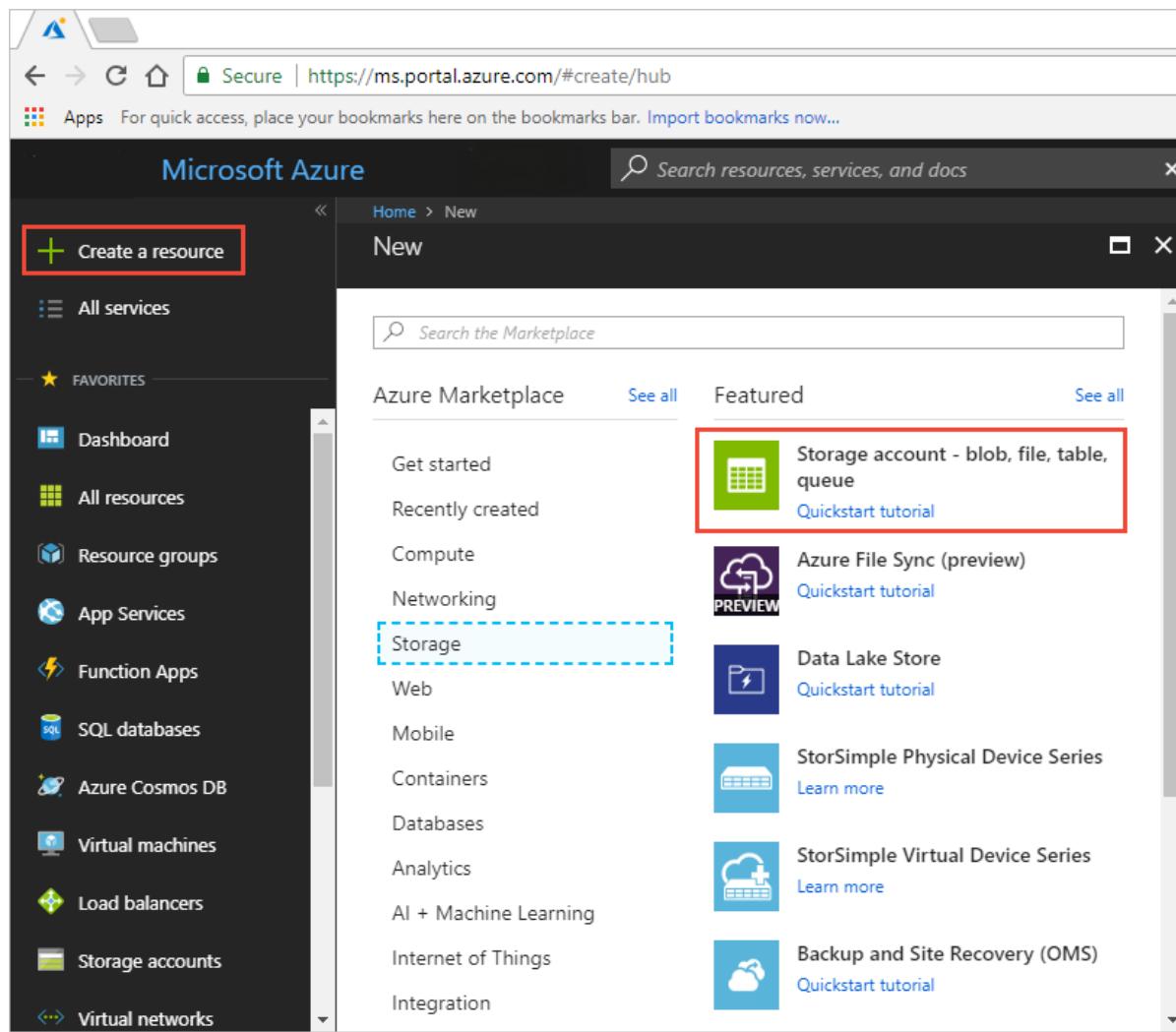
You can use several methods to create a storage account, including the Azure portal and PowerShell. This quickstart demonstrates how to use the Azure portal.

### To create a storage account for an Azure subscription

1. In the Azure portal, in the upper left, select **Create a resource**.

The **New** pane appears.

2. Select **Storage**, then select **Storage account - blob, file, table, queue**.



The **Create storage account** pane appears.

The cost of your storage account depends on the usage and the options you choose below.  
[Learn more](#)

\* Name  .core.windows.net

Deployment model  Resource manager  Classic

Account kind  Storage (general purpose v1)  Storage (general purpose v2)

\* Location  Central US  East US  West Europe  West US  South Central US  South US  North Europe  East Asia  West Asia

Replication  Locally-redundant storage (LRS)  Zoned redundant storage (ZRS)

Performance  Standard  Premium

\* Secure transfer required  Disabled  Enabled

\* Subscription

\* Resource group  Create new  Use existing

Pin to dashboard

**Create** [Automation options](#)

- In the **Name** box, enter a subdomain name. This entry can contain 3-24 lowercase letters and numbers.

This value becomes the host name within the URI that's used to address blob, queue, or table resources for the subscription. To address a container resource in Blob storage, use a URI in the following format:

`http://<StorageAccountLabel>.blob.core.windows.net/<mycontainer>`

where `<StorageAccountLabel>` refers to the value you entered in the **Name** box.

#### IMPORTANT

The URL label forms the subdomain of the storage account URI and must be unique among all hosted services in Azure.

This value is also used as the name of the storage account in the portal or when you're accessing this account programmatically.

- For the remainder of the settings, use the values specified in the following table:

SETTING	VALUE
<b>Deployment model</b>	Use the default value.
<b>Account kind</b>	Use the default value.

SETTING	VALUE
<b>Location</b>	Select <b>Central US</b> from the drop-down list.
<b>Replication</b>	Use the default value.
<b>Performance</b>	Use the default value.
<b>Secure transfer required</b>	Use the default value.
<b>Subscription</b>	Select an Azure subscription from the drop-down list.
<b>Resource group</b>	Select <b>Create new</b> and enter <i>my-resource-group-123</i> for your resource group name. This name must be globally unique. If it is already in use, you may enter a different name or you can select <b>Use existing</b> and select <b>my-resource-group-123</b> from the drop-down list. For information about resource groups, see <a href="#">Azure Resource Manager overview</a> .
<b>Configure virtual networks</b>	Use the default value.

5. Select **Pin to dashboard** to save the storage account to your dashboard after it is created.
6. Select **Create**. Creating the storage account might take several minutes to finish.

## Enable Azure CDN for the storage account

You can enable Azure CDN for your storage account directly from your storage account. If you want to specify advanced configuration settings for your CDN endpoint, such as [large file download optimization](#), you can instead use the [Azure CDN extension](#) to create a CDN profile and endpoint.

1. Select a storage account from the dashboard, then select **Azure CDN** from the left pane. If the **Azure CDN** button is not immediately visible, you can enter CDN in the **Search** box of the left pane to find it.

The **Azure CDN** page appears.

The screenshot shows the 'New endpoint' configuration page. It includes fields for creating a new CDN profile, selecting a pricing tier, naming the endpoint, specifying the origin hostname, and a 'Create' button at the bottom.

Setting	Value
CDN profile	<input checked="" type="radio"/> Create new
Pricing tier	<input type="text"/>
CDN endpoint name	<input type="text"/> .azureedge.net
Origin hostname	<input type="text"/> 091784eastus.blob.core.windows.net

2. Create a new endpoint by entering the required information specified in the following table:

SETTING	VALUE
<b>CDN profile</b>	Select <b>Create new</b> and enter your profile name, for example, <i>my-cdn-profile-123</i> . This name must be globally unique.
<b>Pricing tier</b>	Select <b>Standard Verizon</b> from the drop-down list.
<b>CDN endpoint name</b>	Enter your endpoint hostname, i.e. <i>my-endpoint-123</i> . This name must be globally unique. This name is used to access your cached resources at the domain <endpoint name>.azureedge.net.
<b>Origin hostname</b>	By default, a new CDN endpoint uses the hostname of your storage account as the origin server.

3. Select **Create**. After the endpoint is created, it appears in the endpoint list.

The screenshot shows the Azure CDN management interface. At the top, there's a header bar with the title 'Azure CDN'. Below it, a main section titled 'Azure Content Delivery Network' provides a brief overview of what CDN does. Underneath, a table lists existing endpoints. One endpoint, 'dustydog.azureedge.net', is highlighted with a red box around its row. The table columns are 'HOSTNAME', 'STATUS', and 'PROTOCOL'. The 'HOSTNAME' column shows 'dustydog.azureedge.net', the 'STATUS' column shows 'Running' with a green checkmark, and the 'PROTOCOL' column shows 'HTTP, HTTPS'. Below the table, a new endpoint creation form is displayed. It includes fields for 'CDN profile' (radio buttons for 'Create new' and 'Use existing'), 'Pricing tier' (dropdown menu), 'CDN endpoint name' (text input field containing '.azureedge.net'), and 'Origin hostname' (text input field containing 'dustydogwebapp.azurewebsites.net'). A large blue 'Create' button is at the bottom of the form.

## Enable additional CDN features

From the storage account **Azure CDN** page, select the CDN endpoint from the list to open the CDN endpoint configuration page. From this page, you can enable additional CDN features for your delivery, such as [compression](#), [query string caching](#), and [geo filtering](#).

## Enable SAS

If you want to grant limited access to private storage containers, you can use the Shared Access Signature (SAS) feature of your Azure storage account. A SAS is a URI that grants restricted access rights to your Azure Storage resources without exposing your account key. For more information, see [Using Azure CDN with SAS](#).

## Access CDN content

To access cached content on the CDN, use the CDN URL provided in the portal. The address for a cached blob has the following format:

`http://<EndpointName>.azureedge.net/<myPublicContainer>/<BlobName>`

### NOTE

After you enable Azure CDN access to a storage account, all publicly available objects are eligible for CDN POP caching. If you modify an object that's currently cached in the CDN, the new content will not be available via Azure CDN until Azure CDN refreshes its content after the time-to-live period for the cached content expires.

## Remove content from Azure CDN

If you no longer want to cache an object in Azure CDN, you can take one of the following steps:

- Make the container private instead of public. For more information, see [Manage anonymous read access to containers and blobs](#).
- Disable or delete the CDN endpoint by using the Azure portal.
- Modify your hosted service to no longer respond to requests for the object.

An object that's already cached in Azure CDN remains cached until the time-to-live period for the object expires or until the endpoint is [purged](#). When the time-to-live period expires, Azure CDN determines whether the CDN endpoint is still valid and the object is still anonymously accessible. If they are not, the object will no longer be cached.

## Clean up resources

In the preceding steps, you created a CDN profile and an endpoint in a resource group. Save these resources if you want to go to [Next steps](#) and learn how to add a custom domain to your endpoint. However, if you don't expect to use these resources in the future, you can delete them by deleting the resource group, thus avoiding

additional charges:

1. From the left-hand menu in the Azure portal, select **Resource groups** and then select **my-resource-group-123**.
2. On the **Resource group** page, select **Delete resource group**, enter *my-resource-group-123* in the text box, then select **Delete**.

This action will delete the resource group, profile, and endpoint that you created in this quickstart.

3. To delete your storage account, select it from the dashboard, then select **Delete** from the top menu.

## Next steps

To learn about adding a custom domain and enable HTTPS on your CDN endpoint, see the following tutorial:

[Tutorial: Access storage blobs using an Azure CDN custom domain over HTTPS](#)

# Quickstart: Create an Azure CDN profile and endpoint using Resource Manager template

11/14/2019 • 2 minutes to read • [Edit Online](#)

In this quickstart, you deploy an Azure Resource Manager template using CLI. The template that you create deploys a CDN profile and CDN endpoint to front your web application. It should take about ten minutes to complete these steps.

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Prerequisites

For the purposes of this quickstart, you must have a Web Application to use as your Origin. The example Web Application used in this quickstart was deployed to <https://cdndemo.azurewebsites.net>

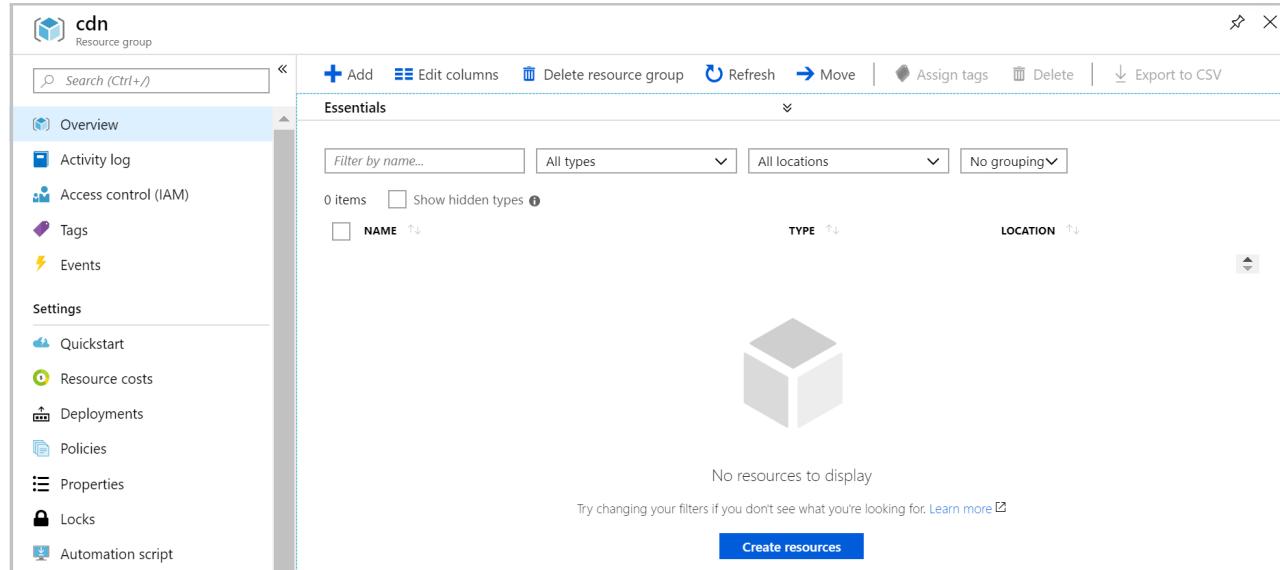
For more information, see [Create a static HTML web app in Azure](#).

## Create a resource group

All resources must be deployed in the same resource group.

Create the resource group in the location that you select. This example shows the creation of a resource group named cdn in the East US location.

```
az group create --name cdn --location eastus
```



## Create the Resource Manager template

In this step, you create a template file that deploys the resources.

While this example walks through a General Website Acceleration scenario, there are many other settings that can be configured. These settings are available in the Azure Resource Manager template reference. Please see references for [CDN Profile](#) and [CDN Profile Endpoint](#).

Note that Microsoft CDN doesn't support modifying the content type list.

Save the template as **resource-manager-cdn.json**.

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "cdnProfileSku": {  
            "type": "string",  
            "allowedValues": [  
                "Standard_Microsoft",  
                "Standard_Akamai",  
                "Standard_Verizon",  
                "Premium_Verizon"  
            ]  
        },  
        "endpointOriginHostName": {  
            "type": "string"  
        }  
    },  
    "variables": {  
        "profile": {  
            "name": "[replace(toLower(parameters('cdnProfileSku')), '_', '-')]"  
        },  
        "endpoint": {  
            "name": "[replace(toLower(parameters('endpointOriginHostName')), '.', '-')]",  
            "originHostName": "[parameters('endpointOriginHostName')]"  
        }  
    },  
    "resources": [  
        {  
            "type": "Microsoft.Cdn/profiles",  
            "apiVersion": "2017-10-12",  
            "location": "[resourceGroup().location]",  
            "name": "[variables('profile').name]",  
            "sku": {  
                "name": "[parameters('cdnProfileSku')]"  
            }  
        },  
        {  
            "dependsOn": [  
                "[resourceId('Microsoft.Cdn/profiles', variables('profile').name)]"  
            ],  
            "type": "Microsoft.Cdn/profiles/endpoints",  
            "apiVersion": "2017-10-12",  
            "location": "[resourceGroup().location]",  
            "name": "[concat(variables('profile').name, '/', variables('endpoint').name)]",  
            "properties": {  
                "hostName": "[concat(variables('endpoint').name, '.azureedge.net')]",  
                "originHostHeader": "[variables('endpoint').originHostName]",  
                "isHttpAllowed": true,  
                "isHttpsAllowed": true,  
                "queryStringCachingBehavior": "IgnoreQueryString",  
                "origins": [  
                    {  
                        "name": "[replace(variables('endpoint').originHostName, '.', '-')]",  
                        "properties": {  
                            "hostName": "[variables('endpoint').originHostName]",  
                            "httpPort": 80,  
                            "httpsPort": 443  
                        }  
                    }  
                ],  
                "contentTypesToCompress": [  
                    "application/eot",  
                    "application/font",  
                    "application/font-sfnt"  
                ]  
            }  
        }  
    ]  
}
```

```

        "application/x-7z-compressed",
        "application/x-cab",
        "application/x-debet",
        "application/x-dosexec",
        "application/x-executable",
        "application/x-freelander",
        "application/x-gnutar",
        "application/x-hx",
        "application/x-javascript",
        "application/x-macbinary",
        "application/x-macinfo",
        "application/x-macpowerpc",
        "application/x-macresource",
        "application/x-macwriter",
        "application/x-mpegurl",
        "application/x-opentype",
        "application/x-otf",
        "application/x-perl",
        "application/x-ttf",
        "font/eot",
        "font/ttf",
        "font/otf",
        "font/opentype",
        "image/svg+xml",
        "text/css",
        "text/csv",
        "text/html",
        "text/javascript",
        "text/js",
        "text/plain",
        "text/richtext",
        "text/tab-separated-values",
        "text/xml",
        "text/x-script",
        "text/x-component",
        "text/x-java-source"
    ],
    "isCompressionEnabled": true,
    "optimizationType": "GeneralWebDelivery"
}
],
"outputs": {
    "cdnUrl": {
        "type": "string",
        "value": "[concat('https://', variables('endpoint').name, '.azureedge.net')]"
    }
}
}

```

## Create the resources

Deploy the template using Azure CLI. You will be prompted for 2 inputs:

**cdnProfileSku** - the CDN provider that you want to use. The options are:

- Standard\_Microsoft
- Standard\_Akamai
- Standard\_Verizon
- Premium\_Verizon.

**endpointOriginHostName** - the endpoint that will be served through the CDN, for example,

cdndemo.azurewebsites.net.

```
az group deployment create --resource-group cdn --template-file arm-cdn.json
```

```
Bash    v | ⚡ ? 🌐 🎯 { }

senthuran@Azure:~$ az group deployment create --resource-group cdn --template-file arm-cdn.json
Please provide string value for 'cdnProfileSku' (? for help):
[1] Standard_Microsoft
[2] Standard_Akamai
[3] Standard_Verizon
[4] Premium_Verizon
Please enter a choice [1]: 1
Please provide string value for 'endpointOriginHostName' (? for help): cdndemo.azurewebsites.net
└- Running ..
```

## View the CDN profile

```
az cdn profile list --resource-group cdn -o table
```

```
Bash    v | ⚡ ? 🌐 🎯 { }

senthuran@Azure:~$ az cdn profile list --resource-group cdn -o table
Location      Name          ProvisioningState  ResourceGroup  ResourceState
-----      -----          -----          -----          -----
EastUs        standard-microsoft  Succeeded       cdn           Active
senthuran@Azure:~$
```

## View the CDN Endpoint for the profile standard-microsoft

```
az cdn endpoint list --profile-name standard-microsoft --resource-group cdn -o table
```

```
Bash    v | ⚡ ? 🌐 🎯 { }

senthuran@Azure:~$ az cdn endpoint list --profile-name standard-microsoft --resource-group cdn -o table
HostName          IsCompressionEnabled  IsHttpAllowed  IsHttpsAllowed  Location  Name          ResourceState
optimizationType  OriginHostHeader     ProvisioningState  QueryStringCachingBehavior  ResourceGroup  ResourceState
-----          -----          -----          -----          -----          -----          -----
cdndemo-azurewebsites-net.azureedge.net  True            True            True            EastUs    cdndemo-azurewebsites-net  G
eneralWebDelivery  cdndemo.azurewebsites.net  Succeeded      IgnoreQueryString  cdn           Running
senthuran@Azure:~$
```

Use the HostName to view the content. For example, access <https://cdndemo-azurewebsites-net.azureedge.net> using your browser.

## Clean up

Deleting the resource group will automatically remove all of the resources that were deployed in it.

```
az group delete --name cdn
```

```
Bash      ▾ | ⌂ ? ⚙ ⌛ ⌜ ⌜ { }

senthuran@Azure:~$ az group delete --name cdn
Are you sure you want to perform this operation? (y/n): y
[- Running ..
```

## References

- CDN Profile - [Azure Resource Manager Template Reference](#)
- CDN Endpoint - [Azure Resource Manager Template Reference Documentation](#)

## Next steps

To learn about adding a custom domain to your CDN endpoint, see the following tutorial:

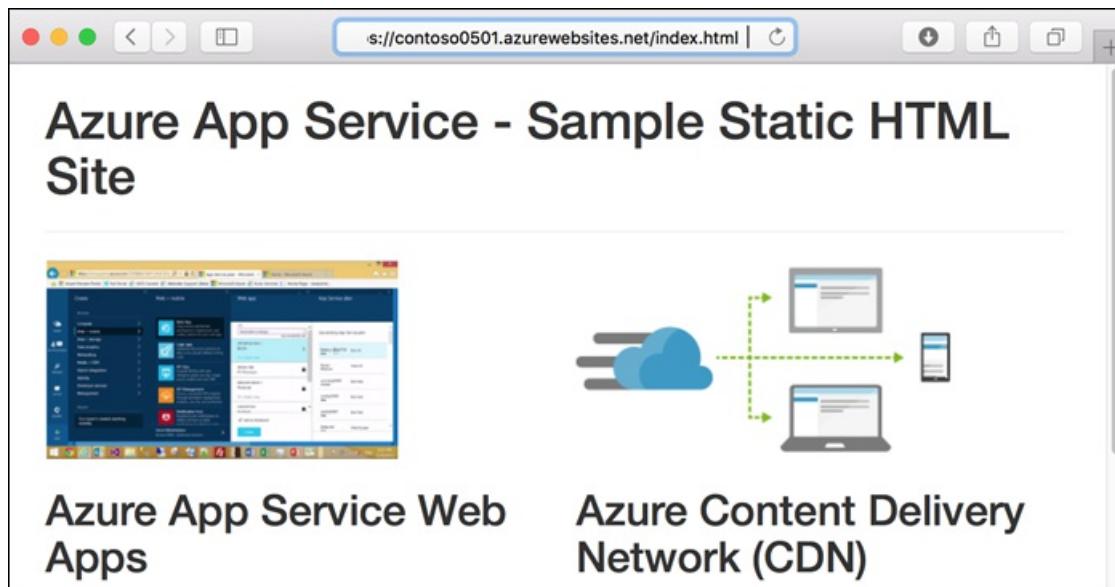
[Tutorial: Add a custom domain to your Azure CDN endpoint](#)

# Tutorial: Add Azure CDN to an Azure App Service web app

7/5/2019 • 6 minutes to read • [Edit Online](#)

This tutorial shows how to add [Azure Content Delivery Network \(CDN\)](#) to a [web app](#) in [Azure App Service](#). Web apps is a service for hosting web applications, REST APIs, and mobile back ends.

Here's the home page of the sample static HTML site that you'll work with:



What you'll learn:

- Create a CDN endpoint.
- Refresh cached assets.
- Use query strings to control cached versions.

## Prerequisites

To complete this tutorial:

- [Install Git](#)
- [Install the Azure CLI](#)

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Create the web app

To create the web app that you'll work with, follow the [static HTML quickstart](#) through the **Browse to the app** step.

## Log in to the Azure portal

Open a browser and navigate to the [Azure portal](#).

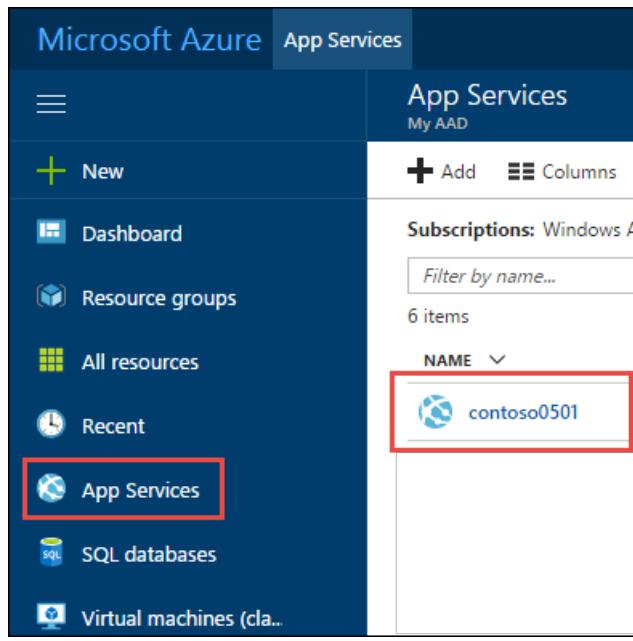
### Dynamic site acceleration optimization

If you want to optimize your CDN endpoint for dynamic site acceleration (DSA), you should use the [CDN portal](#) to

create your profile and endpoint. With [DSA optimization](#), the performance of web pages with dynamic content is measurably improved. For instructions about how to optimize a CDN endpoint for DSA from the CDN portal, see [CDN endpoint configuration to accelerate delivery of dynamic files](#). Otherwise, if you don't want to optimize your new endpoint, you can use the web app portal to create it by following the steps in the next section. Note that for [Azure CDN from Verizon](#) profiles, you cannot change the optimization of a CDN endpoint after it has been created.

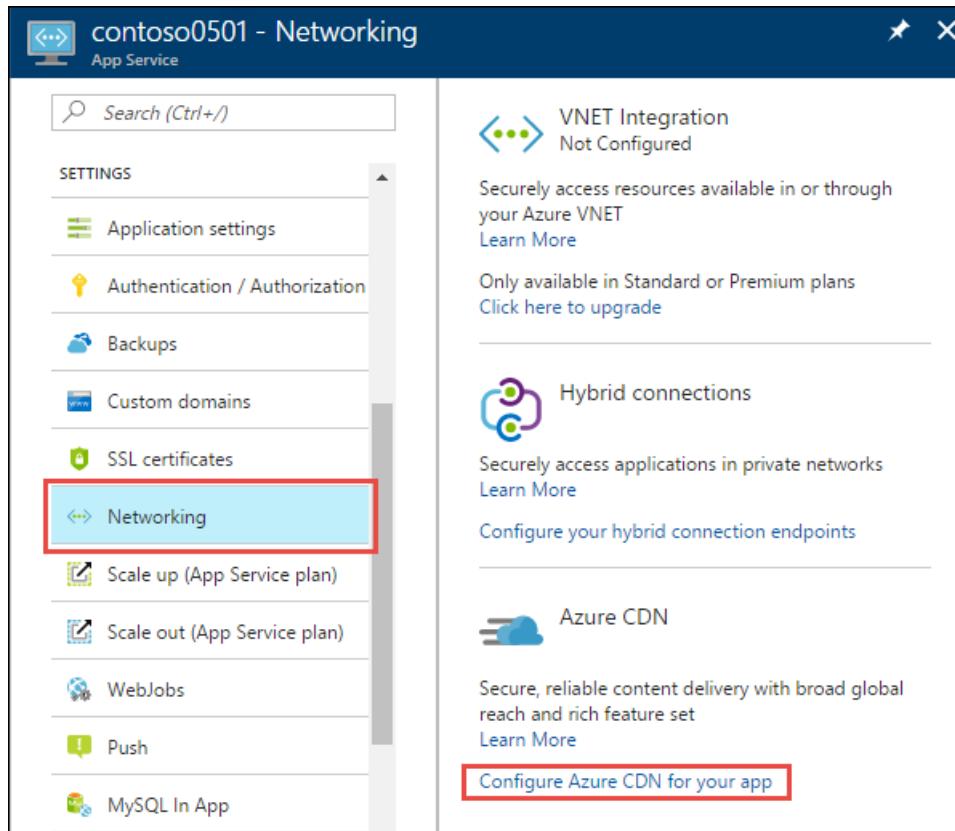
## Create a CDN profile and endpoint

In the left navigation, select **App Services**, and then select the app that you created in the [static HTML quickstart](#).



The screenshot shows the Microsoft Azure portal's App Services blade. On the left, there's a sidebar with various icons: New, Dashboard, Resource groups, All resources, Recent, App Services (which is selected and highlighted with a red box), SQL databases, and Virtual machines (cla...). The main pane displays a list of subscriptions under 'Subscriptions: Windows A...', with 'Filter by name...' and '6 items'. Below that is a table with a single row for 'contoso0501', which is also highlighted with a red box.

In the **App Service** page, in the **Settings** section, select **Networking > Configure Azure CDN for your app**.



The screenshot shows the 'Networking' page for the 'contoso0501' app service. The left sidebar lists settings like Application settings, Authentication / Authorization, Backups, Custom domains, SSL certificates, Networking (which is selected and highlighted with a red box), Scale up (App Service plan), Scale out (App Service plan), WebJobs, Push, and MySQL In App. The main pane contains sections for VNET Integration (Not Configured), Hybrid connections, and Azure CDN. The 'Configure Azure CDN for your app' button is highlighted with a red box.

In the **Azure Content Delivery Network** page, provide the **New endpoint** settings as specified in the table.

Azure CDN

The screenshot shows the Azure Content Delivery Network (CDN) interface. At the top, there's a header with the title "Azure Content Delivery Network". Below the header, a descriptive text explains that the Azure CDN is designed to send audio, video, images, and other files faster and more reliably to customers using servers that are closest to the users. It also links to "Learn more".

The main area is titled "Endpoints" and contains a table with columns: HOSTNAME, STATUS, and PROTOCOL. A message below the table states, "There are no CDN endpoints linked to your resource. You may create new endpoints below."

Below the table, there's a section titled "New endpoint" with fields for creating a new endpoint:

- CDN profile**: A radio button group with "Create new" selected (highlighted with a red box). The input field "myCDNProfile" contains a green checkmark.
- Pricing tier**: A dropdown menu showing "Standard Akamai" (highlighted with a red box).
- CDN endpoint name**: An input field containing "contoso0501" (highlighted with a red box), followed by ".azureedge.net".
- Origin hostname**: An input field containing "contoso0501.azurewebsites.net".

A large blue "Create" button is located at the bottom left of the form.

SETTING	SUGGESTED VALUE	DESCRIPTION
<b>CDN profile</b>	myCDNProfile	A CDN profile is a collection of CDN endpoints with the same pricing tier.
<b>Pricing tier</b>	Standard Akamai	The <a href="#">pricing tier</a> specifies the provider and available features. This tutorial uses <i>Standard Akamai</i> .
<b>CDN endpoint name</b>	Any name that is unique in the azureedge.net domain	You access your cached resources at the domain <endpointname>.azureedge.net.

Select **Create** to create a CDN profile.

Azure creates the profile and endpoint. The new endpoint appears in the **Endpoints** list, and when it's provisioned, the status is **Running**.

Azure CDN

Azure Content Delivery Network

The Azure Content Delivery Network (CDN) is designed to send audio, video, images, and other files faster and more reliably to customers using servers that are closest to the users. This dramatically increases speed and availability, resulting in significant user experience improvements. [Learn more](#)

**Endpoints**

HOSTNAME	STATUS	PROTOCOL	...
contoso0501.azureedge.net	Running	HTTP, HTTPS	...

## Test the CDN endpoint

Because it takes time for the registration to propagate, the endpoint isn't immediately available for use:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes within 90 minutes.

The sample app has an *index.html* file and *css*, *img*, and *js* folders that contain other static assets. The content paths for all of these files are the same at the CDN endpoint. For example, both of the following URLs access the *bootstrap.css* file in the *css* folder:

`http://<appname>.azurewebsites.net/css/bootstrap.css`

`http://<endpointname>.azureedge.net/css/bootstrap.css`

Navigate a browser to the following URL:

`http://<endpointname>.azureedge.net/index.html`

Azure App Service - Sample Static HTML Site

Azure App Service Web Apps

Azure Content Delivery Network (CDN)

You see the same page that you ran earlier in an Azure web app. Azure CDN has retrieved the origin web app's assets and is serving them from the CDN endpoint

To ensure that this page is cached in the CDN, refresh the page. Two requests for the same asset are sometimes required for the CDN to cache the requested content.

For more information about creating Azure CDN profiles and endpoints, see [Getting started with Azure CDN](#).

## Purge the CDN

The CDN periodically refreshes its resources from the origin web app based on the time-to-live (TTL) configuration. The default TTL is seven days.

At times you might need to refresh the CDN before the TTL expiration; for example, when you deploy updated content to the web app. To trigger a refresh, manually purge the CDN resources.

In this section of the tutorial, you deploy a change to the web app and purge the CDN to trigger the CDN to refresh its cache.

### Deploy a change to the web app

Open the *index.html* file and add - V2 to the H1 heading, as shown in the following example:

```
<h1>Azure App Service - Sample Static HTML Site - V2</h1>
```

Commit your change and deploy it to the web app.

```
git commit -am "version 2"  
git push azure master
```

Once deployment has completed, browse to the web app URL to see the change.

```
http://<appname>.azurewebsites.net/index.html
```



If you browse to the CDN endpoint URL for the home page, you won't see the change because the cached version in the CDN hasn't expired yet.

```
http://<endpointname>.azureedge.net/index.html
```



### Purge the CDN in the portal

To trigger the CDN to update its cached version, purge the CDN.

In the portal left navigation, select **Resource groups**, and then select the resource group that you created for your web app (*myResourceGroup*).

The screenshot shows the Microsoft Azure portal's 'Resource groups' page. On the left sidebar, 'Resource groups' is selected and highlighted with a red box. In the main content area, the title 'Resource groups' is displayed above a table header 'NAME'. A specific resource group, 'myResourceGroup', is highlighted with a red box.

In the list of resources, select your CDN endpoint.

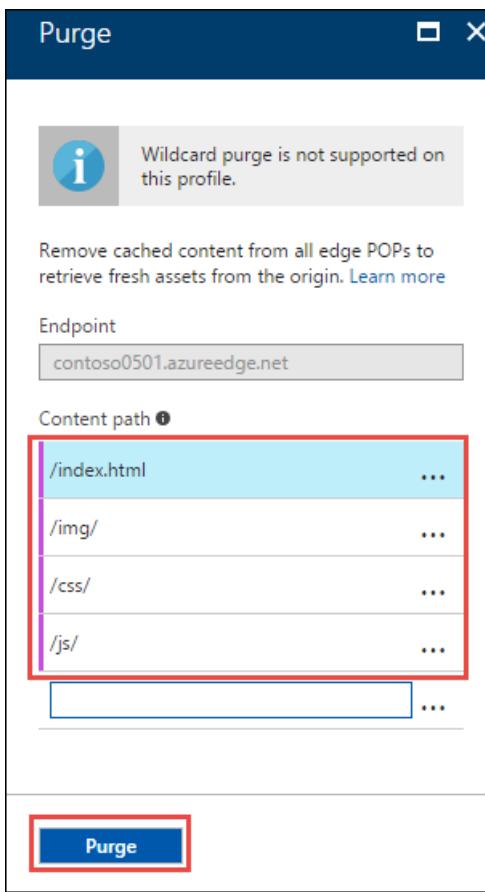
The screenshot shows the 'myResourceGroup' resource group details page. The left sidebar includes options like 'Overview' (which is selected and highlighted with a blue box), 'Activity log', 'Access control (IAM)', 'Tags', 'SETTINGS' (with 'Quickstart', 'Resource costs', 'Deployments', and 'Properties' listed), and a 'Search (Ctrl+/' input field). The main content area shows the 'Essentials' section with subscription information and deployment status. Below this is a table listing resources by name and type. One resource, 'contoso0501', is highlighted with a red box and identified as an 'Endpoint'.

At the top of the **Endpoint** page, select **Purge**.

The screenshot shows the 'contoso0501' endpoint details page. The left sidebar includes 'Overview' (selected and highlighted with a blue box), 'Activity log', and 'Access control (IAM)'. The main content area shows the 'Essentials' section with resource group information and a status of 'Running'. At the top right, there are buttons for 'Custom domain', 'Purge' (highlighted with a red box), 'Load', and 'Stop'.

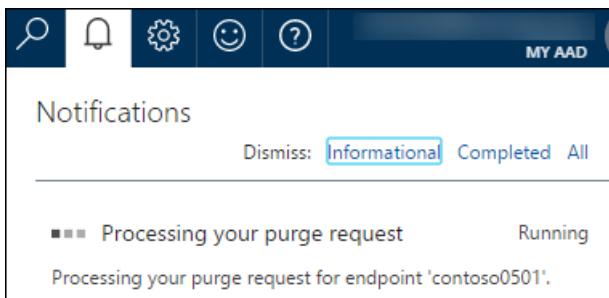
Enter the content paths you want to purge. You can pass a complete file path to purge an individual file, or a path segment to purge and refresh all content in a folder. Because you changed *index.html*, ensure that is in one of the paths.

At the bottom of the page, select **Purge**.



### Verify that the CDN is updated

Wait until the purge request finishes processing, which is typically a couple of minutes. To see the current status, select the bell icon at the top of the page.



When you browse to the CDN endpoint URL for *index.html*, you'll see the V2 that you added to the title on the home page, which indicates that the CDN cache has been refreshed.

A screenshot of a web browser window. The address bar shows "https://contoso0501.azureedge.net/index.html". The main content area displays the title "Azure App Service - Sample Static HTML Site - V2". The browser interface includes standard controls like back, forward, and refresh.

For more information, see [Purge an Azure CDN endpoint](#).

## Use query strings to version content

Azure CDN offers the following caching behavior options:

- Ignore query strings

- Bypass caching for query strings
- Cache every unique URL

The first option is the default, which means there is only one cached version of an asset regardless of the query string in the URL.

In this section of the tutorial, you change the caching behavior to cache every unique URL.

### Change the cache behavior

In the Azure portal **CDN Endpoint** page, select **Cache**.

Select **Cache every unique URL** from the **Query string caching behavior** drop-down list.

Select **Save**.

**contoso0501 - Cache**

**Overview** **Activity log** **Access control (IAM)** **Tags** **Diagnose and solve problems**

**SETTINGS**

**Origin** **Custom domains** **Compression** **Cache**

**About This Feature**  
Decide how CDN caches requests that include query strings. You can ignore any query contain query strings from being cached, or cache every request with a unique URL.  
[Learn more](#)

**Configure**

Query string caching behavior

- Ignore query strings
- Ignore query strings
- Bypass caching for query strings
- Cache every unique URL**

### Verify that unique URLs are cached separately

In a browser, navigate to the home page at the CDN endpoint, and include a query string:

```
http://<endpointname>.azureedge.net/index.html?q=1
```

Azure CDN returns the current web app content, which includes V2 in the heading.

To ensure that this page is cached in the CDN, refresh the page.

Open *index.html*, change V2 to V3, then deploy the change.

```
git commit -am "version 3"
git push azure master
```

In a browser, go to the CDN endpoint URL with a new query string, such as `q=2`. Azure CDN gets the current *index.html* file and displays V3. However, if you navigate to the CDN endpoint with the `q=1` query string, you see V2.

```
http://<endpointname>.azureedge.net/index.html?q=2
```



```
http://<endpointname>.azureedge.net/index.html?q=1
```



This output shows that each query string is treated differently:

- q=1 was used before, so cached contents are returned (V2).
- q=2 is new, so the latest web app contents are retrieved and returned (V3).

For more information, see [Control Azure CDN caching behavior with query strings](#).

## Clean up resources

In the preceding steps, you created Azure resources in a resource group. If you don't expect to need these resources in the future, delete the resource group by running the following command in the Cloud Shell:

```
az group delete --name myResourceGroup
```

This command may take a minute to run.

## Next steps

What you learned:

- Create a CDN endpoint.
- Refresh cached assets.
- Use query strings to control cached versions.

Learn how to optimize CDN performance in the following articles:

[Tutorial: Add a custom domain to your Azure CDN endpoint](#)

# Tutorial: Access storage blobs using an Azure CDN custom domain over HTTPS

11/20/2019 • 3 minutes to read • [Edit Online](#)

After you've integrated your Azure storage account with Azure Content Delivery Network (CDN), you can add a custom domain and enable HTTPS on that domain for your custom blob storage endpoint.

## Prerequisites

Before you can complete the steps in this tutorial, you must first integrate your Azure storage account with Azure CDN. For more information, see [Quickstart: Integrate an Azure storage account with Azure CDN](#).

## Add a custom domain

When you create a CDN endpoint in your profile, the endpoint name, which is a subdomain of `azureedge.net`, is included in the URL for delivering CDN content by default. You also have the option of associating a custom domain with a CDN endpoint. With this option, you deliver your content with a custom domain in your URL instead of an endpoint name. To add a custom domain to your endpoint, follow the instructions in this tutorial: [Add a custom domain to your Azure CDN endpoint](#).

## Configure HTTPS

By using the HTTPS protocol on your custom domain, you ensure that your data is delivered securely on the internet via TLS/SSL encryption. When your web browser is connected to a web site via HTTPS, it validates the web site's security certificate and verifies it's issued by a legitimate certificate authority. To configure HTTPS on your custom domain, follow the instructions in this tutorial: [Configure HTTPS on an Azure CDN custom domain](#).

## Shared Access Signatures

If your blob storage endpoint is configured to disallow anonymous read access, you should provide a [Shared Access Signature \(SAS\)](#) token in each request you make to your custom domain. By default, blob storage endpoints disallow anonymous read access. For more information about SAS, see [Managing anonymous read access to containers and blobs](#).

Azure CDN ignores any restrictions added to the SAS token. For example, all SAS tokens have an expiration time, which means that content can still be accessed with an expired SAS until that content is purged from the CDN point-of-presence (POP) servers. You can control how long data is cached on Azure CDN by setting the cache response header. For more information, see [Managing expiration of Azure Storage blobs in Azure CDN](#).

If you create multiple SAS URLs for the same blob endpoint, consider enabling query string caching. Doing so ensures that each URL is treated as a unique entity. For more information, see [Controlling Azure CDN caching behavior with query strings](#).

## HTTP-to-HTTPS redirection

You can elect to redirect HTTP traffic to HTTPS by creating a URL redirect rule with the [Standard rules engine](#) or the [Verizon Premium rules engine](#). Standard Rules engine is available only for Azure CDN from Microsoft profiles, while Verizon premium rules engine is available only from Azure CDN Premium from Verizon profiles.

The screenshot shows the Azure CDN rule configuration interface. At the top, there's a search bar with the placeholder 'Search' and a dropdown menu with options like 'All rules', 'My rules', and 'Import'. Below the search bar, the rule name 'EnforceHTTPS' is entered. To the right of the name are buttons for 'Add condition' and 'Add action', and icons for sorting and deleting. The rule structure is defined by 'If' and 'Then' sections. The 'If' section specifies 'Request protocol Equals HTTP~'. The 'Then' section is a 'URL redirect' with type 'Found (302)'. The 'Protocol' field is set to 'HTTPS~', while 'Hostname', 'Path', 'Query string', and 'Fragment' fields are left empty. There are also 'Matches' and 'Add' buttons.

In the above rule, leaving Hostname, Path, Query string, and Fragment will result in the incoming values being used in the redirect.

This screenshot shows a different view of the Azure CDN rule configuration, specifically the 'Edit rule' screen. It displays a single rule with the name 'Redirect HTTP to HTTPS'. The rule consists of an 'IF' condition (Request Scheme: Http) and an 'URL Redirect' action. The 'URL Redirect' settings include a code of 301, a source path of '/xxxxxxxx/cdn-endpoint-name/origin-path/(.\*)', and a destination path of 'https://\${host}/\$1'. There are also 'Matches' and 'Add' buttons.

In the above rule, *Cdn-endpoint-name* refers to the name that you configured for your CDN endpoint, which you can select from the drop-down list. The value for *origin-path* refers to the path within your origin storage account where your static content resides. If you're hosting all static content in a single container, replace *origin-path* with the name of that container.

## Pricing and billing

When you access blobs through Azure CDN, you pay [Blob storage prices](#) for traffic between the POP servers and the origin (Blob storage), and [Azure CDN pricing](#) for data accessed from the POP servers.

If, for example, you have a storage account in the United States that's being accessed using Azure CDN and someone in Europe attempts to access one of the blobs in that storage account via Azure CDN, Azure CDN first checks the POP closest to Europe for that blob. If found, Azure CDN accesses that copy of the blob and uses CDN pricing, because it's being accessed on Azure CDN. If it's not found, Azure CDN copies the blob to the POP server, which results in egress and transaction charges as specified in the Blob storage pricing, and then accesses the file on the POP server, which results in Azure CDN billing.

## Next steps

[Tutorial: Set Azure CDN caching rules](#)

# Tutorial: Add a custom domain to your Azure CDN endpoint

11/8/2019 • 8 minutes to read • [Edit Online](#)

This tutorial shows how to add a custom domain to an Azure Content Delivery Network (CDN) endpoint. When you use a CDN endpoint to deliver content, a custom domain is necessary if you would like your own domain name to be visible in your CDN URL. Having a visible domain name can be convenient for your customers and useful for branding purposes.

After you create a CDN endpoint in your profile, the endpoint name, which is a subdomain of azureedge.net, is included in the URL for delivering CDN content by default (for example, <https://contoso.azureedge.net/photo.png>). For your convenience, Azure CDN provides the option of associating a custom domain with a CDN endpoint. With this option, you deliver your content with a custom domain in your URL instead of an endpoint name (for example, <https://www.contoso.com/photo.png>).

In this tutorial, you learn how to:

- Create a CNAME DNS record.
- Associate the custom domain with your CDN endpoint.
- Verify the custom domain.

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Prerequisites

Before you can complete the steps in this tutorial, you must first create a CDN profile and at least one CDN endpoint. For more information, see [Quickstart: Create an Azure CDN profile and endpoint](#).

If you do not already have a custom domain, you must first purchase one with a domain provider. For example, see [Buy a custom domain name](#).

If you are using Azure to host your [DNS domains](#), you must delegate the domain provider's domain name system (DNS) to an Azure DNS. For more information, see [Delegate a domain to Azure DNS](#). Otherwise, if you are using a domain provider to handle your DNS domain, proceed to [Create a CNAME DNS record](#).

## Create a CNAME DNS record

Before you can use a custom domain with an Azure CDN endpoint, you must first create a canonical name (CNAME) record with your domain provider to point to your CDN endpoint. A CNAME record is a type of DNS record that maps a source domain name to a destination domain name. For Azure CDN, the source domain name is your custom domain name and the destination domain name is your CDN endpoint hostname. After Azure CDN verifies the CNAME record that you create, traffic addressed to the source custom domain (such as [www.contoso.com](http://www.contoso.com)) is routed to the specified destination CDN endpoint hostname (such as [contoso.azureedge.net](http://contoso.azureedge.net)).

A custom domain and its subdomain can be associated with only a single endpoint at a time. However, you can use different subdomains from the same custom domain for different Azure service endpoints by using multiple CNAME records. You can also map a custom domain with different subdomains to the same CDN endpoint.

#### NOTE

Any alias record type can be used for Custom domains if you're using Azure DNS as your domain provider. This walkthrough uses the CNAME record type. If you're using A or AAAA record types, follow the same steps below and replace CNAME with the record type of your choice. If you're using an alias record to add a root domain as a custom domain and you want to enable SSL, you must use manual validation as described in [this article](#). For more information, see [Point zone apex to Azure CDN endpoints](#).

## Map the temporary cdnverify subdomain

When you map an existing domain that is in production, there are special considerations. While you are registering your custom domain in the Azure portal, a brief period of downtime for the domain can occur. To avoid interruption of web traffic, first map your custom domain to your CDN endpoint hostname with the Azure cdnverify subdomain to create a temporary CNAME mapping. With this method, users can access your domain without interruption while the DNS mapping occurs.

Otherwise, if you are using your custom domain for the first time and no production traffic is running on it, you can directly map your custom domain to your CDN endpoint. Proceed to [Map the permanent custom domain](#).

To create a CNAME record with the cdnverify subdomain:

1. Sign in to the web site of the domain provider for your custom domain.
2. Find the page for managing DNS records by consulting the provider's documentation or searching for areas of the web site labeled **Domain Name**, **DNS**, or **Name server management**.
3. Create a CNAME record entry for your custom domain and complete the fields as shown in the following table (field names may vary):

SOURCE	TYPE	DESTINATION
cdnverify.www.contoso.com	CNAME	cdnverify.contoso.azureedge.net

- Source: Enter your custom domain name, including the cdnverify subdomain, in the following format: cdnverify.<custom domain name>. For example, cdnverify.www.contoso.com.
- Type: Enter **CNAME**.
- Destination: Enter your CDN endpoint hostname, including the cdnverify subdomain, in the following format: cdnverify.<*endpoint name*>.azureedge.net. For example, cdnverify.contoso.azureedge.net.

4. Save your changes.

For example, the procedure for the GoDaddy domain registrar is as follows:

1. Sign in and select the custom domain you want to use.
2. In the Domains section, select **Manage All**, then select **DNS | Manage Zones**.
3. For **Domain Name**, enter your custom domain, then select **Search**.
4. From the **DNS Management** page, select **Add**, then select **CNAME** in the **Type** list.
5. Complete the following fields of the CNAME entry:

The screenshot shows the 'DNS records' configuration page. At the top, there are three input fields: 'Type' set to 'CNAME', 'Host' set to 'cdnverify.www', and 'Points to' set to 'cdnverify.contoso.azureedge.net'. Below these, a 'TTL' field is set to '1 Hour'. In the bottom right corner, there are two buttons: a green 'Save' button and a white 'Cancel' button.

- Type: Leave *CNAME* selected.
- Host: Enter the subdomain of your custom domain to use, including the cdnverify subdomain name. For example, cdnverify.www.
- Points to: Enter the host name of your CDN endpoint, including the cdnverify subdomain name. For example, cdnverify.contoso.azureedge.net.
- TTL: Leave *1 Hour* selected.

6. Select **Save**.

The CNAME entry is added to the DNS records table.

Records			
Last updated 3/13/2018 11:07 AM			
Type	Name	Value	TTL
A	@	Parked	600 seconds
CNAME	cdnverify.www	cdnverify.contoso.azureedge.net	1 Hour

## Associate the custom domain with your CDN endpoint

After you've registered your custom domain, you can then add it to your CDN endpoint.

1. Sign in to the [Azure portal](#) and browse to the CDN profile containing the endpoint that you want to map to a custom domain.
2. On the **CDN profile** page, select the CDN endpoint to associate with the custom domain.

The **Endpoint** page opens.

3. Select **Custom domain**.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons like Dashboard, All resources, Resource groups, App Services, Function Apps, SQL databases, Azure Cosmos DB, and Virtual machines. The main content area is titled 'cdndocdemo1 Endpoint'. At the top right, there are several buttons: '+ Custom domain', 'Purge', 'Load', 'Stop', and 'Delete'. The '+ Custom domain' button is highlighted with a red box. Below these buttons, there's a section titled 'Essentials' with details such as Resource group (CdnDemoRG), Status (Running), Location (West US), and Protocols (HTTP, HTTPS). To the right of this, it shows Endpoint hostname (<https://cdndocdemo1.azureedge.net>), Origin hostname (<https://cdndocdemo.blob.core.windows.net>), and Optimization type (General web delivery). Under 'Custom domains', it says 'There are no custom domains to display'.

The **Add a custom domain** page opens.

4. For **Endpoint hostname**, the endpoint host name to use as the destination domain of your CNAME record is prefilled and is derived from your CDN endpoint URL: <endpoint hostname>.azureedge.net. It cannot be changed.
5. For **Custom hostname**, enter your custom domain, including the subdomain, to use as the source domain of your CNAME record. For example, www.contoso.com or cdn.contoso.com. Do not use the cdnverify subdomain name.

The screenshot shows the 'Add a custom domain' dialog box. It has a header 'Add a custom domain' with a close button. Below the header, there's a note: 'Create a DNS mapping from your custom hostname to the CDN endpoint hostname with your DNS provider. Then type the custom hostname here for verification.' A 'Learn more' link is provided. The next section contains two input fields: 'Endpoint hostname' with the value 'cdndocdemo1.azureedge.net' and 'Custom hostname' with an empty input field. Both fields have a red border around them, indicating they are required. At the bottom, there are two buttons: 'Add' (in a blue box) and 'Automation options'.

6. Select **Add**.

Azure verifies that the CNAME record exists for the custom domain name you entered. If the CNAME is correct, your custom domain will be validated.

It can take some time for the new custom domain settings to propagate to all CDN edge nodes:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes in 10 minutes.

## Verify the custom domain

After you have completed the registration of your custom domain, verify that the custom domain references your CDN endpoint.

1. Ensure that you have public content that is cached at the endpoint. For example, if your CDN endpoint is associated with a storage account, Azure CDN will cache the content in a public container. To test the custom domain, verify that your container is set to allow public access and contains at least one file.
2. In your browser, navigate to the address of the file by using the custom domain. For example, if your custom domain is `www.contoso.com`, the URL to the cached file should be similar to the following URL: `http://www.contoso.com/my-public-container/my-file.jpg`. Verify that the result is that same as when you access the CDN endpoint directly at `<endpoint hostname>.azureedge.net`.

## Map the permanent custom domain

If you have verified that the `cdnverify` subdomain has been successfully mapped to your endpoint (or if you are using a new custom domain that is not in production), you can then map the custom domain directly to your CDN endpoint hostname.

To create a CNAME record for your custom domain:

1. Sign in to the web site of the domain provider for your custom domain.
2. Find the page for managing DNS records by consulting the provider's documentation or searching for areas of the web site labeled **Domain Name**, **DNS**, or **Name Server Management**.
3. Create a CNAME record entry for your custom domain and complete the fields as shown in the following table (field names may vary):

SOURCE	TYPE	DESTINATION
<code>&lt;www.contoso.com&gt;</code>	CNAME	<code>contoso.azureedge.net</code>

- Source: Enter your custom domain name (for example, `www.contoso.com`).
  - Type: Enter **CNAME**.
  - Destination: Enter your CDN endpoint hostname. It must be in the following format: `<endpoint name>.azureedge.net`. For example, `contoso.azureedge.net`.
4. Save your changes.
  5. If you're previously created a temporary `cdnverify` subdomain CNAME record, delete it.
  6. If you are using this custom domain in production for the first time, follow the steps for [Associate the custom domain with your CDN endpoint](#) and [Verify the custom domain](#).

For example, the procedure for the GoDaddy domain registrar is as follows:

1. Sign in and select the custom domain you want to use.
2. In the Domains section, select **Manage All**, then select **DNS | Manage Zones**.
3. For **Domain Name**, enter your custom domain, then select **Search**.
4. From the **DNS Management** page, select **Add**, then select **CNAME** in the **Type** list.
5. Complete the fields of the CNAME entry:

Type: CNAME

Host: www

Points to: contoso.azureedge.net

TTL: 1 Hour

**Save** **Cancel**

- Type: Leave *CNAME* selected.
- Host: Enter the subdomain of your custom domain to use. For example, www or cdn.
- Points to: Enter the host name of your CDN endpoint. For example, contoso.azureedge.net.
- TTL: Leave *1 Hour* selected.

#### 6. Select **Save**.

The CNAME entry is added to the DNS records table.

Records			
Last updated 3/15/2018 11:13 AM			
Type	Name	Value	TTL
A	@	Parked	600 seconds
CNAME	www	contoso.azureedge.net	1 Hour

7. If you have a cdnverify CNAME record, select the pencil icon next to it, then select the trash can icon.

8. Select **Delete** to delete the CNAME record.

## Clean up resources

In the preceding steps, you added a custom domain to a CDN endpoint. If you no longer want to associate your endpoint with a custom domain, you can remove the custom domain by performing these steps:

1. In your CDN profile, select the endpoint with the custom domain that you want to remove.
2. From the **Endpoint** page, under Custom domains, right-click the custom domain that you want to remove, then select **Delete** from the context menu.

The custom domain is disassociated from your endpoint.

## Next steps

In this tutorial, you learned how to:

- Create a CNAME DNS record.
- Associate the custom domain with your CDN endpoint.
- Verify the custom domain.

Advance to the next tutorial to learn how to configure HTTPS on an Azure CDN custom domain.

[Tutorial: Configure HTTPS on an Azure CDN custom domain](#)



# Tutorial: Configure HTTPS on an Azure CDN custom domain

12/17/2019 • 13 minutes to read • [Edit Online](#)

This tutorial shows how to enable the HTTPS protocol for a custom domain that's associated with an Azure CDN endpoint. By using the HTTPS protocol on your custom domain (for example, <https://www.contoso.com>), you ensure that your sensitive data is delivered securely via TLS/SSL encryption when it is sent across the internet. When your web browser is connected to a web site via HTTPS, it validates the web site's security certificate and verifies it's issued by a legitimate certificate authority. This process provides security and protects your web applications from attacks.

Azure CDN supports HTTPS on a CDN endpoint hostname, by default. For example, if you create a CDN endpoint (such as <https://contoso.azureedge.net>), HTTPS is automatically enabled.

Some of the key attributes of the custom HTTPS feature are:

- No additional cost: There are no costs for certificate acquisition or renewal and no additional cost for HTTPS traffic. You pay only for GB egress from the CDN.
- Simple enablement: One-click provisioning is available from the [Azure portal](#). You can also use REST API or other developer tools to enable the feature.
- Complete certificate management is available: All certificate procurement and management is handled for you. Certificates are automatically provisioned and renewed prior to expiration, which removes the risks of service interruption due to a certificate expiring.

In this tutorial, you learn how to:

- Enable the HTTPS protocol on your custom domain.
- Use a CDN-managed certificate
- Use your own certificate
- Validate the domain
- Disable the HTTPS protocol on your custom domain.

## Prerequisites

### NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Before you can complete the steps in this tutorial, you must first create a CDN profile and at least one CDN endpoint. For more information, see [Quickstart: Create an Azure CDN profile and endpoint](#).

In addition, you must associate an Azure CDN custom domain on your CDN endpoint. For more information, see [Tutorial: Add a custom domain to your Azure CDN endpoint](#).

## IMPORTANT

CDN-managed certificates are not available for root or apex domains. If your Azure CDN custom domain is a root or apex domain, you must use the Bring your own certificate feature.

## SSL certificates

To enable the HTTPS protocol for securely delivering content on an Azure CDN custom domain, you must use an SSL certificate. You can choose to use a certificate that is managed by Azure CDN or use your own certificate.

- [Option 1 \(default\): Enable HTTPS with a CDN-managed certificate](#)
- [Option 2: Enable HTTPS with your own certificate](#)

When you use a CDN-managed certificate, the HTTPS feature can be turned on with just a few clicks. Azure CDN completely handles certificate management tasks such as procurement and renewal. After you enable the feature, the process starts immediately. If the custom domain is already mapped to the CDN endpoint, no further action is required. Azure CDN will process the steps and complete your request automatically. However, if your custom domain is mapped elsewhere, you must use email to validate your domain ownership.

To enable HTTPS on a custom domain, follow these steps:

1. Go to the [Azure portal](#) to find a certificate managed by your Azure CDN. Search for and select **CDN profiles**.
2. Choose your **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Akamai**, **Azure CDN Standard from Verizon**, or **Azure CDN Premium from Verizon** profile.
3. In the list of CDN endpoints, select the endpoint containing your custom domain.

The screenshot shows the Azure portal's Endpoint page for a CDN profile named 'CdnDemoRG'. The left sidebar lists navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Locks, Automation script, Quickstart, and Properties. The main content area has tabs for 'Essentials' and 'Endpoints'. Under 'Essentials', it shows the Resource group (CdnDemoRG), Status (Active), Location (West US), and Subscription information. Under 'Endpoints', there are three entries: 'cdndocdemo.azureedge.net' (Status: Running, Protocol: HTTP, HTTPS, Origin Type: Storage, Custom Domains: seattle.azurecdn.us), 'cdndocdemo2.azureedge.net' (Status: Running, Protocol: HTTP, HTTPS, Origin Type: Storage), and 'cdndocdemo3.azureedge.net' (Status: Running, Protocol: HTTP, HTTPS, Origin Type: Storage). The first endpoint is highlighted with a red border.

The **Endpoint** page appears.

4. In the list of custom domains, select the custom domain for which you want to enable HTTPS.

The **Custom domain** page appears.

5. Under Certificate management type, select **CDN managed**.
6. Select **On** to enable HTTPS.

Custom domain HTTPS	On	Off
* Certificate management type	<input checked="" type="radio"/> CDN managed	<input type="radio"/> Use my own certificate

7. Proceed to [Validate the domain](#).

## Validate the domain

If you already have a custom domain in use that is mapped to your custom endpoint with a CNAME record or you're using your own certificate, proceed to

[Custom domain is mapped to your CDN endpoint](#). Otherwise, if the CNAME record entry for your endpoint no longer exists or it contains the cdnverify subdomain, proceed to [Custom domain is not mapped to your CDN endpoint](#).

## Custom domain is mapped to your CDN endpoint by a CNAME record

When you added a custom domain to your endpoint, you created a CNAME record in the DNS table of your domain registrar to map it to your CDN endpoint hostname. If that CNAME record still exists and does not contain the cdnverify subdomain, the DigiCert CA uses it to automatically validate ownership of your custom domain.

If you're using your own certificate, domain validation is not required.

Your CNAME record should be in the following format, where *Name* is your custom domain name and *Value* is your CDN endpoint hostname:

NAME	TYPE	VALUE
<www.contoso.com>	CNAME	contoso.azureedge.net

For more information about CNAME records, see [Create the CNAME DNS record](#).

If your CNAME record is in the correct format, DigiCert automatically verifies your custom domain name and creates a dedicated certificate for your domain name. DigiCert won't send you a verification email and you won't need to approve your request. The certificate is valid for one year and will be auto-renewed before it expires.

Proceed to [Wait for propagation](#).

Automatic validation typically takes a few hours. If you don't see your domain validated in 24 hours, open a support ticket.

### NOTE

If you have a Certificate Authority Authorization (CAA) record with your DNS provider, it must include DigiCert as a valid CA. A CAA record allows domain owners to specify with their DNS providers which CAs are authorized to issue certificates for their domain. If a CA receives an order for a certificate for a domain that has a CAA record and that CA is not listed as an authorized issuer, it is prohibited from issuing the certificate to that domain or subdomain. For information about managing CAA records, see [Manage CAA records](#). For a CAA record tool, see [CAA Record Helper](#).

## Custom domain is not mapped to your CDN endpoint

### NOTE

If you are using **Azure CDN from Akamai**, the following CNAME should be set up to enable automated domain validation.  
"\_acme-challenge.<custom domain hostname> -> CNAME -> <custom domain hostname>.ak-acme-challenge.azureedge.net"

If the CNAME record entry contains the cdnverify subdomain, follow the rest of the instructions in this step.

DigiCert sends a verification email to the following email addresses. Verify that you can approve directly from one of the following addresses:

admin@<your-domain-name.com>  
administrator@<your-domain-name.com>  
webmaster@<your-domain-name.com>  
hostmaster@<your-domain-name.com>  
postmaster@<your-domain-name.com>

You should receive an email in a few minutes, similar to the following example, asking you to approve the request. If you are using a spam filter, add verification@digicert.com to its allow list. If you don't receive an email within 24 hours, contact Microsoft support.

**From:** DigiCert [mailto:[admin@digicert.com](mailto:admin@digicert.com)]  
**Sent:** Wednesday, September 6, 2017 5:49 PM  
**To:** [REDACTED]  
**Subject:** Please validate ownership of your domain seattle.azurecdn.us -- DigiCert order [REDACTED]



**Verizon Digital Media Services, Inc.**

DigiCert recently received a certificate request that requires approval prior to the certificate's issuance.

To review and approve this request, visit the following link:

<https://www.digicert.com/link/approve.php?t=vrg6wz2gvg97vx95hg>

Domain names requiring your approval: seattle.azurecdn.us

The certificate was requested by:

Verizon Digital Media Services  
VDMS  
[ssl-orders@verizondigitalmedia.com](mailto:ssl-orders@verizondigitalmedia.com)  
+1 (877) 334-3236

If you do not approve of this request, or if you have any questions or concerns please contact us directly.

Thanks!

The DigiCert Team

Phone: 1-801-701-9600  
Email: [support@digicert.com](mailto:support@digicert.com)  
Live Chat: [www.digicert.com](http://www.digicert.com)

When you click on the approval link, you are directed to the following online approval form:

**DigiCert Approval Order # [REDACTED]**

Please review and approve the request below. Approval is required prior to the certificate's issuance. If this approval request contains any errors or if you wish to cancel this request, please contact DigiCert by phone or email.

Domain **seattle.azurecdn.us**  
Organization **Verizon Digital Media Services, Inc.**  
Los Angeles, California, USA  
Product **1 Year Standard SSL**

**Authorization:**

I am the registrant or am authorized by the registrant of the domain name(s) referenced above and have the authority to approve certificate requests for the domain. I confirm and agree to the following:

1. **Verizon Digital Media Services** (account # [REDACTED]) has the authority to apply for SSL Certificates for this domain on behalf of **Verizon Digital Media Services, Inc.** (order # [REDACTED]),
2. **Verizon Digital Media Services, Inc.** has the right to use and obtain digital certificates for both the domain(s) referenced above and any sub domain(s) of the referenced domains,
3. DigiCert may rely on this authorization for any subsequent digital certificate renewals and/or orders by **Verizon Digital Media Services, Inc.** until this authorization is revoked by written notice to DigiCert using one of the following methods: 1) Email to [admin@digicert.com](mailto:admin@digicert.com) or 2) Mail to DigiCert, Inc. (Attention: Legal) - 2801 N. Thanksgiving Way - Suite 500 - Lehi, UT 84043
4. I will promptly notify DigiCert if this authorization is revoked or if a domain name listed above is transferred to a third party, and
5. DigiCert may periodically (typically once every three years) reconfirm the Applicant's control over the domain name(s) and approval of the corresponding certificate(s) using a reconfirmation email sent to this email address. I acknowledge that I may not opt out of receiving reconfirmation emails.

I approve all future orders placed through account # [REDACTED] for orders under the azurecdn.us domain name(s). (Recommended)

I approve the specific host names used in this request.

Enter your first and last name (e.g. John Smith, not company name) :

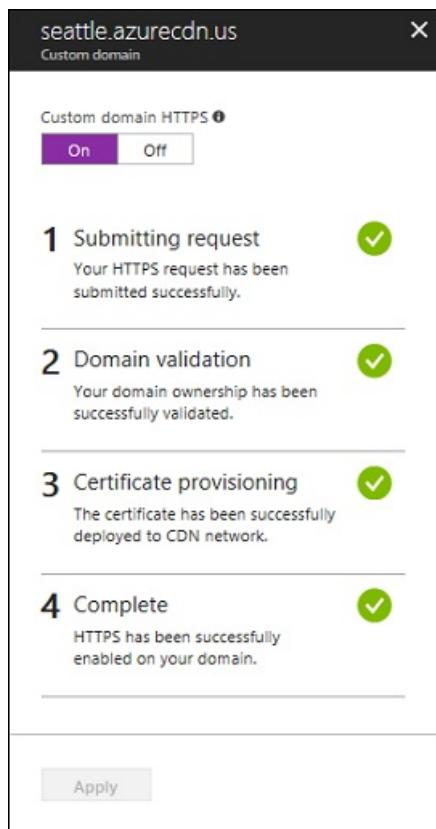
Follow the instructions on the form; you have two verification options:

- You can approve all future orders placed through the same account for the same root domain; for example, contoso.com. This approach is recommended if you plan to add additional custom domains for the same root domain.
- You can approve just the specific host name used in this request. Additional approval is required for subsequent requests.

After approval, DigiCert completes the certificate creation for your custom domain name. The certificate is valid for one year and will be auto-renewed before it's expired.

## Wait for propagation

After the domain name is validated, it can take up to 6-8 hours for the custom domain HTTPS feature to be activated. When the process is complete, the custom HTTPS status in the Azure portal is set to **Enabled** and the four operation steps in the custom domain dialog are marked as complete. Your custom domain is now ready to use HTTPS.



### Operation progress

The following table shows the operation progress that occurs when you enable HTTPS. After you enable HTTPS, four operation steps appear in the custom domain dialog. As each step becomes active, additional substep details appear under the step as it progresses. Not all of these substeps will occur. After a step successfully completes, a green check mark appears next to it.

OPERATION STEP	OPERATION SUBSTEP DETAILS
1 Submitting request	Submitting request
	Your HTTPS request is being submitted.
	Your HTTPS request has been submitted successfully.

OPERATION STEP	OPERATION SUBSTEP DETAILS
2 Domain validation	Domain is automatically validated if it is CNAME mapped to the CDN Endpoint. Otherwise, a verification request will be sent to the email listed in your domain's registration record (WHOIS registrant). Please verify the domain as soon as possible.
	Your domain ownership has been successfully validated.
	Domain ownership validation request expired (customer likely didn't respond within 6 days). HTTPS will not be enabled on your domain. *
	Domain ownership validation request was rejected by the customer. HTTPS will not be enabled on your domain. *
3 Certificate provisioning	The certificate authority is currently issuing the certificate needed to enable HTTPS on your domain.
	The certificate has been issued and is currently being deployed to CDN network. This could take up to 6 hours.
	The certificate has been successfully deployed to CDN network.
4 Complete	HTTPS has been successfully enabled on your domain.

\* This message doesn't appear unless an error has occurred.

If an error occurs before the request is submitted, the following error message is displayed:

We encountered an unexpected error while processing your HTTPS request. Please try again and contact support if the issue persists.

## Clean up resources - disable HTTPS

In the preceding steps, you enabled the HTTPS protocol on your custom domain. If you no longer want to use your custom domain with HTTPS, you can disable HTTPS by performing these steps:

### Disable the HTTPS feature

1. In the [Azure portal](#), search for and select **CDN profiles**.
2. Choose your **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, or **Azure CDN Premium from Verizon** profile.
3. In the list of endpoints, pick the endpoint containing your custom domain.
4. Choose the custom domain for which you want to disable HTTPS.

The screenshot shows the Azure portal interface for managing custom domains. On the left, there's a sidebar with various settings like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. Under SETTINGS, 'Custom domains' is selected and highlighted in blue. The main pane displays a table titled 'Custom domains' with columns: HOSTNAME, CUSTOM HTTPS, and DETAILS. A single row is shown for 'seattle.azurecdn.us', which has 'Enabled' under CUSTOM HTTPS and 'Certificate successfully deployed' under DETAILS. A red arrow points to the 'seattle.azurecdn.us' row.

5. Choose **Off** to disable HTTPS, then select **Apply**.

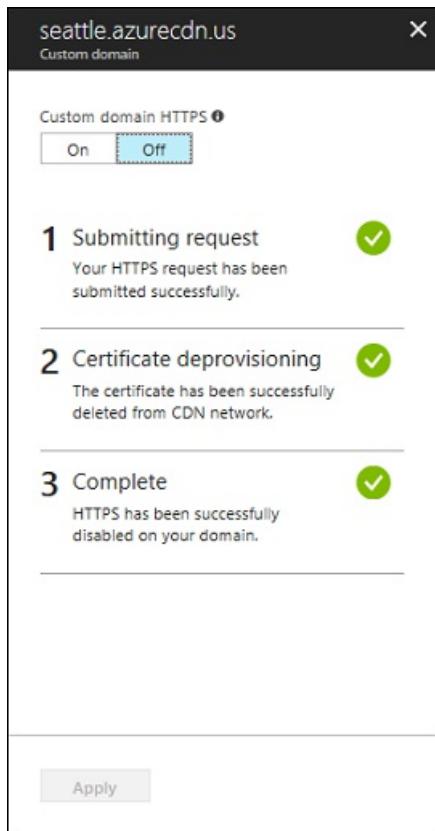
The screenshot shows a modal dialog for managing the custom domain 'seattle.azurecdn.us'. The title bar says 'Custom domain'. Inside, it says 'Custom domain HTTPS' with a help icon. There are two buttons: 'On' (white) and 'Off' (purple). Below this, there are four numbered steps with green checkmarks:

- 1 Submitting request: Your HTTPS request has been submitted successfully.
- 2 Domain validation: Your domain ownership has been successfully validated.
- 3 Certificate provisioning: The certificate has been successfully deployed to CDN network.
- 4 Complete: HTTPS has been successfully enabled on your domain.

At the bottom of the dialog is a blue 'Apply' button.

#### Wait for propagation

After the custom domain HTTPS feature is disabled, it can take up to 6-8 hours for it to take effect. When the process is complete, the custom HTTPS status in the Azure portal is set to **Disabled** and the three operation steps in the custom domain dialog are marked as complete. Your custom domain can no longer use HTTPS.



#### Operation progress

The following table shows the operation progress that occurs when you disable HTTPS. After you disable HTTPS, three operation steps appear in the Custom domain dialog. As each step becomes active, additional details appear under the step. After a step successfully completes, a green check mark appears next to it.

OPERATION PROGRESS	OPERATION DETAILS
1 Submitting request	Submitting your request
2 Certificate deprovisioning	Deleting certificate
3 Complete	Certificate deleted

## Frequently asked questions

### 1. Who is the certificate provider and what type of certificate is used?

For both **Azure CDN from Verizon** and **Azure CDN from Microsoft**, a dedicated/single certificate provided by DigiCert is used for your custom domain.

### 2. Do you use IP-based or SNI TLS/SSL?

Both **Azure CDN from Verizon** and **Azure CDN Standard from Microsoft** use SNI TLS/SSL.

### 3. What if I don't receive the domain verification email from DigiCert?

If you have a CNAME entry for your custom domain that points directly to your endpoint hostname (and you are not using the cdnverify subdomain name), you won't receive a domain verification email.

Validation occurs automatically. Otherwise, if you don't have a CNAME entry and you haven't received an email within 24 hours, contact Microsoft support.

### 4. Is using a SAN certificate less secure than a dedicated certificate?

A SAN certificate follows the same encryption and security standards as a dedicated certificate. All issued

SSL certificates use SHA-256 for enhanced server security.

5. *Do I need a Certificate Authority Authorization record with my DNS provider?*

No, a Certificate Authority Authorization record is not currently required. However, if you do have one, it must include DigiCert as a valid CA.

6. *On June 20, 2018, Azure CDN from Verizon started using a dedicated certificate with SNI TLS/SSL by default. What happens to my existing custom domains using Subject Alternative Names (SAN) certificate and IP-based TLS/SSL?*

Your existing domains will be gradually migrated to single certificate in the upcoming months if Microsoft analyzes that only SNI client requests are made to your application. If Microsoft detects there some non-SNI client requests made to your application, your domains will stay in the SAN certificate with IP-based TLS/SSL. In any case, there will be no interruption to your service or support to your client requests regardless of whether those requests are SNI or non-SNI.

7. *How do cert renewals work with Bring Your Own Certificate?*

To ensure a newer certificate is deployed to PoP infrastructure, simply upload your new certificate to Azure KeyVault, and then in your SSL settings on Azure CDN, choose the newest certificate version and hit save. Azure CDN will then propagate your new updated cert.

## Next steps

In this tutorial, you learned how to:

- Enable the HTTPS protocol on your custom domain.
- Use a CDN-managed certificate
- Use your own certificate
- Validate the domain.
- Disable the HTTPS protocol on your custom domain.

Advance to the next tutorial to learn how to configure caching on your CDN endpoint.

[Tutorial: Set Azure CDN caching rules](#)

# Tutorial: Set Azure CDN caching rules

11/20/2019 • 2 minutes to read • [Edit Online](#)

## NOTE

Caching rules are available only for **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles. For **Azure CDN from Microsoft** profiles, you must use the [Standard rules engine](#). For **Azure CDN Premium from Verizon** profiles, you must use the [Verizon Premium rules engine](#) in the **Manage** portal for similar functionality.

This tutorial describes how you can use Azure Content Delivery Network (CDN) caching rules to set or modify default cache expiration behavior both globally and with custom conditions, such as a URL path and file extension. Azure CDN provides two types of caching rules:

- Global caching rules: You can set one global caching rule for each endpoint in your profile, which affects all requests to the endpoint. The global caching rule overrides any HTTP cache-directive headers, if set.
- Custom caching rules: You can set one or more custom caching rules for each endpoint in your profile. Custom caching rules match specific paths and file extensions, are processed in order, and override the global caching rule, if set.

In this tutorial, you learn how to:

- Open the caching rules page.
- Create a global caching rule.
- Create a custom caching rule.

If you don't have an [Azure subscription](#), create a [free account](#) before you begin.

## Prerequisites

Before you can complete the steps in this tutorial, you must first create a CDN profile and at least one CDN endpoint. For more information, see [Quickstart: Create an Azure CDN profile and endpoint](#).

## Open the Azure CDN caching rules page

1. In the [Azure portal](#), select a CDN profile, then select an endpoint.
2. In the left pane under Settings, select **Caching rules**.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons like Dashboard, App Services, and Storage accounts. The main area shows the 'cdndocdemo1' endpoint details. In the center-left, there's a navigation pane with 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Origin', 'Custom domains', 'Compression', and 'Caching rules'. The 'Caching rules' item is highlighted with a red box. On the right, under the 'Essentials' section, it shows the endpoint hostname as <https://cdndocdemo1.azureedge.net>, the origin hostname as <https://cdndocdemo.blob.core.windows.net>, and other configuration details like location (West US), protocols (HTTP, HTTPS), and optimization type (General web delivery). Below this, there's a section for 'Custom domains' which is currently empty.

The **Caching rules** page appears.

The screenshot shows the 'cdndocdemo1 - Caching rules' page. At the top, there are save and discard buttons. Below that is an 'About This Feature' section with a link to 'Learn more'. It shows 'Optimized for' as 'General web delivery', 'Default caching behavior' as 'Set if missing', and 'Default cache expiration duration' as '7 days'. Under 'Global caching rules', it says these rules affect all requests and can be overridden by custom rules. It includes fields for 'Caching behavior' (set to 'Not set'), 'Cache expiration duration' (set to 0 days, 0 hours, 0 minutes, 0 seconds), and 'Query string caching behavior' (set to 'Ignore query strings'). At the bottom, there's a table for 'Custom caching rules' with columns for 'MATCH CONDITION', 'MATCH VALUE(S)', 'CACHING BEHAVIOR', 'DAYS', 'HOURS', 'MINUTES', and 'SECONDS'. There are buttons for 'Move up', 'Move down', 'Move to top', 'Move to bottom', 'Insert', and 'Delete'.

## Set global caching rules

Create a global caching rule as follows:

1. Under **Global caching rules**, set **Query string caching behavior** to **Ignore query strings**.
2. Set **Caching behavior** to **Set if missing**.
3. For **Cache expiration duration**, enter 10 in the **Days** field.

The global caching rule affects all requests to the endpoint. This rule honors the origin cache-directive headers, if they exist (`Cache-Control` or `Expires`); otherwise, if they are not specified, it sets the cache to 10 days.

**Global caching rules**

These rules affect the CDN caching behavior for all requests, and can be overridden using Custom Cache Rules below for certain scenarios. Note that the Query string caching behavior setting does not affect files that are not cached by the CDN.

Caching behavior <small>i</small>	Set if missing			
Cache expiration duration <small>i</small>	Days 10	Hours 0	Minutes 0	Seconds 0
Query string caching behavior <small>i</small>	Ignore query strings			

## Set custom caching rules

Create a custom caching rule as follows:

- Under **Custom caching rules**, set **Match condition** to **Path** and **Match value** to `/images/*.jpg`.
- Set **Caching behavior** to **Override** and enter 30 in the **Days** field.

This custom caching rule sets a cache duration of 30 days on any `.jpg` image files in the `/images` folder of your endpoint. It overrides any `Cache-Control` or `Expires` HTTP headers that are sent by the origin server.

**Custom caching rules**

Create caching rules based on specific match conditions. These rules override the default settings above, and are evaluated from top to down. This means that rules lower on the list can override rules above it in the list, as well as the global caching rules and default behavior. Therefore it makes more sense to have more specific rules towards the bottom of the list so they are not overwritten by a general rule under them. For example a rule for path `'/folder/images/*'` should be below a rule for path `'/folder/*'`.

		Move up	Move down	Move to top	Move to bottom	Insert	Delete	
<input type="checkbox"/>	MATCH CONDITION	MATCH VALUE(S)		CACHING BEHAVIOR	DAYS	HOURS	MINUTES	SECONDS
<input checked="" type="checkbox"/>	Path	<code>/images/*.jpg</code>		Override	30	0	0	0
					0	0	0	0

## Clean up resources

In the preceding steps, you created caching rules. If you no longer want to use these caching rules, you can remove them by performing these steps:

- Select a CDN profile, then select the endpoint with the caching rules you want to remove.
- In the left pane under Settings, select **Caching rules**.
- Under **Global caching rules**, set **Caching behavior** to **Not set**.
- Under **Custom caching rules**, select the check box next to the rule you want to delete.
- Select **Delete**.
- From the top of the page, select **Save**.

## Next steps

In this tutorial, you learned how to:

- Open the caching rules page.
- Create a global caching rule.
- Create a custom caching rule.

Advance to the next article to learn how to configure additional caching rule settings.

[Control Azure CDN caching behavior with caching rules](#)

# Set up the Standard rules engine for Azure CDN

11/18/2019 • 2 minutes to read • [Edit Online](#)

This article describes how to set up and use the Standard rules engine for Azure Content Delivery Network (Azure CDN).

## Standard rules engine

You can use the Standard rules engine for Azure CDN to customize how HTTP requests are handled. For example, you can use the rules engine to enforce content delivery to use specific protocols, to define a caching policy, or to modify an HTTP header. This article demonstrates how to create a rule that automatically redirects users to HTTPS.

### NOTE

The rules engine that's described in this article is available only for Standard Azure CDN from Microsoft.

## Redirect users to HTTPS

1. In your Microsoft profiles, go to Azure Content Delivery Network.
2. On the **CDN profile** page, select the endpoint you want to create rules for.
3. Select the **Rules Engine** tab.

The **Rules Engine** pane opens and displays the list of available global rules.

The screenshot shows the 'Rules Engine' pane with a single global rule named 'Global'. The pane includes buttons for Save, Add rule, and Export. A note at the top says: 'Use the global rule to configure actions that will always trigger. For example, cache TTL settings. Later rules with scoped match conditions will take priority.' Below the note, it says 'There are no actions.'

### IMPORTANT

The order in which multiple rules are listed affects how rules are handled. The actions that are specified in a rule might be overwritten by a subsequent rule.

4. Select **Add rule** and enter a rule name. Rule names must begin with a letter and can contain only numbers and letters.
5. To identify the type of requests the rule applies to, create a match condition:
  - a. Select **Add condition**, and then select the **Request protocol** match condition.
  - b. For **Operator**, select **Equals**.
  - c. For **Value**, select **HTTP**.

Name \* EnforceHTTPS ✓

+ Add condition ✓ + Add action ✓ | ⌂ ⌃ ⌄ ⌅ | ⌂

If Request protocol Operator \* Value \*

Operator Equals Value HTTP✓

There are no actions.

The action cannot be empty.

#### NOTE

You can select from multiple match conditions in the **Add condition** drop-down list. For a detailed list of match conditions, see [Match conditions in the Standard rules engine](#).

6. Select the action to apply to the requests that satisfy the match condition:

- Select **Add action**, and then select **URL redirect**.
- For **Type**, select **Found (302)**.
- For **Protocol**, select **HTTPS**.
- Leave all other fields blank to use incoming values.

Name \* EnforceHTTPS ✓

+ Add condition ✓ + Add action ✓ | ⌂ ⌃ ⌄ ⌅ | ⌂

If Request protocol Operator \* Value \*

Operator Equals Value HTTP✓

Then URL redirect

Type \* Found (302)✓ Protocol HTTPS✓ Hostname ⓘ Path ⓘ Query string ⓘ Fragment ⓘ

#### NOTE

You can select from multiple actions in the **Add action** drop-down list. For a detailed list of actions, see [Actions in the Standard rules engine](#).

7. Select **Save** to save the new rule. The rule is now available to use.

#### IMPORTANT

Rule changes might take up to 15 minutes to propagate through Azure CDN.

## Next steps

- [Azure CDN overview](#)
- [Standard rules engine reference](#)
- [Match conditions in the Standard rules engine](#)
- [Actions in the Standard rules engine](#)

# How caching works

7/5/2019 • 9 minutes to read • [Edit Online](#)

This article provides an overview of general caching concepts and how [Azure Content Delivery Network \(CDN\)](#) uses caching to improve performance. If you'd like to learn about how to customize caching behavior on your CDN endpoint, see [Control Azure CDN caching behavior with caching rules](#) and [Control Azure CDN caching behavior with query strings](#).

## Introduction to caching

Caching is the process of storing data locally so that future requests for that data can be accessed more quickly. In the most common type of caching, web browser caching, a web browser stores copies of static data locally on a local hard drive. By using caching, the web browser can avoid making multiple round-trips to the server and instead access the same data locally, thus saving time and resources. Caching is well-suited for locally managing small, static data such as static images, CSS files, and JavaScript files.

Similarly, caching is used by a content delivery network on edge servers close to the user to avoid requests traveling back to the origin and reducing end-user latency. Unlike a web browser cache, which is used only for a single user, the CDN has a shared cache. In a CDN shared cache, a file that is requested by one user can be accessed later by other users, which greatly decreases the number of requests to the origin server.

Dynamic resources that change frequently or are unique to an individual user cannot be cached. Those types of resources, however, can take advantage of dynamic site acceleration (DSA) optimization on the Azure Content Delivery Network for performance improvements.

Caching can occur at multiple levels between the origin server and the end user:

- Web server: Uses a shared cache (for multiple users).
- Content delivery network: Uses a shared cache (for multiple users).
- Internet service provider (ISP): Uses a shared cache (for multiple users).
- Web browser: Uses a private cache (for one user).

Each cache typically manages its own resource freshness and performs validation when a file is stale. This behavior is defined in the HTTP caching specification, [RFC 7234](#).

### Resource freshness

Because a cached resource can potentially be out-of-date, or stale (as compared to the corresponding resource on the origin server), it is important for any caching mechanism to control when content is refreshed. To save time and bandwidth consumption, a cached resource is not compared to the version on the origin server every time it is accessed. Instead, as long as a cached resource is considered to be fresh, it is assumed to be the most current version and is sent directly to the client. A cached resource is considered to be fresh when its age is less than the age or period defined by a cache setting. For example, when a browser reloads a web page, it verifies that each cached resource on your hard drive is fresh and loads it. If the resource is not fresh (stale), an up-to-date copy is loaded from the server.

### Validation

If a resource is considered to be stale, the origin server is asked to validate it, that is, determine whether the data in the cache still matches what's on the origin server. If the file has been modified on the origin server, the cache updates its version of the resource. Otherwise, if the resource is fresh, the data is delivered directly from the cache without validating it first.

## CDN caching

Caching is integral to the way a CDN operates to speed up delivery and reduce origin load for static assets such as images, fonts, and videos. In CDN caching, static resources are selectively stored on strategically placed servers that are more local to a user and offers the following advantages:

- Because most web traffic is static (for example, images, fonts, and videos), CDN caching reduces network latency by moving content closer to the user, thus reducing the distance that data travels.
- By offloading work to a CDN, caching can reduce network traffic and the load on the origin server. Doing so reduces cost and resource requirements for the application, even when there are large numbers of users.

Similar to how caching is implemented in a web browser, you can control how caching is performed in a CDN by sending cache-directive headers. Cache-directive headers are HTTP headers, which are typically added by the origin server. Although most of these headers were originally designed to address caching in client browsers, they are now also used by all intermediate caches, such as CDNs.

Two headers can be used to define cache freshness: `Cache-Control` and `Expires`. `Cache-Control` is more current and takes precedence over `Expires`, if both exist. There are also two types of headers used for validation (called validators): `ETag` and `Last-Modified`. `ETag` is more current and takes precedence over `Last-Modified`, if both are defined.

## Cache-directive headers

### IMPORTANT

By default, an Azure CDN endpoint that is optimized for DSA ignores cache-directive headers and bypasses caching. For **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles, you can adjust how an Azure CDN endpoint treats these headers by using [CDN caching rules](#) to enable caching. For **Azure CDN Premium from Verizon** profiles only, you use the [rules engine](#) to enable caching.

Azure CDN supports the following HTTP cache-directive headers, which define cache duration and cache sharing.

### Cache-Control:

- Introduced in HTTP 1.1 to give web publishers more control over their content and to address the limitations of the `Expires` header.
- Overrides the `Expires` header, if both it and `Cache-Control` are defined.
- When used in an HTTP request from the client to the CDN POP, `Cache-Control` is ignored by all Azure CDN profiles, by default.
- When used in an HTTP response from the client to the CDN POP:
  - **Azure CDN Standard/Premium from Verizon** and **Azure CDN Standard from Microsoft** support all `Cache-Control` directives.
  - **Azure CDN Standard from Akamai** supports only the following `Cache-Control` directives; all others are ignored:
    - `max-age` : A cache can store the content for the number of seconds specified. For example, `Cache-Control: max-age=5`. This directive specifies the maximum amount of time the content is considered to be fresh.
    - `no-cache` : Cache the content, but validate the content every time before delivering it from the cache. Equivalent to `Cache-Control: max-age=0`.
    - `no-store` : Never cache the content. Remove content if it has been previously stored.

### Expires:

- Legacy header introduced in HTTP 1.0; supported for backwards compatibility.

- Uses a date-based expiration time with second precision.
- Similar to `Cache-Control: max-age`.
- Used when `Cache-Control` doesn't exist.

#### **Pragma:**

- Not honored by Azure CDN, by default.
- Legacy header introduced in HTTP 1.0; supported for backwards compatibility.
- Used as a client request header with the following directive: `no-cache`. This directive instructs the server to deliver a fresh version of the resource.
- `Pragma: no-cache` is equivalent to `Cache-Control: no-cache`.

## Validators

When the cache is stale, HTTP cache validators are used to compare the cached version of a file with the version on the origin server. **Azure CDN Standard/Premium from Verizon** supports both `ETag` and `Last-Modified` validators by default, while **Azure CDN Standard from Microsoft** and **Azure CDN Standard from Akamai** supports only `Last-Modified` by default.

#### **ETag:**

- **Azure CDN Standard/Premium from Verizon** supports `ETag` by default, while **Azure CDN Standard from Microsoft** and **Azure CDN Standard from Akamai** do not.
- `ETag` defines a string that is unique for every file and version of a file. For example, `ETag: "17f0ddd99ed5bbe4edffdd6496d7131f"`.
- Introduced in HTTP 1.1 and is more current than `Last-Modified`. Useful when the last modified date is difficult to determine.
- Supports both strong validation and weak validation; however, Azure CDN supports only strong validation. For strong validation, the two resource representations must be byte-for-byte identical.
- A cache validates a file that uses `ETag` by sending an `If-None-Match` header with one or more `ETag` validators in the request. For example, `If-None-Match: "17f0ddd99ed5bbe4edffdd6496d7131f"`. If the server's version matches an `ETag` validator on the list, it sends status code 304 (Not Modified) in its response. If the version is different, the server responds with status code 200 (OK) and the updated resource.

#### **Last-Modified:**

- For **Azure CDN Standard/Premium from Verizon** only, `Last-Modified` is used if `ETag` is not part of the HTTP response.
- Specifies the date and time that the origin server has determined the resource was last modified. For example, `Last-Modified: Thu, 19 Oct 2017 09:28:00 GMT`.
- A cache validates a file using `Last-Modified` by sending an `If-Modified-Since` header with a date and time in the request. The origin server compares that date with the `Last-Modified` header of the latest resource. If the resource has not been modified since the specified time, the server returns status code 304 (Not Modified) in its response. If the resource has been modified, the server returns status code 200 (OK) and the updated resource.

## Determining which files can be cached

Not all resources can be cached. The following table shows what resources can be cached, based on the type of HTTP response. Resources delivered with HTTP responses that don't meet all of these conditions cannot be cached. For **Azure CDN Premium from Verizon** only, you can use the rules engine to customize some of these conditions.

	AZURE CDN FROM MICROSOFT	AZURE CDN FROM VERIZON	AZURE CDN FROM AKAMAI
HTTP status codes	200, 203, 206, 300, 301, 410, 416	200	200, 203, 300, 301, 302, 401
HTTP methods	GET, HEAD	GET	GET
File size limits	300 GB	300 GB	- General web delivery optimization: 1.8 GB - Media streaming optimizations: 1.8 GB - Large file optimization: 150 GB

For **Azure CDN Standard from Microsoft** caching to work on a resource, the origin server must support any HEAD and GET HTTP requests and the content-length values must be the same for any HEAD and GET HTTP responses for the asset. For a HEAD request, the origin server must support the HEAD request, and must respond with the same headers as if it had received a GET request.

## Default caching behavior

The following table describes the default caching behavior for the Azure CDN products and their optimizations.

	MICROSOFT: GENERAL WEB DELIVERY	VERIZON: GENERAL WEB DELIVERY	VERIZON: DSA	AKAMAI: GENERAL WEB DELIVERY	AKAMAI: DSA	AKAMAI: LARGE FILE DOWNLOAD	AKAMAI: GENERAL OR VOD MEDIA STREAMING
<b>Honor origin</b>	Yes	Yes	No	Yes	No	Yes	Yes
<b>CDN cache duration</b>	2 days	7 days	None	7 days	None	1 day	1 year

**Honor origin:** Specifies whether to honor the supported cache-directive headers if they exist in the HTTP response from the origin server.

**CDN cache duration:** Specifies the amount of time for which a resource is cached on the Azure CDN. However, if **Honor origin** is Yes and the HTTP response from the origin server includes the cache-directive header `Expires` or `Cache-Control: max-age`, Azure CDN uses the duration value specified by the header instead.

## Next steps

- To learn how to customize and override the default caching behavior on the CDN through caching rules, see [Control Azure CDN caching behavior with caching rules](#).
- To learn how to use query strings to control caching behavior, see [Control Azure CDN caching behavior with query strings](#).

# China content delivery with Azure CDN

7/5/2019 • 2 minutes to read • [Edit Online](#)

Azure Content Delivery Network (CDN) global can serve content to China users with point-of-presence (POP) locations near China or any POP that provides the best performance to requests originating from China. However, if China is a significant market for your customers and they need fast performance, consider using Azure CDN China instead.

Azure CDN China differs from Azure CDN global in that it delivers content from POPs inside of China by partnering with a number of local providers. Due to Chinese compliance and regulation, you must register a separate subscription to use Azure CDN China and your websites need to have an ICP license. The portal and API experience to enable and manage content delivery is identical between Azure CDN global and Azure CDN China.

## Comparison of Azure CDN global and Azure CDN China

Azure CDN global and Azure CDN China have the following features:

- Azure CDN global:
  - Portal: <https://portal.azure.com>
  - Performs content delivery outside of China
  - Four pricing tiers: Microsoft standard, Verizon standard, Verizon premium, and Akamai standard
  - [Documentation](#)
- Azure CDN China:
  - Portal: <https://portal.azure.cn>
  - Performs content delivery inside of China
  - Two pricing tiers: Standard and premium
  - [Documentation](#)

## Next steps

To learn more about Azure CDN China, see:

- [Content Delivery Network features](#)
- [Overview of Azure Content Delivery Network](#)
- [Use the Azure Content Delivery Network](#)
- [Azure service availability in China](#)

# HTTP/2 Support in Azure CDN

7/12/2019 • 2 minutes to read • [Edit Online](#)

HTTP/2 is a major revision to HTTP/1.1. It provides faster web performance, reduced response time, and improved user experience, while maintaining the familiar HTTP methods, status codes, and semantics. Though HTTP/2 is designed to work with HTTP and HTTPS, many client web browsers only support HTTP/2 over TLS.

## HTTP/2 Benefits

The benefits of HTTP/2 include:

- **Multiplexing and concurrency**

Using HTTP 1.1, making multiple resource requests requires multiple TCP connections, and each connection has performance overhead associated with it. HTTP/2 allows multiple resources to be requested on a single TCP connection.

- **Header compression**

By compressing the HTTP headers for served resources, time on the wire is reduced significantly.

- **Stream dependencies**

Stream dependencies allow the client to indicate to the server which resources have priority.

## HTTP/2 Browser Support

All of the major browsers have implemented HTTP/2 support in their current versions. Non-supported browsers automatically fallback to HTTP/1.1.

BROWSER	MINIMUM VERSION
Microsoft Edge	12
Google Chrome	43
Mozilla Firefox	38
Opera	32
Safari	9

## Enabling HTTP/2 Support in Azure CDN

Currently, HTTP/2 support is active for all Azure CDN profiles. No further action is required from customers.

## Next Steps

To see the benefits of HTTP/2 in action, see [this demo from Akamai](#).

To learn more about HTTP/2, visit the following resources:

- [HTTP/2 specification homepage](#)

- [Official HTTP/2 FAQ](#)
- [Akamai HTTP/2 information](#)

To learn more about Azure CDN's available features, see the [Azure CDN Overview](#).

# Retrieve the current POP IP list for Azure CDN

10/11/2019 • 2 minutes to read • [Edit Online](#)

## Retrieve the current Verizon POP IP list for Azure CDN

You can use the REST API to retrieve the set of IPs for Verizon's point of presence (POP) servers. These POP servers make requests to origin servers that are associated with Azure Content Delivery Network (CDN) endpoints on a Verizon profile ([Azure CDN Standard from Verizon](#) or [Azure CDN Premium from Verizon](#)). Note that this set of IPs is different from the IPs that a client would see when making requests to the POPs.

For the syntax of the REST API operation for retrieving the POP list, see [Edge Nodes - List](#).

## Retrieve the current Microsoft POP IP list for Azure CDN

To lock down your application to accept traffic only from Azure CDN from Microsoft, you will need to set up IP ACLs for your backend. You may also restrict the set of accepted values for the header 'X-Forwarded-Host' sent by Azure CDN from Microsoft. These steps are detailed out as below:

Configure IP ACLing for your backends to accept traffic from Azure CDN from Microsoft's backend IP address space and Azure's infrastructure services only.

- Azure CDN from Microsoft's IPv4 backend IP space: 147.243.0.0/16
- Azure CDN from Microsoft's IPv6 backend IP space: 2a01:111:2050::/44

IP Ranges and Service tags for Microsoft services can be found [here](#)

## Typical use case

For security purposes, you can use this IP list to enforce that requests to your origin server are made only from a valid Verizon POP. For example, if someone discovered the hostname or IP address for a CDN endpoint's origin server, one could make requests directly to the origin server, therefore bypassing the scaling and security capabilities provided by Azure CDN. By setting the IPs in the returned list as the only allowed IPs on an origin server, this scenario can be prevented. To ensure that you have the latest POP list, retrieve it at least once a day.

## Next steps

For information about the REST API, see [Azure CDN REST API](#).

# Migrate an Azure CDN profile from Standard Verizon to Premium Verizon

11/14/2019 • 2 minutes to read • [Edit Online](#)

When you create an Azure Content Delivery Network (CDN) profile to manage your endpoints, Azure CDN offers four different products for you to choose from. For information about the different products and their available features, see [Compare Azure CDN product features](#).

If you've created an **Azure CDN Standard from Verizon** profile and are using it to manage your CDN endpoints, you have the option to upgrade it to an **Azure CDN Premium from Verizon** profile. When you upgrade, your CDN endpoints and all of your data will be preserved.

## IMPORTANT

Once you've upgraded to an **Azure CDN Premium from Verizon** profile, you cannot later convert it back to an **Azure CDN Standard from Verizon** profile.

To upgrade an **Azure CDN Standard from Verizon** profile, contact [Microsoft Support](#).

## Profile comparison

**Azure CDN Premium from Verizon** profiles have the following key differences from **Azure CDN Standard from Verizon** profiles:

- For certain Azure CDN features such as [compression](#), [caching rules](#), and [geo filtering](#), you cannot use the Azure CDN interface; you must use the Verizon portal via the **Manage** button.
- API: Unlike with Standard Verizon, you cannot use the API to control those features that are accessed from the Premium Verizon portal. However, you can use the API to control other common features, such as creating/deleting an endpoint, purging/loading cached assets, and enabling/disabling a custom domain.
- Pricing: Premium Verizon has a different pricing structure for data transfers than Standard Verizon. For more information, see [Content Delivery Network pricing](#).

**Azure CDN Premium from Verizon** profiles have the following additional features:

- **Token authentication:** Allows users to obtain and use a token to fetch secure resources.
- **Rules engine:** Enables you to customize how HTTP requests are handled.
- Advanced analytics tools:
  - [Detailed HTTP analytics](#)
  - [Edge performance analytics](#)
  - [Real-time analytics](#)

## Next steps

To learn more about the rules engine, see [Azure CDN rules engine reference](#).

# Azure CDN DDoS Protection

7/5/2019 • 2 minutes to read • [Edit Online](#)

A content delivery network provides DDoS protection by design. In addition to the global capacity to absorb volumetric attacks, Azure CDN has additional DDoS protection as outlined below, for no extra cost.

## Azure CDN from Microsoft

Azure CDN from Microsoft is protected by [Azure Basic DDoS](#). It is integrated into the Azure CDN from Microsoft platform by default and at no additional cost. The full scale and capacity of Azure CDN from Microsoft's globally deployed network provides defense against common network layer attacks through always-on traffic monitoring and real-time mitigation. Basic DDoS protection also defends against the most common, frequently occurring Layer 7 DNS Query Floods and Layer 3 and 4 volumetric attacks that target CDN endpoints. This service also has a proven track record in protecting Microsoft's enterprise and consumer services from large-scale attacks.

## Azure CDN from Verizon

Azure CDN from Verizon is protected by Verizon's proprietary DDoS mitigation platform. It's integrated into Azure CDN from Verizon by default and at no additional cost. It provides basic protection against the most common, frequently occurring Layer 7 DNS Query Floods and Layer 3 and 4 volumetric attacks that target CDN endpoints.

## Azure CDN from Akamai

Azure CDN from Akamai is protected by Akamai's proprietary DDoS mitigation platform. It's integrated into Azure CDN from Akamai by default and at no additional cost. It provides basic protection against the most common, frequently occurring Layer 7 DNS Query Floods and Layer 3 and 4 volumetric attacks that target CDN endpoints.

## Next steps

Learn more about [Azure DDoS](#).

# Optimize Azure CDN for the type of content delivery

7/5/2019 • 5 minutes to read • [Edit Online](#)

When you deliver content to a large global audience, it's critical to ensure the optimized delivery of your content. [Azure Content Delivery Network \(CDN\)](#) can optimize the delivery experience based on the type of content you have. The content can be a website, a live stream, a video, or a large file for download. When you create a CDN endpoint, you specify a scenario in the **Optimized for** option. Your choice determines which optimization is applied to the content delivered from the CDN endpoint.

Optimization choices are designed to use best-practice behaviors to improve content delivery performance and better origin offload. Your scenario choices affect performance by modifying configurations for partial caching, object chunking, and the origin failure retry policy.

This article provides an overview of various optimization features and when you should use them. For more information on features and limitations, see the respective articles on each individual optimization type.

## NOTE

When you create a CDN endpoint, the **Optimized for** options can vary based on the type of profile the endpoint is created in. Azure CDN providers apply enhancement in different ways, depending on the scenario.

## Provider options

**Azure CDN Standard from Microsoft** profiles supports the following optimizations:

- [General web delivery](#). This optimization is also used for media streaming and large file download.

## NOTE

Dynamic site acceleration from Microsoft is offered via [Azure Front Door Service](#).

**Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles support the following optimizations:

- [General web delivery](#). This optimization is also used for media streaming and large file download.
- [Dynamic site acceleration](#)

**Azure CDN Standard from Akamai** profiles support the following optimizations:

- [General web delivery](#)
- [General media streaming](#)
- [Video-on-demand media streaming](#)
- [Large file download](#)
- [Dynamic site acceleration](#)

Microsoft recommends that you test performance variations between different providers to select the optimal provider for your delivery.

# Select and configure optimization types

When you create a CDN endpoint, select an optimization type that best matches the scenario and type of content that you want the endpoint to deliver. **General web delivery** is the default selection. For existing **Azure CDN Standard from Akamai** endpoints only, you can update the optimization option at any time. This change doesn't interrupt delivery from Azure CDN.

1. In an **Azure CDN Standard from Akamai** profile, select an endpoint.

The screenshot shows the 'Endpoints' section of the Azure CDN Standard from Akamai blade. It lists two endpoints:

HOSTNAME	STATUS	PROTOCOL	ORIGIN TYPE	CUSTOM DOMAINS
azuredge.net	Running	HTTP, HTTPS	Storage	
azuredge.net	Running	HTTP, HTTPS	Storage	

2. Under SETTINGS, select **Optimization**. Then, select a type from the **Optimized for** drop-down list.

The screenshot shows the 'Optimization' settings page. The 'Optimized for' dropdown menu is open, displaying several options:

- General web delivery
- Large file download
- Video on demand media streaming
- General media streaming
- General web delivery

## Optimization for specific scenarios

You can optimize the CDN endpoint for one of these scenarios.

### General web delivery

General web delivery is the most common optimization option. It's designed for general web content optimization, such as webpages and web applications. This optimization also can be used for file and video downloads.

A typical website contains static and dynamic content. Static content includes images, JavaScript libraries, and style

sheets that can be cached and delivered to different users. Dynamic content is personalized for an individual user, such as news items that are tailored to a user profile. Dynamic content, such as shopping cart contents, isn't cached because it's unique to each user. General web delivery can optimize your entire website.

#### NOTE

If you are using an **Azure CDN Standard from Akamai** profile, select this optimization type if your average file size is smaller than 10 MB. Otherwise, if your average file size is larger than 10 MB, select **Large file download** from the **Optimized for** drop-down list.

## General media streaming

If you need to use the endpoint for live streaming and video-on-demand streaming, select the general media streaming optimization type.

Media streaming is time-sensitive, because packets that arrive late on the client, such as frequent buffering of video content, can cause a degraded viewing experience. Media streaming optimization reduces the latency of media content delivery and provides a smooth streaming experience for users.

This scenario is common for Azure media service customers. When you use Azure media services, you get a single streaming endpoint that can be used for both live and on-demand streaming. With this scenario, customers don't need to switch to another endpoint when they change from live to on-demand streaming. General media streaming optimization supports this type of scenario.

For **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, and **Azure CDN Premium from Verizon**, use the general web delivery optimization type to deliver general streaming media content.

For more information about media streaming optimization, see [Media streaming optimization](#).

## Video-on-demand media streaming

Video-on-demand media streaming optimization improves video-on-demand streaming content. If you use an endpoint for video-on-demand streaming, use this option.

For **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, and **Azure CDN Premium from Verizon** profiles, use the general web delivery optimization type to deliver video-on-demand streaming media content.

For more information about media streaming optimization, see [Media streaming optimization](#).

#### NOTE

If the CDN endpoint primarily serves video-on-demand content, use this optimization type. The major difference between this optimization type and the general media streaming optimization type is the connection retry time-out. The time-out is much shorter to work with live streaming scenarios.

## Large file download

For **Azure CDN Standard from Akamai** profiles, large file downloads are optimized for content larger than 10 MB. If your average file size is smaller than 10 MB, use general web delivery. If your average files sizes are consistently larger than 10 MB, it might be more efficient to create a separate endpoint for large files. For example, firmware or software updates typically are large files. To deliver files larger than 1.8 GB, the large file download optimization is required.

For **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, and **Azure CDN Premium from Verizon** profiles, use the general web delivery optimization type to deliver large file download content. There is no limitation on file download size.

For more information about large file optimization, see [Large file optimization](#).

## Dynamic site acceleration

Dynamic site acceleration (DSA) is available for **Azure CDN Standard from Akamai**, **Azure CDN Standard from Verizon**, and **Azure CDN Premium from Verizon** profiles. This optimization involves an additional fee to use; for more information, see [Content Delivery Network pricing](#).

### NOTE

Dynamic site acceleration from Microsoft is offered via [Azure Front Door Service](#) which is a global [anycast](#) service leveraging Microsoft's private global network to deliver your app workloads.

DSA includes various techniques that benefit the latency and performance of dynamic content. Techniques include route and network optimization, TCP optimization, and more.

You can use this optimization to accelerate a web app that includes numerous responses that aren't cacheable. Examples are search results, checkout transactions, or real-time data. You can continue to use core Azure CDN caching capabilities for static data.

For more information about dynamic site acceleration, see [Dynamic site acceleration](#).

# Large file download optimization with Azure CDN

7/5/2019 • 8 minutes to read • [Edit Online](#)

File sizes of content delivered over the internet continue to grow due to enhanced functionality, improved graphics, and rich media content. This growth is driven by many factors: broadband penetration, larger inexpensive storage devices, widespread increase of high-definition video, and internet-connected devices (IoT). A fast and efficient delivery mechanism for large files is critical to ensure a smooth and enjoyable consumer experience.

Delivery of large files has several challenges. First, the average time to download a large file can be significant because applications might not download all data sequentially. In some cases, applications might download the last part of a file before the first part. When only a small amount of a file is requested or a user pauses a download, the download can fail. The download also might be delayed until after the content delivery network (CDN) retrieves the entire file from the origin server.

Second, the latency between a user's machine and the file determines the speed at which they can view content. In addition, network congestion and capacity problems also affect throughput. Greater distances between servers and users create additional opportunities for packet loss to occur, which reduces quality. The reduction in quality caused by limited throughput and increased packet loss might increase the wait time for a file download to finish.

Third, many large files are not delivered in their entirety. Users might cancel a download halfway through or watch only the first few minutes of a long MP4 video. Therefore, software and media delivery companies want to deliver only the portion of a file that's requested. Efficient distribution of the requested portions reduces the egress traffic from the origin server. Efficient distribution also reduces the memory and I/O pressure on the origin server.

## Optimize for delivery of large files with Azure CDN from Microsoft

**Azure CDN Standard from Microsoft** endpoints deliver large files without a cap on file size. Additional features are turned on by default to make delivery of large files faster.

### Object chunking

**Azure CDN Standard from Microsoft** uses a technique called object chunking. When a large file is requested, the CDN retrieves smaller pieces of the file from the origin. After the CDN POP server receives a full or byte-range file request, the CDN edge server requests the file from the origin in chunks of 8 MB.

After the chunk arrives at the CDN edge, it's cached and immediately served to the user. The CDN then prefetches the next chunk in parallel. This prefetch ensures that the content stays one chunk ahead of the user, which reduces latency. This process continues until the entire file is downloaded (if requested), all byte ranges are available (if requested), or the client terminates the connection.

For more information on the byte-range request, see [RFC 7233](#).

The CDN caches any chunks as they're received. The entire file doesn't need to be cached on the CDN cache. Subsequent requests for the file or byte ranges are served from the CDN cache. If not all the chunks are cached on the CDN, prefetch is used to request chunks from the origin. This optimization relies on the ability of the origin server to support byte-range requests; if the origin server doesn't support byte-range requests, this optimization isn't effective.

### Conditions for large file optimization

Large file optimization features for **Azure CDN Standard from Microsoft** are turned on by default when you use the general web delivery optimization type. There are no limits on maximum file size.

# Optimize for delivery of large files with Azure CDN from Verizon

**Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** endpoints deliver large files without a cap on file size. Additional features are turned on by default to make delivery of large files faster.

## Complete cache fill

The default complete cache fill feature enables the CDN to pull a file into the cache when an initial request is abandoned or lost.

Complete cache fill is most useful for large assets. Typically, users don't download them from start to finish. They use progressive download. The default behavior forces the edge server to initiate a background fetch of the asset from the origin server. Afterward, the asset is in the edge server's local cache. After the full object is in the cache, the edge server fulfills byte-range requests to the CDN for the cached object.

The default behavior can be disabled through the rules engine in **Azure CDN Premium from Verizon**.

## Peer cache fill hot-filing

The default peer cache fill hot-filing feature uses a sophisticated proprietary algorithm. It uses additional edge caching servers based on bandwidth and aggregate requests metrics to fulfill client requests for large, highly popular objects. This feature prevents a situation in which large numbers of extra requests are sent to a user's origin server.

## Conditions for large file optimization

Large file optimization features for **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** are turned on by default when you use the general web delivery optimization type. There are no limits on maximum file size.

# Optimize for delivery of large files with Azure CDN Standard from Akamai

**Azure CDN Standard from Akamai** profile endpoints offer a feature that delivers large files efficiently to users across the globe at scale. The feature reduces latencies because it reduces the load on the origin servers.

The large file optimization type feature turns on network optimizations and configurations to deliver large files faster and more responsively. General web delivery with **Azure CDN Standard from Akamai** endpoints caches files only below 1.8 GB and can tunnel (not cache) files up to 150 GB. Large file optimization caches files up to 150 GB.

Large file optimization is effective when certain conditions are satisfied. Conditions include how the origin server operates and the sizes and types of the files that are requested.

## Configure an Akamai CDN endpoint to optimize delivery of large files

You can configure your **Azure CDN Standard from Akamai** endpoint to optimize delivery for large files via the Azure portal. You can also use the REST APIs or any of the client SDKs to do this. The following steps show the process via the Azure portal for an **Azure CDN Standard from Akamai** profile:

1. To add a new endpoint, on an Akamai **CDN profile** page, select **Endpoint**.

2. In the **Optimized for** drop-down list, select **Large file download**.

After you create the CDN endpoint, it applies the large file optimizations for all files that match certain criteria. The following section describes this process.

### Object chunking

Large file optimization with **Azure CDN Standard from Akamai** uses a technique called object chunking. When a large file is requested, the CDN retrieves smaller pieces of the file from the origin. After the CDN POP server receives a full or byte-range file request, it checks whether the file type is supported for this optimization. It also checks whether the file type meets the file size requirements. If the file size is greater than 10 MB, the CDN edge server requests the file from the origin in chunks of 2 MB.

After the chunk arrives at the CDN edge, it's cached and immediately served to the user. The CDN then prefetches the next chunk in parallel. This prefetch ensures that the content stays one chunk ahead of the user, which reduces

latency. This process continues until the entire file is downloaded (if requested), all byte ranges are available (if requested), or the client terminates the connection.

For more information on the byte-range request, see [RFC 7233](#).

The CDN caches any chunks as they're received. The entire file doesn't need to be cached on the CDN cache. Subsequent requests for the file or byte ranges are served from the CDN cache. If not all the chunks are cached on the CDN, prefetch is used to request chunks from the origin. This optimization relies on the ability of the origin server to support byte-range requests; if the origin server doesn't support byte-range requests, this optimization isn't effective.

## Caching

Large file optimization uses different default caching-expiration times from general web delivery. It differentiates between positive caching and negative caching based on HTTP response codes. If the origin server specifies an expiration time via a cache-control or expires header in the response, the CDN honors that value. When the origin doesn't specify and the file matches the type and size conditions for this optimization type, the CDN uses the default values for large file optimization. Otherwise, the CDN uses defaults for general web delivery.

	GENERAL WEB	LARGE FILE OPTIMIZATION
Caching: Positive HTTP 200, 203, 300, 301, 302, and 410	7 days	1 day
Caching: Negative HTTP 204, 305, 404, and 405	None	1 second

## Deal with origin failure

The origin read-timeout length increases from two seconds for general web delivery to two minutes for the large file optimization type. This increase accounts for the larger file sizes to avoid a premature timeout connection.

When a connection times out, the CDN retries a number of times before it sends a "504 - Gateway Timeout" error to the client.

## Conditions for large file optimization

The following table lists the set of criteria to be satisfied for large file optimization:

CONDITION	VALUES
Supported file types	3g2, 3gp, asf, avi, bz2, dmg, exe, f4v, flv, gz, hdp, iso, jxr, m4v, mkv, mov, mp4, mpeg, mpg, mts, pkg, qt, rm, swf, tar, tgz, wdp, webm, webp, wma, wmv, zip
Minimum file size	10 MB
Maximum file size	150 GB
Origin server characteristics	Must support byte-range requests

## Additional considerations

Consider the following additional aspects for this optimization type:

- The chunking process generates additional requests to the origin server. However, the overall volume of

data delivered from the origin is much smaller. Chunking results in better caching characteristics at the CDN.

- Memory and I/O pressure are reduced at the origin because smaller pieces of the file are delivered.
- For chunks cached at the CDN, there are no additional requests to the origin until the content expires or it's evicted from the cache.
- Users can make range requests to the CDN, which are treated like any normal file. Optimization applies only if it's a valid file type and the byte range is between 10 MB and 150 GB. If the average file size requested is smaller than 10 MB, use general web delivery instead.

# Media streaming optimization with Azure CDN

7/5/2019 • 5 minutes to read • [Edit Online](#)

Use of high-definition video is increasing on the internet, which creates difficulties for efficient delivery of large files. Customers expect smooth playback of video on demand or live video assets on a variety of networks and clients all over the world. A fast and efficient delivery mechanism for media streaming files is critical to ensure a smooth and enjoyable consumer experience.

Live streaming media is especially difficult to deliver because of the large sizes and number of concurrent viewers. Long delays cause users to leave. Because live streams can't be cached ahead of time and large latencies aren't acceptable to viewers, video fragments must be delivered in a timely manner.

The request patterns of streaming also provide some new challenges. When a popular live stream or a new series is released for video on demand, thousands to millions of viewers might request the stream at the same time. In this case, smart request consolidation is vital to not overwhelm the origin servers when the assets aren't cached yet.

## Media streaming optimizations for Azure CDN from Microsoft

**Azure CDN Standard from Microsoft** endpoints deliver streaming media assets directly by using the general web delivery optimization type.

Media streaming optimization for **Azure CDN Standard from Microsoft** is effective for live or video-on-demand streaming media that uses individual media fragments for delivery. This process is different from a single large asset transferred via progressive download or by using byte-range requests. For information on that style of media delivery, see [Large file download optimization with Azure CDN](#).

The general media delivery or video-on-demand media delivery optimization types use Azure Content Delivery Network (CDN) with back-end optimizations to deliver media assets faster. They also use configurations for media assets based on best practices learned over time.

### Partial cache sharing

Partial cache sharing allows the CDN to serve partially cached content to new requests. For example, if the first request to the CDN results in a cache miss, the request is sent to the origin. Although this incomplete content is loaded into the CDN cache, other requests to the CDN can start getting this data.

## Media streaming optimizations for Azure CDN from Verizon

**Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** endpoints deliver streaming media assets directly by using the general web delivery optimization type. A few features on the CDN directly assist in delivering media assets by default.

### Partial cache sharing

Partial cache sharing allows the CDN to serve partially cached content to new requests. For example, if the first request to the CDN results in a cache miss, the request is sent to the origin. Although this incomplete content is loaded into the CDN cache, other requests to the CDN can start getting this data.

### Cache fill wait time

The cache fill wait time feature forces the edge server to hold any subsequent requests for the same resource until HTTP response headers arrive from the origin server. If HTTP response headers from the origin arrive before the timer expires, all requests that were put on hold are served out of the growing cache. At the same time, the cache

is filled by data from the origin. By default, the cache fill wait time is set to 3,000 milliseconds.

## Media streaming optimizations for Azure CDN from Akamai

**Azure CDN Standard from Akamai** offers a feature that delivers streaming media assets efficiently to users across the globe at scale. The feature reduces latencies because it reduces the load on the origin servers. This feature is available with the standard Akamai pricing tier.

Media streaming optimization for **Azure CDN Standard from Akamai** is effective for live or video-on-demand streaming media that uses individual media fragments for delivery. This process is different from a single large asset transferred via progressive download or by using byte-range requests. For information on that style of media delivery, see [Large file optimization](#).

The general media delivery or video-on-demand media delivery optimization types use a CDN with back-end optimizations to deliver media assets faster. They also use configurations for media assets based on best practices learned over time.

### Configure an Akamai CDN endpoint to optimize media streaming

You can configure your content delivery network (CDN) endpoint to optimize delivery for large files via the Azure portal. You can also use the REST APIs or any of the client SDKs to do this. The following steps show the process via the Azure portal for an **Azure CDN Standard from Akamai** profile:

1. To add a new endpoint, on an Akamai **CDN profile** page, select **Endpoint**.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various service icons like App Services, Function Apps, and Storage accounts. The main area is titled 'CDNDemoAkamai' and shows the 'Endpoints' section. At the top of this section, there's a button labeled '+ Endpoint'. Below it, there's a table with columns for 'HOSTNAME', 'STATUS', 'PROTOCOL', 'ORIGIN TYPE', and 'CUSTOM DOMAINS'. A note says 'No endpoints are associated with this profile'. In the top right corner of the main area, there's a box labeled 'Pricing tier Standard Akamai' with a red border around it.

2. In the **Optimized for** drop-down list, select **Video on demand media streaming** for video-on-demand assets. If you do a combination of live and video-on-demand streaming, select **General media streaming**.

Add an endpoint X

Allows configuring content delivery behavior an...

\* Name

.azureedge.net

\* Origin type

\* Origin hostname

Origin path  /Path

Origin host header

Protocol  HTTP  80

HTTPS  443

Optimized for

General web delivery

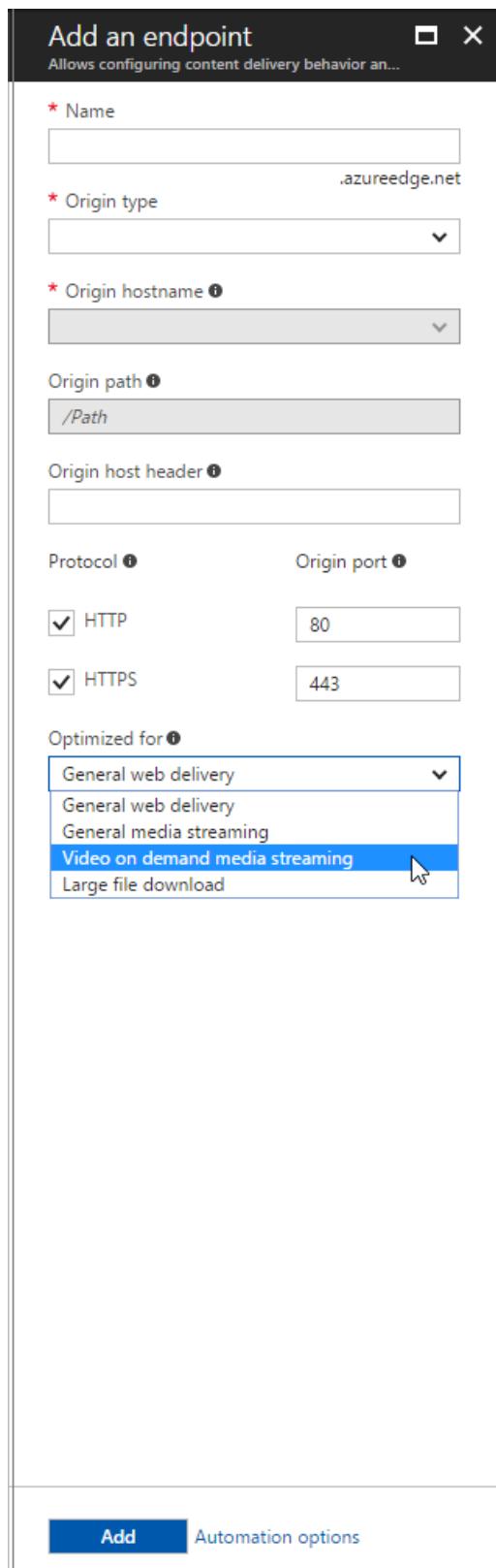
General web delivery

General media streaming

Video on demand media streaming Selected

Large file download

Add Automation options



After you create the endpoint, it applies the optimization for all files that match certain criteria. The following section describes this process.

## Caching

If **Azure CDN Standard from Akamai** detects that the asset is a streaming manifest or fragment, it uses different caching expiration times from general web delivery. (See the full list in the following table.) As always, cache-control or Expires headers sent from the origin are honored. If the asset is not a media asset, it caches by using the expiration times for general web delivery.

The short negative caching time is useful for origin offload when many users request a fragment that doesn't exist yet. An example is a live stream where the packets aren't available from the origin that second. The longer caching

interval also helps offload requests from the origin because video content isn't typically modified.

	GENERAL WEB DELIVERY	GENERAL MEDIA STREAMING	VIDEO-ON-DEMAND MEDIA STREAMING
Caching: Positive HTTP 200, 203, 300, 301, 302, and 410	7 days	365 days	365 days
Caching: Negative HTTP 204, 305, 404, and 405	None	1 second	1 second

### Deal with origin failure

General media delivery and video-on-demand media delivery also have origin timeouts and a retry log based on best practices for typical request patterns. For example, because general media delivery is for live and video-on-demand media delivery, it uses a shorter connection timeout due to the time-sensitive nature of live streaming.

When a connection times out, the CDN retries a number of times before it sends a "504 - Gateway Timeout" error to the client.

When a file matches the file type and size conditions list, the CDN uses the behavior for media streaming. Otherwise, it uses general web delivery.

### Conditions for media streaming optimization

The following table lists the set of criteria to be satisfied for media streaming optimization:

SUPPORTED STREAMING TYPES	FILE EXTENSIONS
Apple HLS	m3u8, m3u, m3ub, key, ts, aac
Adobe HDS	f4m, f4x, drmmeta, bootstrap, f4f, Seg-Frag URL structure (matching regex: ^(.*)Seq(\d+)-Frag(\d+)
DASH	mpd, dash, divx, ismv, m4s, m4v, mp4, mp4v, sidx, webm, mp4a, m4a, isma
Smooth streaming	/manifest/, /QualityLevels/Fragments/

# Dynamic site acceleration via Azure CDN

7/5/2019 • 8 minutes to read • [Edit Online](#)

With the explosion of social media, electronic commerce, and the hyper-personalized web, a rapidly increasing percentage of the content served to end users is generated in real time. Users expect a fast, reliable, and personalized web experience, independent of their browser, location, device, or network. However, the very innovations that make these experiences so engaging also slow page downloads and put the quality of the consumer experience at risk.

Standard content delivery network (CDN) capability includes the ability to cache files closer to end users to speed up delivery of static files. However, with dynamic web applications, caching that content in edge locations isn't possible because the server generates the content in response to user behavior. Speeding up the delivery of such content is more complex than traditional edge caching and requires an end-to-end solution that finely tunes each element along the entire data path from inception to delivery. With Azure CDN dynamic site acceleration (DSA) optimization, the performance of web pages with dynamic content is measurably improved.

**Azure CDN from Akamai** and **Azure CDN from Verizon** both offer DSA optimization through the **Optimized for** menu during endpoint creation. Dynamic site acceleration from Microsoft is offered via [Azure Front Door Service](#).

## IMPORTANT

For **Azure CDN from Akamai** profiles, you are allowed to change the optimization of a CDN endpoint after it has been created.

For **Azure CDN from Verizon** profiles, you cannot change the optimization of a CDN endpoint after it has been created.

## CDN endpoint configuration to accelerate delivery of dynamic files

To configure a CDN endpoint to optimize delivery of dynamic files, you can either use the Azure portal, the REST APIs, or any of the client SDKs to do the same thing programmatically.

### To configure a CDN endpoint for DSA optimization by using the Azure portal:

1. In the **CDN profile** page, select **Endpoint**.

The screenshot shows the Azure portal interface for a CDN profile named 'CDNDemoAkamai'. On the left, there's a sidebar with various navigation links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS (Locks, Automation script), GENERAL (Quickstart, Properties), and a search bar at the top. The main content area is titled 'Essentials' and shows details about the resource group 'CdnDemoRG', status 'Active', location 'West US', subscription name 'AZURE-CDN-TEST', and subscription ID '<subscription ID>'. Below this is a section for 'Endpoints' which currently shows 0 endpoints. A table header for 'HOSTNAME', 'STATUS', 'PROTOCOL', 'ORIGIN TYPE', and 'CUSTOM DOMAINS' is present, followed by a message 'No endpoints are associated with this profile'.

The **Add an endpoint** pane appears.

2. Under **Optimized for**, select **Dynamic site acceleration**.

The screenshot shows the 'Add an endpoint' configuration pane. It includes fields for Name (with a placeholder '.azureedge.net'), Origin type (selected as 'Azure website'), Origin hostname (selected as 'www'), Origin path ('/Path'), Origin host header (''), Protocol ('HTTP' checked, port 80) and Origin port ('HTTPS' checked, port 443). The 'Optimized for' dropdown is set to 'Dynamic site acceleration' and is highlighted with a red box. A warning message below it states: 'You will incur additional charges on traffic for this endpoint if you continue with 'Dynamic Site Acceleration' optimization type.' A 'Learn more' link is provided. At the bottom, there's a 'Probe path' field containing '/path/check.txt' and a note explaining Azure CDN uses it to detect the optimal route. Finally, there are 'Add' and 'Automation options' buttons at the bottom.

3. For **Probe path**, enter a valid path to a file.

Probe path is a feature specific to DSA, and a valid path is required for creation. DSA uses a small *probe path* file placed on the origin server to optimize network routing configurations for the CDN. For the probe path file, you can download and upload the sample file to your site, or use an existing asset on your origin that is about 10 KB in size.

4. Enter the other required endpoint options (for more information, see [Create a new CDN endpoint](#)), then select **Add**.

After the CDN endpoint is created, it applies the DSA optimizations for all files that match certain criteria.

**To configure an existing endpoint for DSA (Azure CDN from Akamai profiles only):**

1. In the **CDN profile** page, select the endpoint you want to modify.

2. From the left pane, select **Optimization**.

The **Optimization** page appears.

3. Under **Optimized for**, select **Dynamic site acceleration**, then select **Save**.

**NOTE**

DSA incurs extra charges. For more information, see [Content Delivery Network pricing](#).

## DSA Optimization using Azure CDN

Dynamic Site Acceleration on Azure CDN speeds up delivery of dynamic assets by using the following techniques:

- [Route optimization](#)
- [TCP optimizations](#)
- [Object prefetch \(Azure CDN from Akamai only\)](#)
- [Adaptive image compression \(Azure CDN from Akamai only\)](#)

### Route Optimization

Route optimization is important because the Internet is a dynamic place, where traffic and temporarily outages are constantly changing the network topology. The Border Gateway Protocol (BGP) is the routing protocol of the Internet, but there may be faster routes via intermediary Point of Presence (PoP) servers.

Route optimization chooses the most optimal path to the origin so that a site is continuously accessible and dynamic content is delivered to end users via the fastest and most reliable route possible.

The Akamai network uses techniques to collect real-time data and compare various paths through different nodes in the Akamai server, as well as the default BGP route across the open Internet to determine the fastest route between the origin and the CDN edge. These techniques avoid Internet congestion points and long routes.

Similarly, the Verizon network uses a combination of Anycast DNS, high capacity support PoPs, and health checks, to determine the best gateways to best route data from the client to the origin.

As a result, fully dynamic and transactional content is delivered more quickly and more reliably to end users, even when it is uncachable.

### TCP Optimizations

Transmission Control Protocol (TCP) is the standard of the Internet protocol suite used to deliver information between applications on an IP network. By default, several back-and-forth requests are required to set up a TCP connection, as well as limits to avoid network congestions, which result in inefficiencies at scale. [Azure CDN from](#)

**Akamai** handles this problem by optimizing in three areas:

- [Eliminating TCP slow start](#)
- [Leveraging persistent connections](#)
- [Tuning TCP packet parameters](#)

#### **Eliminating TCP slow start**

TCP *slow start* is an algorithm of the TCP protocol that prevents network congestion by limiting the amount of data sent over the network. It starts off with small congestion window sizes between sender and receiver until the maximum is reached or packet loss is detected.

Both **Azure CDN from Akamai** and **Azure CDN from Verizon** profiles eliminate TCP slow start with the following three steps:

1. Health and bandwidth monitoring is used to measure the bandwidth of connections between edge PoP servers.
2. Metrics are shared between edge PoP servers so that each server is aware of the network conditions and server health of the other PoPs around them.
3. The CDN edge servers make assumptions about some transmission parameters, such as what the optimal window size should be when communicating with other CDN edge servers in its proximity. This step means that the initial congestion window size can be increased if the health of the connection between the CDN edge servers is capable of higher packet data transfers.

#### **Leveraging persistent connections**

Using a CDN, fewer unique machines connect to your origin server directly compared with users connecting directly to your origin. Azure CDN also pools user requests together to establish fewer connections with the origin.

As previously mentioned, several handshake requests are required to establish a TCP connection. Persistent connections, which are implemented by the `Keep-Alive` HTTP header, reuse existing TCP connections for multiple HTTP requests to save round-trip times and speed up delivery.

**Azure CDN from Verizon** also sends periodic keep-alive packets over the TCP connection to prevent an open connection from being closed.

#### **Tuning TCP packet parameters**

**Azure CDN from Akamai** tunes the parameters that govern server-to-server connections and reduces the amount of long-haul round trips required to retrieve content embedded in the site by using the following techniques:

- Increasing the initial congestion window so that more packets can be sent without waiting for an acknowledgement.
- Decreasing the initial retransmit timeout so that a loss is detected, and retransmission occurs more quickly.
- Decreasing the minimum and maximum retransmit timeout to reduce the wait time before assuming packets were lost in transmission.

#### **Object prefetch (Azure CDN from Akamai only)**

Most websites consist of an HTML page, which references various other resources such as images and scripts. Typically, when a client requests a webpage, the browser first downloads and parses the HTML object, and then makes additional requests to linked assets that are required to fully load the page.

*Prefetch* is a technique to retrieve images and scripts embedded in the HTML page while the HTML is served to the browser, and before the browser even makes these object requests.

With the prefetch option turned on at the time when the CDN serves the HTML base page to the client's browser, the CDN parses the HTML file and make additional requests for any linked resources and store it in its cache. When the client makes the requests for the linked assets, the CDN edge server already has the requested objects

and can serve them immediately without a round trip to the origin. This optimization benefits both cacheable and non-cacheable content.

#### Adaptive image compression (Azure CDN from Akamai only)

Some devices, especially mobile ones, experience slower network speeds from time to time. In these scenarios, it is more beneficial for the user to receive smaller images in their webpage more quickly rather than waiting a long time for full resolution images.

This feature automatically monitors network quality, and employs standard JPEG compression methods when network speeds are slower to improve delivery time.

ADAPTIVE IMAGE COMPRESSION	FILE EXTENSIONS
JPEG compression	.jpg, .jpeg, .jpe, .jig, .jgig, .jgi

## Caching

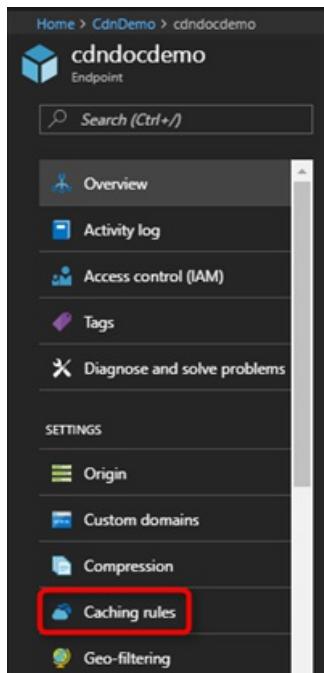
With DSA, caching is turned off by default on the CDN, even when the origin includes `Cache-Control` or `Expires` headers in the response. DSA is typically used for dynamic assets that should not be cached because they are unique to each client. Caching can break this behavior.

If you have a website with a mix of static and dynamic assets, it is best to take a hybrid approach to get the best performance.

For **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles, you can turn on caching for specific DSA endpoints by using [caching rules](#).

To access caching rules:

1. From the **CDN profile** page, under settings, select **Caching rules**.



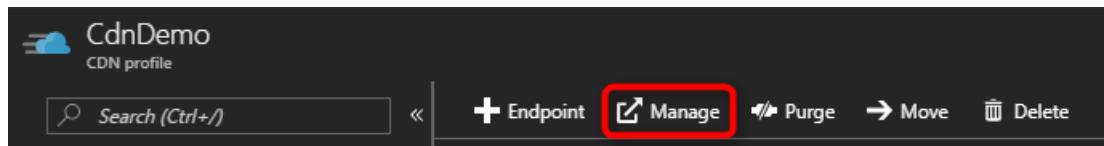
The **Caching rules** page opens.

2. Create a global or custom caching rule to turn on caching for your DSA endpoint.

For **Azure CDN Premium from Verizon** profiles only, you turn on caching for specific DSA endpoints by using the [rules engine](#). Any rules that are created affect only those endpoints of your profile that are optimized for DSA.

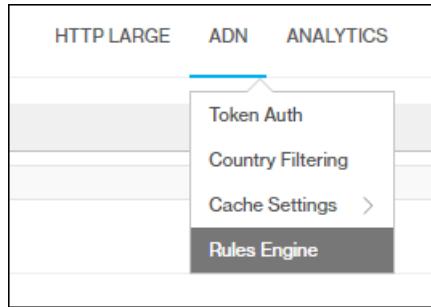
To access the rules engine:

1. From the **CDN profile** page, select **Manage**.



The CDN management portal opens.

2. From the CDN management portal, select **ADN**, then select **Rules Engine**.



Alternatively, you can use two CDN endpoints: one endpoint optimized with DSA to deliver dynamic assets and another endpoint optimized with a static optimization type, such as general web delivery, to delivery cacheable assets. Modify your webpage URLs to link directly to the asset on the CDN endpoint you plan to use.

For example: `mydynamic.azureedge.net/index.html` is a dynamic page and is loaded from the DSA endpoint. The html page references multiple static assets such as JavaScript libraries or images that are loaded from the static CDN endpoint, such as `mystatic.azureedge.net/banner.jpg` and `mystatic.azureedge.net/scripts.js`.

# Create an Azure CDN endpoint

7/5/2019 • 4 minutes to read • [Edit Online](#)

This article describes all the settings for creating an [Azure Content Delivery Network \(CDN\)](#) endpoint in an existing CDN profile. After you've created a profile and an endpoint, you can start delivering content to your customers. For a quickstart on creating a profile and endpoint, see [Quickstart: Create an Azure CDN profile and endpoint](#).

## Prerequisites

Before you can create a CDN endpoint, you must have created at least one CDN profile, which can contain one or more CDN endpoints. To organize your CDN endpoints by internet domain, web application, or some other criteria, you can use multiple profiles. Because CDN pricing is applied at the CDN profile level, you must create multiple CDN profiles if you want to use a mix of Azure CDN pricing tiers. To create a CDN profile, see [Create a new CDN profile](#).

## Log in to the Azure portal

Log in to the [Azure portal](#) with your Azure account.

## Create a new CDN endpoint

1. In the [Azure portal](#), navigate to your CDN profile. You may have pinned it to the dashboard in the previous step. If not, you can find it by selecting **All services**, then selecting **CDN profiles**. In the **CDN profiles** pane, select the profile to which you plan to add your endpoint.

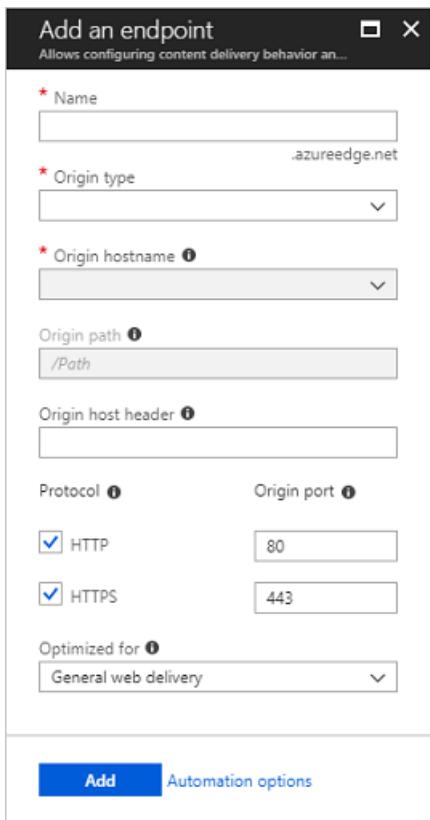
The CDN profile pane appears.

2. Select **Endpoint**.

The screenshot shows the Azure CDN profile pane. At the top, there are several actions: '+ Endpoint' (highlighted with a red box), 'Manage', 'Purge', 'Move', and 'Delete'. Below this is a 'Essentials' section with the following details:

Resource group ( <a href="#">change</a> ) my-resource-group-123	Pricing tier Standard Verizon
Status Active	
Location Central US	
Subscription name ( <a href="#">change</a> ) <subscription name>	
Subscription ID <subscription ID>	

The **Add an endpoint** page appears.



3. For **Name**, enter a unique name for the new CDN endpoint. This name is used to access your cached resources at the domain <endpointname>.azureedge.net.
4. For **Origin type**, choose one of the following origin types:
  - **Storage** for Azure Storage
  - **Cloud service** for Azure Cloud Services
  - **Web App** for Azure Web Apps
  - **Custom origin** for any other publicly accessible origin web server (hosted in Azure or elsewhere)
5. For **Origin hostname**, select or enter your origin server domain. The drop-down lists all available origin servers of the type you specified in step 4. If you selected **Custom origin** as your origin type, enter the domain of your custom origin server.
6. For **Origin path**, enter the path to the resources that you want to cache. To allow caching of any resource at the domain you specified in step 5, leave this setting blank.
7. For **Origin host header**, enter the host header you want Azure CDN to send with each request, or leave the default.

**NOTE**

Some types of origins, such as Azure Storage and Web Apps, require the host header to match the domain of the origin. Unless you have an origin that requires a host header different from its domain, you should leave the default value.

8. For **Protocol** and **Origin port**, specify the protocols and ports to use to access your resources at the origin server. At least one protocol (HTTP or HTTPS) must be selected. Use the CDN-provided domain (<endpointname>.azureedge.net) to access HTTPS content.

#### NOTE

The **Origin port** value determines only the port the endpoint uses to retrieve information from the origin server. The endpoint itself is available only to end clients on the default HTTP and HTTPS ports (80 and 443), regardless of the **Origin port** value.

Endpoints in **Azure CDN from Akamai** profiles do not allow the full TCP port range for origin ports. For a list of origin ports that are not allowed, see [Azure CDN from Akamai Allowed Origin Ports](#).

HTTPS support for Azure CDN custom domains is not supported on **Azure CDN from Akamai** products. For more information, see [Configure HTTPS on an Azure CDN custom domain](#).

9. For **Optimized for**, select an optimization type that best matches the scenario and type of content that you want the endpoint to deliver. For more information, see [Optimize Azure CDN for the type of content delivery](#).

The following optimization type settings are supported, according to profile type:

- **Azure CDN Standard from Microsoft** profiles:
  - [General web delivery](#)
- **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles:
  - [General web delivery](#)
  - [Dynamic site acceleration](#)
- **Azure CDN Standard from Akamai** profiles:
  - [General web delivery](#)
  - [General media streaming](#)
  - [Video on demand media streaming](#)
  - [Large file download](#)
  - [Dynamic site acceleration](#)

10. Select **Add** to create the new endpoint.

After the endpoint is created, it appears in the list of endpoints for the profile.

The screenshot shows the Azure portal interface for managing endpoints. At the top, there are navigation icons for Endpoint, Manage, Purge, Move, and Delete. A success message indicates 'Successfully created endpoint 'my-endpoint-123''. Below this, the 'Essentials' section displays basic details: Resource group (my-resource-group-123), Status (Active), Location (Central US), Subscription name (<subscription name>), and Subscription ID (<subscription ID>). The Pricing tier is listed as Standard Verizon. The 'Endpoints' section shows a summary of 1 endpoint. A table lists the endpoint details: Hostname (my-endpoint-123.azureedge.net), Status (Running), Protocol (HTTP, HTTPS), and Origin Type (Custom origin). The row for the endpoint is highlighted with a red border.

HOSTNAME	STATUS	PROTOCOL	ORIGIN TYPE	CUSTOM DOMAINS
my-endpoint-123.azureedge.net	Running	HTTP, HTTPS	Custom origin	

Because it takes time for the registration to propagate, the endpoint isn't immediately available for use:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes within 90 minutes.

If you attempt to use the CDN domain name before the endpoint configuration has propagated to the point-of-presence (POP) servers, you might receive an HTTP 404 response status. If it's been several hours since you created your endpoint and you're still receiving a 404 response status, see [Troubleshooting Azure CDN endpoints that return a 404 status code](#).

## Clean up resources

To delete an endpoint when it is no longer needed, select it and then select **Delete**.

## Next steps

To learn about custom domains, continue to the tutorial for adding a custom domain to your CDN endpoint.

[Add a custom domain](#)

# Manage Azure CDN with PowerShell

11/21/2019 • 6 minutes to read • [Edit Online](#)

PowerShell provides one of the most flexible methods to manage your Azure CDN profiles and endpoints. You can use PowerShell interactively or by writing scripts to automate management tasks. This tutorial demonstrates several of the most common tasks you can accomplish with PowerShell to manage your Azure CDN profiles and endpoints.

## Prerequisites

### NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

To use PowerShell to manage your Azure CDN profiles and endpoints, you must have the Azure PowerShell module installed. To learn how to install Azure PowerShell and connect to Azure using the `Connect-AzAccount` cmdlet, see [How to install and configure Azure PowerShell](#).

### IMPORTANT

You must log in with `Connect-AzAccount` before you can execute Azure PowerShell cmdlets.

## Listing the Azure CDN cmdlets

You can list all the Azure CDN cmdlets using the `Get-Command` cmdlet.

```
PS C:\> Get-Command -Module Az.Cdn
```

CommandType	Name	Version	Source
Cmdlet	Confirm-AzCdnEndpointProbeURL	1.4.0	Az.Cdn
Cmdlet	Disable-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	Disable-AzCdnCustomDomainHttps	1.4.0	Az.Cdn
Cmdlet	Enable-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	Enable-AzCdnCustomDomainHttps	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnEdgeNode	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnEndpointNameAvailability	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnEndpointResourceUsage	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnOrigin	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnProfile	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnProfileResourceUsage	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnProfileSsoUrl	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnProfileSupportedOptimizationType	1.4.0	Az.Cdn
Cmdlet	Get-AzCdnSubscriptionResourceUsage	1.4.0	Az.Cdn
Cmdlet	New-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	New-AzCdnDeliveryPolicy	1.4.0	Az.Cdn
Cmdlet	New-AzCdnDeliveryRule	1.4.0	Az.Cdn
Cmdlet	New-AzCdnDeliveryRuleAction	1.4.0	Az.Cdn
Cmdlet	New-AzCdnDeliveryRuleCondition	1.4.0	Az.Cdn
Cmdlet	New-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	New-AzCdnProfile	1.4.0	Az.Cdn
Cmdlet	Publish-AzCdnEndpointContent	1.4.0	Az.Cdn
Cmdlet	Remove-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	Remove-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	Remove-AzCdnProfile	1.4.0	Az.Cdn
Cmdlet	Set-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	Set-AzCdnOrigin	1.4.0	Az.Cdn
Cmdlet	Set-AzCdnProfile	1.4.0	Az.Cdn
Cmdlet	Start-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	Stop-AzCdnEndpoint	1.4.0	Az.Cdn
Cmdlet	Test-AzCdnCustomDomain	1.4.0	Az.Cdn
Cmdlet	Unpublish-AzCdnEndpointContent	1.4.0	Az.Cdn

## Getting help

You can get help with any of these cmdlets using the `Get-Help` cmdlet. `Get-Help` provides usage and syntax, and optionally shows examples.

```
PS C:\> Get-Help Get-AzCdnProfile

NAME
    Get-AzCdnProfile

SYNOPSIS
    Gets an Azure CDN profile.

SYNTAX
    Get-AzCdnProfile [-ProfileName <String>] [-ResourceGroupName <String>] [-InformationAction <ActionPreference>] [-InformationVariable <String>] [<CommonParameters>]

DESCRIPTION
    Gets an Azure CDN profile and all related information.

RELATED LINKS

REMARKS
    To see the examples, type: "get-help Get-AzCdnProfile -examples".
    For more information, type: "get-help Get-AzCdnProfile -detailed".
    For technical information, type: "get-help Get-AzCdnProfile -full".
```

## Listing existing Azure CDN profiles

The `Get-AzCdnProfile` cmdlet without any parameters retrieves all your existing CDN profiles.

```
Get-AzCdnProfile
```

This output can be piped to cmdlets for enumeration.

```
# Output the name of all profiles on this subscription.
Get-AzCdnProfile | ForEach-Object { Write-Host $_.Name }

# Return only **Azure CDN from Verizon** profiles.
Get-AzCdnProfile | Where-Object { $_.Sku.Name -eq "Standard_Verizon" }
```

You can also return a single profile by specifying the profile name and resource group.

```
Get-AzCdnProfile -ProfileName CdnDemo -ResourceGroupName CdnDemoRG
```

### TIP

It is possible to have multiple CDN profiles with the same name, so long as they are in different resource groups. Omitting the `ResourceGroupName` parameter returns all profiles with a matching name.

## Listing existing CDN endpoints

`Get-AzCdnEndpoint` can retrieve an individual endpoint or all the endpoints on a profile.

```

# Get a single endpoint.
Get-AzCdnEndpoint -ProfileName CdnDemo -ResourceGroupName CdnDemoRG -EndpointName cdndocdemo

# Get all of the endpoints on a given profile.
Get-AzCdnEndpoint -ProfileName CdnDemo -ResourceGroupName CdnDemoRG

# Return all of the endpoints on all of the profiles.
Get-AzCdnProfile | Get-AzCdnEndpoint

# Return all of the endpoints in this subscription that are currently running.
Get-AzCdnProfile | Get-AzCdnEndpoint | Where-Object { $_.ResourceState -eq "Running" }

```

## Creating CDN profiles and endpoints

`New-AzCdnProfile` and `New-AzCdnEndpoint` are used to create CDN profiles and endpoints. The following SKUs are supported:

- Standard\_Verizon
- Premium\_Verizon
- Custom\_Verizon
- Standard\_Akamai
- Standard\_Microsoft
- Standard\_ChinaCdn

```

# Create a new profile
New-AzCdnProfile -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -Sku Standard_Akamai -Location "Central US"

# Create a new endpoint
New-AzCdnEndpoint -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -Location "Central US" -EndpointName cdnposhdoc -OriginName "Contoso" -OriginHostName "www.contoso.com"

# Create a new profile and endpoint (same as above) in one line
New-AzCdnProfile -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -Sku Standard_Akamai -Location "Central US" | New-AzCdnEndpoint -EndpointName cdnposhdoc -OriginName "Contoso" -OriginHostName "www.contoso.com"

```

## Checking endpoint name availability

`Get-AzCdnEndpointNameAvailability` returns an object indicating if an endpoint name is available.

```

# Retrieve availability
$availability = Get-AzCdnEndpointNameAvailability -EndpointName "cdnposhdoc"

# If available, write a message to the console.
If($availability.NameAvailable) { Write-Host "Yes, that endpoint name is available." }
Else { Write-Host "No, that endpoint name is not available." }

```

## Adding a custom domain

`New-AzCdnCustomDomain` adds a custom domain name to an existing endpoint.

## IMPORTANT

You must set up the CNAME with your DNS provider as described in [How to map Custom Domain to Content Delivery Network \(CDN\) endpoint](#). You can test the mapping before modifying your endpoint using `Test-AzCdnCustomDomain`.

```
# Get an existing endpoint
$endpoint = Get-AzCdnEndpoint -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -EndpointName cdnposhdoc

# Check the mapping
$result = Test-AzCdnCustomDomain -CdnEndpoint $endpoint -CustomDomainHostName "cdn.contoso.com"

# Create the custom domain on the endpoint
If($result.CustomDomainValidated){ New-AzCdnCustomDomain -CustomDomainName Contoso -HostName "cdn.contoso.com"
-CdnEndpoint $endpoint }
```

## Modifying an endpoint

`Set-AzCdnEndpoint` modifies an existing endpoint.

```
# Get an existing endpoint
$endpoint = Get-AzCdnEndpoint -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -EndpointName cdnposhdoc

# Set up content compression
$endpoint.IsCompressionEnabled = $true
$endpoint.ContentTypesToCompress = "text/javascript","text/css","application/json"

# Save the changed endpoint and apply the changes
Set-AzCdnEndpoint -CdnEndpoint $endpoint
```

## Purging/Pre-loading CDN assets

`Unpublish-AzCdnEndpointContent` purges cached assets, while `Publish-AzCdnEndpointContent` pre-loads assets on supported endpoints.

```
# Purge some assets.
Unpublish-AzCdnEndpointContent -ProfileName CdnDemo -ResourceGroupName CdnDemoRG -EndpointName cdndocdemo -
PurgeContent "/images/kitten.png","/video/rickroll.mp4"

# Pre-load some assets.
Publish-AzCdnEndpointContent -ProfileName CdnDemo -ResourceGroupName CdnDemoRG -EndpointName cdndocdemo -
LoadContent "/images/kitten.png","/video/rickroll.mp4"

# Purge everything in /images/ on all endpoints.
Get-AzCdnProfile | Get-AzCdnEndpoint | Unpublish-AzCdnEndpointContent -PurgeContent "/images/*"
```

## Starting/Stopping CDN endpoints

`Start-AzCdnEndpoint` and `Stop-AzCdnEndpoint` can be used to start and stop individual endpoints or groups of endpoints.

```
# Stop the cdndocdemo endpoint  
Stop-AzCdnEndpoint -ProfileName CdnDemo -ResourceGroupName CdnDemoRG -EndpointName cdndocdemo  
  
# Stop all endpoints  
Get-AzCdnProfile | Get-AzCdnEndpoint | Stop-AzCdnEndpoint  
  
# Start all endpoints  
Get-AzCdnProfile | Get-AzCdnEndpoint | Start-AzCdnEndpoint
```

## Creating Standard Rules engine policy and applying to an existing CDN endpoint

`New-AzCdnDeliveryRule`, `New-AzCdnDeliveryRuleCondition`, and `New-AzCdnDeliveryRuleAction` can be used to configure the Azure CDN Standard Rules engine on Azure CDN from Microsoft profiles.

```
# Create a new http to https redirect rule  
$Condition=New-AzCdnDeliveryRuleCondition -MatchVariable RequestProtocol -Operator Equal -MatchValue HTTP  
$Action=New-AzCdnDeliveryRuleAction -RedirectType Found -DestinationProtocol HTTPS  
$HttpToHttpsRedirectRule=New-AzCdnDeliveryRule -Name "HttpToHttpsRedirectRule" -Order 2 -Condition $Condition  
-Action $Action  
  
# Create a path based Response header modification rule.  
$Cond1=New-AzCdnDeliveryRuleCondition -MatchVariable UrlPath -Operator BeginsWith -MatchValue "/images/"  
$Action1=New-AzCdnDeliveryRuleAction -HeaderActionType ModifyResponseHeader -Action Overwrite -HeaderName  
"Access-Control-Allow-Origin" -Value "*"  
$PathBasedCacheOverrideRule=New-AzCdnDeliveryRule -Name "PathBasedCacheOverride" -Order 1 -Condition $Cond1 -  
Action $action1  
  
# Create a delivery policy with above deliveryRules.  
$Policy = New-AzCdnDeliveryPolicy -Description "DeliveryPolicy" -Rule $HttpToHttpsRedirectRule,$UrlRewriteRule  
  
# Update existing endpoint with created delivery policy  
$ep = Get-AzCdnEndpoint -EndpointName cdndocdemo -ProfileName CdnDemo -ResourceGroupName CdnDemoRG  
$ep.DeliveryPolicy = $Policy  
Set-AzCdnEndpoint -CdnEndpoint $ep
```

## Deleting CDN resources

`Remove-AzCdnProfile` and `Remove-AzCdnEndpoint` can be used to remove profiles and endpoints.

```
# Remove a single endpoint  
Remove-AzCdnEndpoint -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG -EndpointName cdnposhdoc  
  
# Remove all the endpoints on a profile and skip confirmation (-Force)  
Get-AzCdnProfile -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG | Get-AzCdnEndpoint | Remove-  
AzCdnEndpoint -Force  
  
# Remove a single profile  
Remove-AzCdnProfile -ProfileName CdnPoshDemo -ResourceGroupName CdnDemoRG
```

## Next Steps

Learn how to automate Azure CDN with [.NET](#) or [Node.js](#).

To learn about CDN features, see [CDN Overview](#).

# Restrict Azure CDN content by country/region

7/5/2019 • 3 minutes to read • [Edit Online](#)

## Overview

When a user requests your content, by default, the content is served regardless of the location of the user making the request. However, in some cases, you may want to restrict access to your content by country/region. With the *geo-filtering* feature, you can create rules on specific paths on your CDN endpoint to allow or block content in selected countries/regions.

### IMPORTANT

**Azure CDN Standard from Microsoft** profiles do not support path-based geo-filtering.

## Standard profiles

The procedures in this section are for **Azure CDN Standard from Akamai** and **Azure CDN Standard from Verizon** profiles only.

For **Azure CDN Premium from Verizon** profiles, you must use the **Manage** portal to activate geo-filtering. For more information, see [Azure CDN Premium from Verizon profiles](#).

### Define the directory path

To access the geo-filtering feature, select your CDN endpoint within the portal, then select **Geo-filtering** under SETTINGS in the left-hand menu.

The screenshot shows the Azure portal interface for managing a CDN endpoint named 'cdndocdemo'. The left sidebar lists various settings: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Origin, Custom domains, Compression, Caching rules, and Geo-filtering. The 'Geo-filtering' option is highlighted with a red box. The main content area is titled 'cdndocdemo - Geo-filtering Endpoint'. It contains sections for 'About This Feature' (describing geo-filtering rules) and 'Configure' (a table for defining rules). The 'Configure' table has columns for PATH, ACTION, and COUNTRY CODES. A single row is shown with the PATH set to '/pictures/' or '/pictures/city.png'.

From the **PATH** box, specify the relative path to the location to which users will be allowed or denied access.

You can apply geo-filtering for all your files with a forward slash (/) or select specific folders by specifying directory paths (for example, /pictures/). You can also apply geo-filtering to a single file (for example /pictures/city.png). Multiple rules are allowed; after you enter a rule, a blank row appears for you to enter the next rule.

For example, all of the following directory path filters are valid:

/  
/Photos/  
/Photos/Strasbourg/  
/Photos/Strasbourg/city.png

### Define the type of action

From the **ACTION** list, select **Allow** or **Block**:

- **Allow:** Only users from the specified countries/regions are allowed access to assets requested from the recursive path.
- **Block:** Users from the specified countries/regions are denied access to the assets requested from the recursive path. If no other country/region filtering options have been configured for that location, then all other users will be allowed access.

For example, a geo-filtering rule for blocking the path /Photos/Strasbourg/ filters the following files:

<http://<endpoint>.azureedge.net/Photos/Strasbourg/1000.jpg>  
<http://<endpoint>.azureedge.net/Photos/Strasbourg/Cathedral/1000.jpg>

### Define the countries/regions

From the **COUNTRY CODES** list, select the countries/regions that you want to block or allow for the path.

After you have finished selecting the countries/regions, select **Save** to activate the new geo-filtering rule.

Save Discard

**About This Feature**  
Create geo-filtering rules on specific paths on your endpoint to block or allow content in the selected countries. The path can match a single file (e.g. '/pictures/city.png') or a folder (e.g. '/pictures/' or '/pictures/cities/'). When a folder is specified (i.e. with a trailing slash), the filter applies to all files and sub-folders under it.  
[Learn more](#)

**Configure**

	PATH	ACTION	COUNTRY CODES
<input type="checkbox"/>	/myfolder/	Allow	US
<input type="checkbox"/>	/myfolder/file.exe	Allow	US,CA

/pictures/ or /pictures/city.png ▼ 0 selected ▼

### Clean up resources

To delete a rule, select it from the list on the **Geo-filtering** page, then choose **Delete**.

## Azure CDN Premium from Verizon profiles

For **For Azure CDN Premium from Verizon** profiles, the user interface for creating a geo-filtering rule is different:

1. From the top menu in your Azure CDN profile, select **Manage**.
2. From the Verizon portal, select **HTTP Large**, then select **Country Filtering**.

The screenshot shows the Microsoft Azure portal interface. At the top, there are tabs for 'HTTP Large', 'ADN', and 'Analytics'. Below these, a navigation bar includes 'Country Filtering', 'Token Auth', 'Cache Settings >', and 'Rules Engine'. A prominent blue button labeled 'Add Country Filter' is located on the right side of the page. The main content area displays the 'Country Filtering' configuration, which currently shows 'No Country Filters.'.

3. Select **Add Country Filter**.

The **Step One:** page appears.

4. Enter the directory path, select **Block** or **Add**, then select **Next**.

The **Step Two:** page appears.

5. Select one or more countries/regions from the list, then select **Finish** to activate the rule.

The new rule appears in the table on the **Country Filtering** page.

The screenshot shows the 'Country Filtering' page in the Microsoft Azure portal. The table lists the following rules:

	Directory Name	Country Codes	Filter Type
	/myfolder/	US	Allow
	/myfolder/file.exe	CA, US	Allow

A blue 'Add Country Filter' button is located at the bottom right of the table area.

#### Clean up resources

In the country/region filtering rules table, select the delete icon next to a rule to delete it or the edit icon to modify it.

## Considerations

- Changes to your geo-filtering configuration do not take effect immediately:
  - For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
  - For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
  - For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes in 10 minutes.
- This feature does not support wildcard characters (for example, \*).
- The geo-filtering configuration associated with the relative path is applied recursively to that path.
- Only one rule can be applied to the same relative path. That is, you cannot create multiple country/region filters that point to the same relative path. However, because country/region filters are recursive, a folder can have multiple country/region filters. In other words, a subfolder of a previously configured folder can be assigned a different country/region filter.
- The geo-filtering feature uses country codes to define the countries/regions from which a request is allowed or blocked for a secured directory. Although Akamai and Verizon profiles support most of the same country codes, there are a few differences. For more information, see [Azure CDN country codes](#).

# Improve performance by compressing files in Azure CDN

7/5/2019 • 4 minutes to read • [Edit Online](#)

File compression is a simple and effective method to improve file transfer speed and increase page-load performance by reducing a file's size before it is sent from the server. File compression can reduce bandwidth costs and provide a more responsive experience for your users.

There are two ways to enable file compression:

- Enable compression on your origin server. In this case, Azure CDN passes along the compressed files and delivers them to clients that request them.
- Enable compression directly on the CDN POP servers (*compression on the fly*). In this case, the CDN compresses the files and serves them to the end users, even if they were not compressed by the origin server.

## IMPORTANT

Azure CDN configuration changes can take some time to propagate through the network:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes in 10 minutes.

If you're setting up compression for the first time for your CDN endpoint, consider waiting 1-2 hours before you troubleshoot to ensure the compression settings have propagated to the POPs.

## Enabling compression

The standard and premium CDN tiers provide the same compression functionality, but the user interface differs. For more information about the differences between standard and premium CDN tiers, see [Azure CDN Overview](#).

### Standard CDN profiles

#### NOTE

This section applies to **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, and **Azure CDN Standard from Akamai** profiles.

1. From the CDN profile page, select the CDN endpoint you want to manage.

The screenshot shows the Azure CDN endpoint configuration page for the 'CdnDemo' profile. It displays the following details:

- Resource group:** CdnDemoRG
- Status:** Active
- Location:** West US
- Subscription name:** AZURE-CDN-TEST
- Subscription ID:** <subscription ID>

**Endpoints:** 2

HOSTNAME	STATUS	PROTOCOL	ORIGIN TYPE	CUSTOM DOMAINS
cdndocdemo.azureedge.net	Running	HTTP, HTTPS	Storage	seattle.azurecdn.us
cdndocdemo3.azureedge.net	Running	HTTP, HTTPS	Storage	

The CDN endpoint page opens.

## 2. Select **Compression**.

The screenshot shows the 'cdndocdemo' endpoint overview page. The left sidebar menu includes the following items:

- Overview (highlighted)
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

**SETTINGS**

- Origin
- Custom domains
- Compression (highlighted with a red box)
- Caching rules
- Geo-filtering

The compression page opens.

## 3. Select **On** to turn on compression.

Save Discard

### About This Feature

Compress files on the fly via CDN to reduce size and improve performance. After the feature is on, you may modify the MIME types list to fine tune which content formats to compress. Already compressed files such as jpeg images or mp3 audio may not see much additional gain from compression. Note that files are only compressed on the fly by the CDN if it is served from CDN cache. Compressed by the origin can still be delivered compressed to the client without being cached.

[Learn more](#)

### Configure

Compression

Formats to compress

text/plain	...
text/html	...
text/css	...
text/javascript	...
application/x-javascript	...
application/javascript	...
application/json	...
application/xml	...
	...

4. Use the default MIME types, or modify the list by adding or removing MIME types.

#### TIP

Although it is possible, it is not recommended to apply compression to compressed formats. For example, ZIP, MP3, MP4, or JPG.

#### NOTE

Modifying the default list of MIME types is currently not supported in Azure CDN Standard from Microsoft.

5. After making your changes, select **Save**.

### Premium CDN profiles

#### NOTE

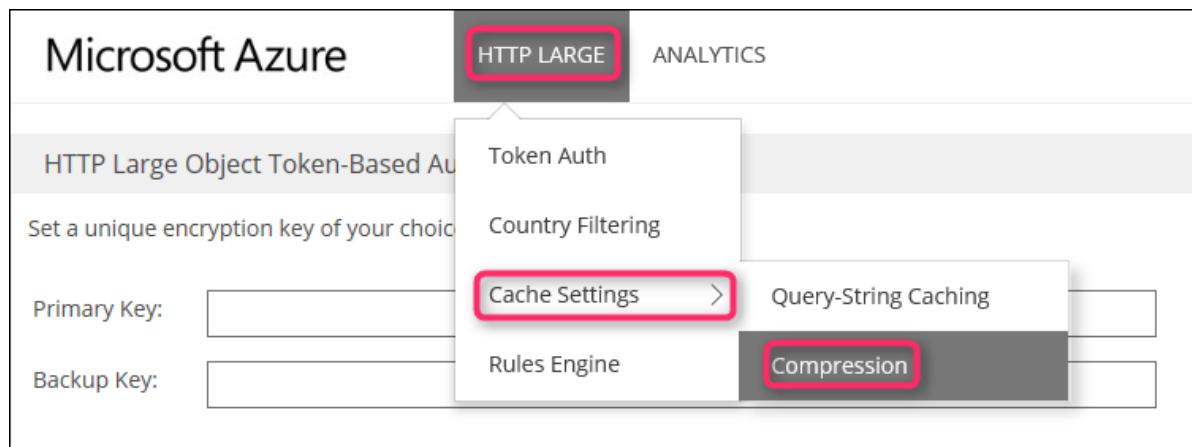
This section applies only to **Azure CDN Premium from Verizon** profiles.

1. From the CDN profile page, select **Manage**.



The CDN management portal opens.

2. Hover over the **HTTP Large** tab, then hover over the **Cache Settings** flyout. Select **Compression**.



The compression options are displayed.

A screenshot of the 'Compression' settings page. It shows a radio button for 'Compression Enabled' (selected) and another for 'Compression Disabled'. Below that is a 'File Types:' section containing a list of MIME types separated by commas: 'text/plain, text/html, text/css, application/x-javascript, ext/javascript'. At the bottom, it shows 'Last Updated: 4/20/2016 8:45:59 PM' and 'Expected Complete Time (1 Hour): 4/20/2016 9:45:59 PM'.

3. Enable compression by selecting **Compression Enabled**. Enter the MIME types you want to compress as a comma-delimited list (no spaces) in the **File Types** box.

**TIP**

Although it is possible, it is not recommended to apply compression to compressed formats. For example, ZIP, MP3, MP4, or JPG.

4. After making your changes, select **Update**.

## Compression rules

### Azure CDN Standard from Microsoft profiles

For **Azure CDN Standard from Microsoft** profiles, only eligible files are compressed. To be eligible for compression, a file must:

- Be of a MIME type that has been [configured for compression](#).
- Be larger than 1 KB
- Be smaller than 8 MB

These profiles support the following compression encodings:

- gzip (GNU zip)
- brotli

If the request supports more than one compression type, brotli compression takes precedence.

When a request for an asset specifies gzip compression and the request results in a cache miss, Azure CDN performs gzip compression of the asset directly on the POP server. Afterward, the compressed file is served from the cache.

### Azure CDN from Verizon profiles

For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, only eligible files are compressed. To be eligible for compression, a file must:

- Be larger than 128 bytes
- Be smaller than 3 MB

These profiles support the following compression encodings:

- gzip (GNU zip)
- DEFLATE
- bzip2
- brotli

If the request supports more than one compression type, those compression types take precedence over brotli compression.

When a request for an asset specifies brotli compression (HTTP header is `Accept-Encoding: br`) and the request results in a cache miss, Azure CDN performs brotli compression of the asset directly on the POP server. Afterward, the compressed file is served from the cache.

### Azure CDN Standard from Akamai profiles

For **Azure CDN Standard from Akamai** profiles, all files are eligible for compression. However, a file must be of a MIME type that has been [configured for compression](#).

These profiles support gzip compression encoding only. When a profile endpoint requests a gzip-encoded file, it is always requested from the origin, regardless of the client request.

## Compression behavior tables

The following tables describe Azure CDN compression behavior for every scenario:

### Compression is disabled or file is ineligible for compression

CLIENT-REQUESTED FORMAT (VIA ACCEPT-ENCODING HEADER)	CACHED-FILE FORMAT	THE CDN RESPONSE TO THE CLIENT	NOTES
Compressed	Compressed	Compressed	
Compressed	Uncompressed	Uncompressed	
Compressed	Not cached	Compressed or Uncompressed	The origin response determines whether CDN performs a compression.
Uncompressed	Compressed	Uncompressed	
Uncompressed	Uncompressed	Uncompressed	
Uncompressed	Not cached	Uncompressed	

## Compression is enabled and file is eligible for compression

CLIENT-REQUESTED FORMAT (VIA ACCEPT-ENCODING HEADER)	CACHED-FILE FORMAT	CDN RESPONSE TO THE CLIENT	NOTES
Compressed	Compressed	Compressed	CDN transcodes between supported formats.
Compressed	Uncompressed	Compressed	CDN performs a compression.
Compressed	Not cached	Compressed	CDN performs a compression if the origin returns an uncompressed file. <b>Azure CDN from Verizon</b> passes the uncompressed file on the first request and then compresses and caches the file for subsequent requests. Files with the <code>Cache-Control: no-cache</code> header are never compressed.
Uncompressed	Compressed	Uncompressed	CDN performs a decompression.
Uncompressed	Uncompressed	Uncompressed	
Uncompressed	Not cached	Uncompressed	

## Media Services CDN Compression

For endpoints enabled for Media Services CDN streaming, compression is enabled by default for the following MIME types:

- application/vnd.ms-sstr+xml
- application/dash+xml
- application/vnd.apple.mpegurl
- application/f4m+xml

## See also

- [Troubleshooting CDN file compression](#)

# Control Azure CDN caching behavior with caching rules

11/20/2019 • 3 minutes to read • [Edit Online](#)

## NOTE

Caching rules are available only for **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles. For **Azure CDN from Microsoft** profiles, you must use the [Standard rules engine](#). For **Azure CDN Premium from Verizon** profiles, you must use the [Verizon Premium rules engine](#) in the **Manage** portal for similar functionality.

Azure Content Delivery Network (CDN) offers two ways to control how your files are cached:

- Caching rules: This article describes how you can use content delivery network (CDN) caching rules to set or modify default cache expiration behavior both globally and with custom conditions, such as a URL path and file extension. Azure CDN provides two types of caching rules:
  - Global caching rules: You can set one global caching rule for each endpoint in your profile, which affects all requests to the endpoint. The global caching rule overrides any HTTP cache-directive headers, if set.
  - Custom caching rules: You can set one or more custom caching rules for each endpoint in your profile. Custom caching rules match specific paths and file extensions, are processed in order, and override the global caching rule, if set.
- Query string caching: You can adjust how the Azure CDN treats caching for requests with query strings. For information, see [Control Azure CDN caching behavior with query strings](#). If the file is not cacheable, the query string caching setting has no effect, based on caching rules and CDN default behaviors.

For information about default caching behavior and caching directive headers, see [How caching works](#).

## Accessing Azure CDN caching rules

1. Open the Azure portal, select a CDN profile, then select an endpoint.
2. In the left pane under Settings, select **Caching rules**.

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu includes 'Create a resource', 'All services', and various service categories like Dashboard, App Services, Function Apps, etc. The main content area is titled 'cdndocdemo1 Endpoint' under 'CdnDocDemo > cdndocdemo1'. The 'Essentials' tab is selected, displaying details such as Resource group (CdnDemoRG), Endpoint hostname (https://cdndocdemo1.azureedge.net), Status (Running), Location (West US), Subscription name (<subscription name>), and Protocols (HTTP, HTTPS). Below this, the 'Custom domains' section shows a table with one row: 'HOSTNAME' (empty), 'CUSTOM HTTPS' (empty), and 'DETAILS' (empty). A red box highlights the 'Caching rules' link in the sidebar.

The **Caching rules** page appears.

This screenshot shows the 'cdndocdemo1 - Caching rules' page. It includes sections for 'About This Feature', 'Global caching rules', and 'Custom caching rules'. Under 'Global caching rules', there are fields for 'Caching behavior' (set to 'Not set'), 'Cache expiration duration' (0 days, 0 hours, 0 minutes, 0 seconds), and 'Query string caching behavior' (set to 'Ignore query strings'). The 'Custom caching rules' section contains a table with columns for 'MATCH CONDITION', 'MATCH VALUE(S)', 'CACHING BEHAVIOR', 'DAYS', 'HOURS', 'MINUTES', and 'SECONDS'. A single row is present in the table, with all values set to 0. Navigation buttons at the top include Save, Discard, Move up, Move down, Move to top, Move to bottom, Insert, and Delete.

## Caching behavior settings

For global and custom caching rules, you can specify the following **Caching behavior** settings:

- **Bypass cache:** Do not cache and ignore origin-provided cache-directive headers.
- **Override:** Ignore origin-provided cache duration; use the provided cache duration instead. This will not override cache-control: no-cache.
- **Set if missing:** Honor origin-provided cache-directive headers, if they exist; otherwise, use the provided cache duration.

**Global caching rules**

These rules affect the CDN caching behavior for all requests, and can be overridden using Custom Cache Rules below for certain scenarios. Note that the Query string caching behavior setting does not affect files that are not cached by the CDN.

Caching behavior <small>?</small>	Set if missing				<small>▼</small>	
Cache expiration duration <small>?</small>	Days 10	Hours 0	Minutes 0	Seconds 0		
Query string caching behavior <small>?</small>	Ignore query strings				<small>▼</small>	

**Custom caching rules**

Create caching rules based on specific match conditions. These rules override the default settings above, and are evaluated from top to down. This means that rules lower on the list can override rules above it in the list, as well as the global caching rules and default behavior. Therefore it makes more sense to have more specific rules towards the bottom of the list so they are not overwritten by a general rule under them. For example a rule for path '/folder/images/\*' should be below a rule for path '/folder/\*'.

<span style="margin-right: 10px;">↑ Move up</span> <span style="margin-right: 10px;">↓ Move down</span> <span style="margin-right: 10px;">⤒ Move to top</span> <span style="margin-right: 10px;">⤓ Move to bottom</span> <span style="margin-right: 10px;">☰ Insert</span> <span style="margin-right: 10px;">trash Delete</span>						
<input type="checkbox"/> MATCH CONDITION	MATCH VALUE(S)	CACHING BEHAVIOR	DAYS	HOURS	MINUTES	SECONDS
Path	/images/*.jpg	Override	30	0	0	0
			0	0	0	0

## Cache expiration duration

For global and custom caching rules, you can specify the cache expiration duration in days, hours, minutes, and seconds:

- For the **Override** and **Set if missing Caching behavior** settings, valid cache durations range between 0 seconds and 366 days. For a value of 0 seconds, the CDN caches the content, but must revalidate each request with the origin server.
- For the **Bypass cache** setting, the cache duration is automatically set to 0 seconds and cannot be changed.

## Custom caching rules match conditions

For custom cache rules, two match conditions are available:

- Path:** This condition matches the path of the URL, excluding the domain name, and supports the wildcard symbol (\*). For example, /myfile.html, /my/folder/\*, and /my/images/\*.jpg. The maximum length is 260 characters.
- Extension:** This condition matches the file extension of the requested file. You can provide a list of comma-separated file extensions to match. For example, jpg, .mp3, or .png. The maximum number of extensions is 50 and the maximum number of characters per extension is 16.

## Global and custom rule processing order

Global and custom caching rules are processed in the following order:

- Global caching rules take precedence over the default CDN caching behavior (HTTP cache-directive header settings).
- Custom caching rules take precedence over global caching rules, where they apply. Custom caching rules are processed in order from top to bottom. That is, if a request matches both conditions, rules at the

bottom of the list take precedence over rules at the top of the list. Therefore, you should place more specific rules lower in the list.

**Example:**

- Global caching rule:
  - Caching behavior: **Override**
  - Cache expiration duration: 1 day
- Custom caching rule #1:
  - Match condition: **Path**
  - Match value: */home/\**
  - Caching behavior: **Override**
  - Cache expiration duration: 2 days
- Custom caching rule #2:
  - Match condition: **Extension**
  - Match value: *.html*
  - Caching behavior: **Set if missing**
  - Cache expiration duration: 3 days

When these rules are set, a request for *<endpoint hostname>.azureedge.net/home/index.html* triggers custom caching rule #2, which is set to: **Set if missing** and 3 days. Therefore, if the *index.html* file has `Cache-Control` or `Expires` HTTP headers, they are honored; otherwise, if these headers are not set, the file is cached for 3 days.

**NOTE**

Files that are cached before a rule change maintain their origin cache duration setting. To reset their cache durations, you must [purge the file](#).

Azure CDN configuration changes can take some time to propagate through the network:

- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** profiles, propagation usually completes in 10 minutes.

## See also

- [How caching works](#)
- [Tutorial: Set Azure CDN caching rules](#)

# Control Azure CDN caching behavior with query strings - standard tier

11/14/2019 • 2 minutes to read • [Edit Online](#)

## Overview

With Azure Content Delivery Network (CDN), you can control how files are cached for a web request that contains a query string. In a web request with a query string, the query string is that portion of the request that occurs after a question mark (?). A query string can contain one or more key-value pairs, in which the field name and its value are separated by an equals sign (=). Each key-value pair is separated by an ampersand (&). For example, `http://www.contoso.com/content.mov?field1=value1&field2=value2`. If there is more than one key-value pair in a query string of a request, their order does not matter.

### IMPORTANT

The Azure CDN standard and premium products provide the same query string caching functionality, but the user interface is different. This article describes the interface for **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Akamai** and **Azure CDN Standard from Verizon**. For query string caching with **Azure CDN Premium from Verizon**, see [Control Azure CDN caching behavior with query strings - premium tier](#).

Three query string modes are available:

- **Ignore query strings:** Default mode. In this mode, the CDN point-of-presence (POP) node passes the query strings from the requestor to the origin server on the first request and caches the asset. All subsequent requests for the asset that are served from the POP ignore the query strings until the cached asset expires.
- **Bypass caching for query strings:** In this mode, requests with query strings are not cached at the CDN POP node. The POP node retrieves the asset directly from the origin server and passes it to the requestor with each request.
- **Cache every unique URL:** In this mode, each request with a unique URL, including the query string, is treated as a unique asset with its own cache. For example, the response from the origin server for a request for `example.ashx?q=test1` is cached at the POP node and returned for subsequent caches with the same query string. A request for `example.ashx?q=test2` is cached as a separate asset with its own time-to-live setting.

### IMPORTANT

Do not use this mode when the query string contains parameters that will change with every request, such as a session ID or a user name, because it will result in a low cache-hit ratio.

## Changing query string caching settings for standard CDN profiles

1. Open a CDN profile, then select the CDN endpoint you want to manage.

## Endpoints



HOSTNAME	STATUS	PROTOCOL	ORIGIN TYPE	CUSTOM DOMAINS
cdndocdemo.azureedge.net	Running	HTTP, HTTPS	Storage	seattle.azurecdn.us
cdndocdemo3.azureedge.net	Running	HTTP, HTTPS	Storage	

2. In the left pane under Settings, click **Caching rules**.

The screenshot shows the Azure portal interface. At the top, there's a breadcrumb navigation: Home > CdnDemo > cdndocdemo. Below this is a section titled "cdndocdemo Endpoint". A search bar labeled "Search (Ctrl+ /)" is present. On the left, a vertical navigation menu lists several options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS. Under SETTINGS, the options are Origin, Custom domains, Compression, Caching rules, and Geo-filtering. The "Caching rules" option is highlighted with a red box.

3. In the **Query string caching behavior** list, select a query string mode, then click **Save**.

The screenshot shows the "Caching rules" configuration page. At the top, there are three buttons: Save (highlighted with a red box), Discard, and Export. Below these are fields for "Default cache expiration duration" (set to 7 days) and "Cache expiration duration" (set to 0 days, 0 hours, 0 minutes, 0 seconds). A dropdown menu for "Query string caching behavior" is open, showing four options: "Ignore query strings" (selected and highlighted with a blue background), "Bypass caching for query strings", and "Cache every unique URL".

Default cache expiration duration ⓘ 7 days

Cache expiration duration ⓘ Days Hours Minutes Seconds  
0 0 0 0

Query string caching behavior ⓘ

- Ignore query strings
- Bypass caching for query strings
- Cache every unique URL

Custom caching rules

Create caching rules based on spec top to down. This means that rules lower on the list can override rules above it in the list, as well as the global caching rules and

#### **IMPORTANT**

Because it takes time for the registration to propagate through Azure CDN, cache string settings changes might not be immediately visible:

- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in 10 minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes in 10 minutes.

# Control Azure CDN caching behavior with query strings - premium tier

11/14/2019 • 2 minutes to read • [Edit Online](#)

## Overview

With Azure Content Delivery Network (CDN), you can control how files are cached for a web request that contains a query string. In a web request with a query string, the query string is that portion of the request that occurs after a question mark (?). A query string can contain one or more key-value pairs, in which the field name and its value are separated by an equals sign (=). Each key-value pair is separated by an ampersand (&). For example, `http://www.contoso.com/content.mov?field1=value1&field2=value2`. If there is more than one key-value pair in a query string of a request, their order does not matter.

### IMPORTANT

The standard and premium CDN products provide the same query string caching functionality, but the user interface is different. This article describes the interface for **Azure CDN Premium from Verizon**. For query string caching with Azure CDN standard products, see [Control Azure CDN caching behavior with query strings - standard tier](#).

Three query string modes are available:

- **standard-cache:** Default mode. In this mode, the CDN point-of-presence (POP) node passes the query strings from the requestor to the origin server on the first request and caches the asset. All subsequent requests for the asset that are served from the POP server ignore the query strings until the cached asset expires.

### IMPORTANT

If token authorization is enabled for any path on this account, standard-cache mode is the only mode that can be used.

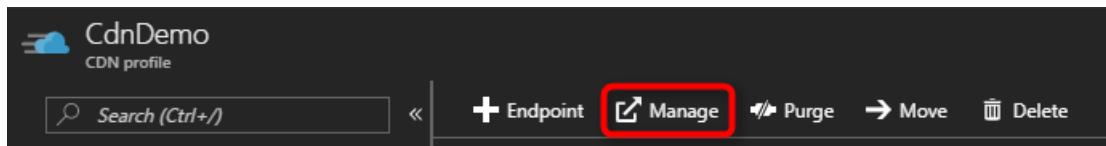
- **no-cache:** In this mode, requests with query strings are not cached at the CDN POP node. The POP node retrieves the asset directly from the origin server and passes it to the requestor with each request.
- **unique-cache:** In this mode, each request with a unique URL, including the query string, is treated as a unique asset with its own cache. For example, the response from the origin server for a request for `example.ashx?q=test1` is cached at the POP node and returned for subsequent caches with the same query string. A request for `example.ashx?q=test2` is cached as a separate asset with its own time-to-live setting.

### IMPORTANT

Do not use this mode when the query string contains parameters that will change with every request, such as a session ID or a user name, because it will result in a low cache-hit ratio.

## Changing query string caching settings for premium CDN profiles

1. Open a CDN profile, then click **Manage**.



The CDN management portal opens.

2. Hover over the **HTTP Large** tab, then hover over the **Cache Settings** flyout menu. Click **Query-String Caching**.

Query string caching options are displayed.

- standard-cache
- no-cache
- unique-cache

3. Select a query string mode, then click **Update**.

**IMPORTANT**

Because it takes time for the registration to propagate through the CDN, cache string settings changes might not be immediately visible. Propagation usually completes in 10 minutes.

# Purge an Azure CDN endpoint

11/26/2019 • 2 minutes to read • [Edit Online](#)

## Overview

Azure CDN edge nodes will cache assets until the asset's time-to-live (TTL) expires. After the asset's TTL expires, when a client requests the asset from the edge node, the edge node will retrieve a new updated copy of the asset to serve the client request and store refresh the cache.

The best practice to make sure your users always obtain the latest copy of your assets is to version your assets for each update and publish them as new URLs. CDN will immediately retrieve the new assets for the next client requests. Sometimes you may wish to purge cached content from all edge nodes and force them all to retrieve new updated assets. This might be due to updates to your web application, or to quickly update assets that contain incorrect information.

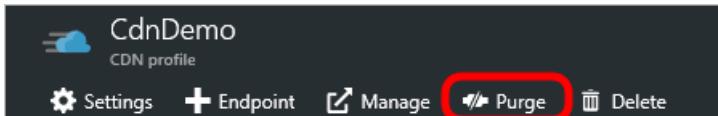
### TIP

Note that purging only clears the cached content on the CDN edge servers. Any downstream caches, such as proxy servers and local browser caches, may still hold a cached copy of the file. It's important to remember this when you set a file's time-to-live. You can force a downstream client to request the latest version of your file by giving it a unique name every time you update it, or by taking advantage of [query string caching](#).

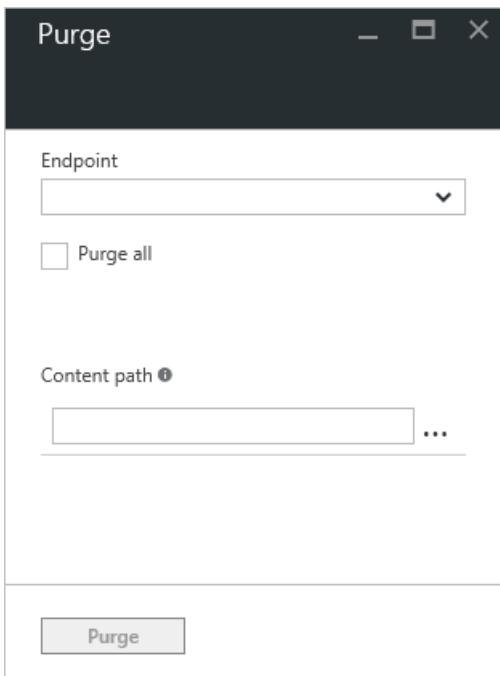
This tutorial walks you through purging assets from all edge nodes of an endpoint.

## Walkthrough

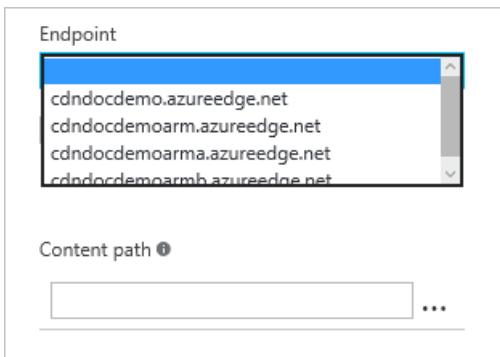
1. In the [Azure Portal](#), browse to the CDN profile containing the endpoint you wish to purge.
2. From the CDN profile blade, click the purge button.



The Purge blade opens.



3. On the Purge blade, select the service address you wish to purge from the URL dropdown.



#### NOTE

You can also get to the Purge blade by clicking the **Purge** button on the CDN endpoint blade. In that case, the **URL** field will be pre-populated with the service address of that specific endpoint.

4. Select what assets you wish to purge from the edge nodes. If you wish to clear all assets, click the **Purge all** checkbox. Otherwise, type the path of each asset you wish to purge in the **Path** textbox. Below formats are supported in the path.

- a. **Single URL purge:** Purge individual asset by specifying the full URL, with or without the file extension, e.g., `/pictures/strasbourg.png` ; `/pictures/strasbourg`
- b. **Wildcard purge:** Asterisk (\*) may be used as a wildcard. Purge all folders, sub-folders and files under an endpoint with `/*` in the path or purge all sub-folders and files under a specific folder by specifying the folder followed by `/*`, e.g., `/pictures/*`. Note that wildcard purge is not supported by Azure CDN from Akamai currently.
- c. **Root domain purge:** Purge the root of the endpoint with "/" in the path.

#### TIP

Paths must be specified for purge and must be a relative URL that fit the following regular expression. [Purge all](#) and [Wildcard purge](#) not supported by [Azure CDN from Akamai](#) currently.

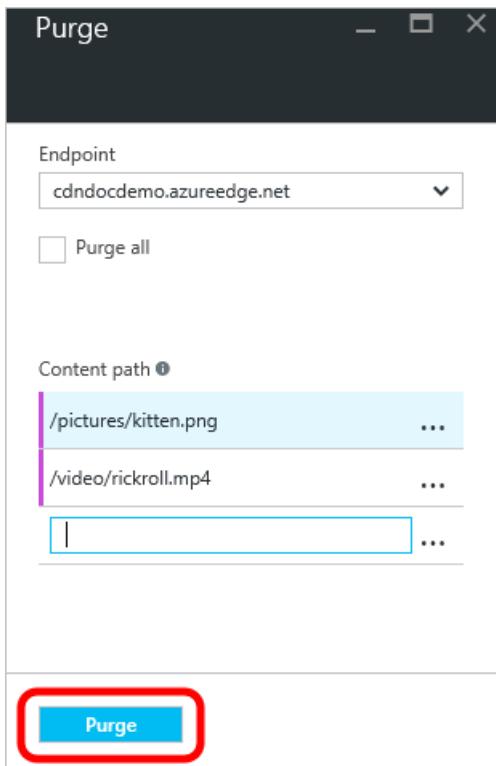
Single URL purge `@"^\/(?:[a-zA-Z0-9-_.=\(\)\u0020]+\\/?)*$";`

Query string `@"^(?:[-_a-zA-Z0-9\%:_!=,\.\+\&\(\)\u0020]*?)?$$";`

Wildcard purge `@"^\/(?:[a-zA-Z0-9-_.=\(\)\u0020]+\\/)*\*$"; .`

More **Path** textboxes will appear after you enter text to allow you to build a list of multiple assets. You can delete assets from the list by clicking the ellipsis (...) button.

5. Click the **Purge** button.



#### IMPORTANT

Purge requests take approximately 10 minutes to process with [Azure CDN from Microsoft](#), approximately 2 minutes with [Azure CDN from Verizon](#) (standard and premium), and approximately 10 seconds with [Azure CDN from Akamai](#). Azure CDN has a limit of 50 concurrent purge requests at any given time at the profile level.

## See also

- [Pre-load assets on an Azure CDN endpoint](#)
- [Azure CDN REST API reference - Purge or Pre-Load an Endpoint](#)

# Pre-load assets on an Azure CDN endpoint

7/5/2019 • 2 minutes to read • [Edit Online](#)

## IMPORTANT

This feature is available only with **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** products. It is not supported on **Azure CDN from Akamai**. For a comparison of CDN features, see [Azure CDN product features](#).

By default, assets are cached only when they're requested. Because the edge servers have not yet cached the content and need to forward the request to the origin server, the first request from each region can take longer than subsequent requests. To avoid this first-hit latency, pre-load your assets. In addition to providing a better customer experience, pre-loading your cached assets can reduce network traffic on the origin server.

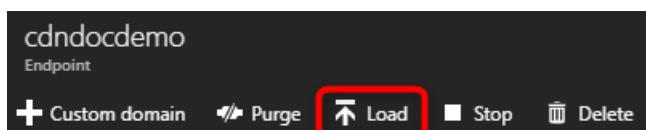
## NOTE

Pre-loading assets is useful for large events or content that becomes simultaneously available to many users, such as a new movie release or a software update.

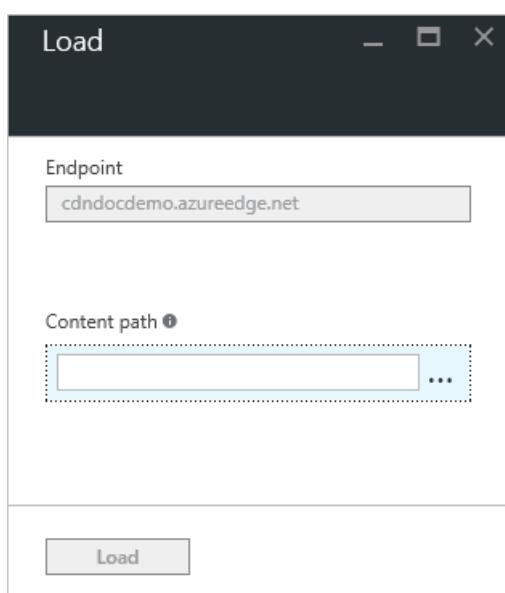
This tutorial walks you through pre-loading cached content on all Azure CDN edge nodes.

## To pre-load assets

1. In the [Azure portal](#), browse to the CDN profile containing the endpoint you wish to pre-load. The profile pane opens.
2. Click the endpoint in the list. The endpoint pane opens.
3. From the CDN endpoint pane, select **Load**.



The **Load** pane opens.



4. For **Content path**, enter the full path of each asset you wish to load (for example, `/pictures/kitten.png`).

**TIP**

After you start entering text, more **Content path** text boxes will appear to allow you to build a list of multiple assets.

To delete assets from the list, select the ellipsis (...) button, then select **Delete**.

Each content path must be a relative URL that fits the following [regular expressions](#):

- Load a single file path: `^(?:\/[a-zA-Z0-9-_.=\u0020]+)+$`
- Load a single file with query string: `^(?:\?[-_a-zA-Z0-9/\%:_!=,.+\^&\u0020]*?)?$_`

Because each asset must have its own path, there's no wildcard functionality for pre-loading assets.

The screenshot shows a list of content paths in a table-like structure. The columns are labeled 'Path' and '...'. The first row contains the path '/pictures/kitten.png' and the second row contains the path '/video/rickroll.mp4'. Each row has an ellipsis button ('...') to its right, which typically indicates a delete or edit function in such interfaces.

Path	...
/pictures/kitten.png	...
/video/rickroll.mp4	...
	...
	...

5. When you are finished entering content paths, select **Load**.

**NOTE**

There's a limit of 10 load requests per minute per CDN profile and 50 concurrent paths can be processed at one time. Each path has a path-length limit of 1024 characters.

## See also

- [Purge an Azure CDN endpoint](#)
- [Azure CDN REST API reference: Pre-load content on an endpoint](#)
- [Azure CDN REST API reference: Purge content from an endpoint](#)

# Manage expiration of web content in Azure CDN

7/5/2019 • 5 minutes to read • [Edit Online](#)

Files from publicly accessible origin web servers can be cached in Azure Content Delivery Network (CDN) until their time-to-live (TTL) elapses. The TTL is determined by the `Cache-Control` header in the HTTP response from the origin server. This article describes how to set `Cache-Control` headers for the Web Apps feature of Microsoft Azure App Service, Azure Cloud Services, ASP.NET applications, and Internet Information Services (IIS) sites, all of which are configured similarly. You can set the `Cache-Control` header either by using configuration files or programmatically.

You can also control cache settings from the Azure portal by setting [CDN caching rules](#). If you create one or more caching rules and set their caching behavior to **Override** or **Bypass cache**, the origin-provided caching settings discussed in this article are ignored. For information about general caching concepts, see [How caching works](#).

## TIP

You can choose to set no TTL on a file. In this case, Azure CDN automatically applies a default TTL of seven days, unless you've set up caching rules in the Azure portal. This default TTL applies only to general web delivery optimizations. For large file optimizations, the default TTL is one day, and for media streaming optimizations, the default TTL is one year.

For more information about how Azure CDN works to speed up access to files and other resources, see [Overview of the Azure Content Delivery Network](#).

## Setting Cache-Control headers by using CDN caching rules

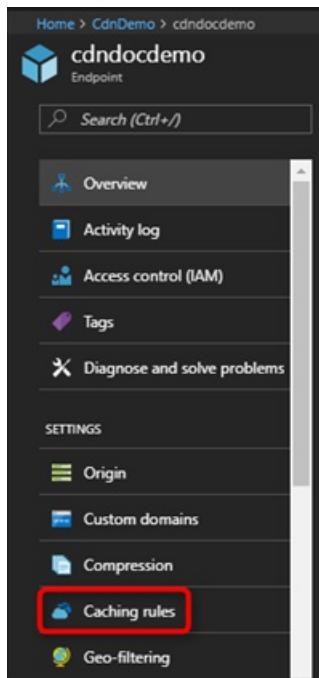
The preferred method for setting a web server's `Cache-Control` header is to use caching rules in the Azure portal. For more information about CDN caching rules, see [Control Azure CDN caching behavior with caching rules](#).

## NOTE

Caching rules are available only for **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles. For **Azure CDN Premium from Verizon** profiles, you must use the [Azure CDN rules engine](#) in the **Manage** portal for similar functionality.

### To navigate to the CDN caching rules page:

1. In the Azure portal, select a CDN profile, then select the endpoint for the web server.
2. In the left pane under Settings, select **Caching rules**.



The **Caching rules** page appears.

MATCH CONDITION	MATCH VALUES(S)	CACHING BEHAVIOR	DAYS	HOURS	MINUTES	SECONDS
Path	/images/*.jpg	Bypass cache	0	0	0	0
			0	0	0	0

#### To set a web server's Cache-Control headers by using global caching rules:

- Under **Global caching rules**, set **Query string caching behavior** to **Ignore query strings** and set **Caching behavior** to **Override**.
- For **Cache expiration duration**, enter 3600 in the **Seconds** box or 1 in the **Hours** box.

MATCH CONDITION	MATCH VALUES(S)	CACHING BEHAVIOR	DAYS	HOURS	MINUTES	SECONDS
Path	/images/*.jpg	Override	0	0	0	3600
			0	0	0	0

This global caching rule sets a cache duration of one hour and affects all requests to the endpoint. It overrides any `Cache-Control` or `Expires` HTTP headers that are sent by the origin server specified by the endpoint.

### 3. Select **Save**.

#### To set a web server file's Cache-Control headers by using custom caching rules:

##### 1. Under **Custom caching rules**, create two match conditions:

- a. For the first match condition, set **Match condition** to **Path** and enter `/webfolder1/*` for **Match value**. Set **Caching behavior** to **Override** and enter 4 in the **Hours** box.
- b. For the second match condition, set **Match condition** to **Path** and enter `/webfolder1/file1.txt` for **Match value**. Set **Caching behavior** to **Override** and enter 2 in the **Hours** box.

Custom caching rules						
Create caching rules based on specific match conditions. These rules override the default settings above, and are evaluated from top to down. This means that rules lower on the list can override rules above it in the list, as well as the global caching rules and default behavior. Therefore it makes more sense to have more specific rules towards the bottom of the list so they are not overwritten by a general rule under them. For example a rule for path '/folder/images/*' should be below a rule for path '/folder/*'.						
Match Condition		Match Value(s)	Caching Behavior	Days	Hours	Minutes
<input type="checkbox"/>	UriPath	<code>/webfolder1/*</code>	Override	4	0	0
<input type="checkbox"/>	UriPath	<code>/webfolder1/file1.txt</code>	Override	2	0	0
				0	0	0

The first custom caching rule sets a cache duration of four hours for any files in the `/webfolder1` folder on the origin server specified by your endpoint. The second rule overrides the first rule for the `file1.txt` file only and sets a cache duration of two hours for it.

### 2. Select **Save**.

## Setting Cache-Control headers by using configuration files

For static content, such as images and style sheets, you can control the update frequency by modifying the **applicationHost.config** or **Web.config** configuration files for your web application. To set the `Cache-Control` header for your content, use the `<system.webServer>/<staticContent>/<clientCache>` element in either file.

### Using ApplicationHost.config files

The **ApplicationHost.config** file is the root file of the IIS configuration system. The configuration settings in an **ApplicationHost.config** file affect all applications on the site, but are overridden by the settings of any **Web.config** files that exist for a web application.

### Using Web.config files

With a **Web.config** file, you can customize the way your entire web application or a specific directory on your web application behaves. Typically, you have at least one **Web.config** file in the root folder of your web application. For each **Web.config** file in a specific folder, the configuration settings affect everything in that folder and its subfolders, unless they are overridden at the subfolder level by another **Web.config** file.

For example, you can set a `<clientCache>` element in a **Web.config** file in the root folder of your web application to cache all static content on your web application for three days. You can also add a **Web.config** file in a subfolder with more variable content (for example, `\frequent`) and set its `<clientCache>` element to cache the subfolder's content for six hours. The net result is that content on the entire web site is cached for three days, except for any content in the `\frequent` directory, which is cached for only six hours.

The following XML configuration file example shows how to set the `<clientCache>` element to specify a maximum

age of three days:

```
<configuration>
  <system.webServer>
    <staticContent>
      <clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="3.00:00:00" />
    </staticContent>
  </system.webServer>
</configuration>
```

To use the **cacheControlMaxAge** attribute, you must set the value of the **cacheControlMode** attribute to `UseMaxAge`. This setting causes the HTTP header and directive, `Cache-Control: max-age=<nnn>`, to be added to the response. The format of the timespan value for the **cacheControlMaxAge** attribute is `<days>.<hours>:<min>:<sec>`. Its value is converted to seconds and is used as the value of the `Cache-Control max-age` directive. For more information about the `<clientCache>` element, see [Client Cache <clientCache>](#).

## Setting Cache-Control headers programmatically

For ASP.NET applications, you control the CDN caching behavior programmatically by setting the **HttpResponse.Cache** property of the .NET API. For information about the **HttpResponse.Cache** property, see [HttpResponse.Cache Property](#) and [HttpCachePolicy Class](#).

To programmatically cache application content in ASP.NET, follow these steps:

1. Verify that the content is marked as cacheable by setting `HttpCacheability` to `Public`.
2. Set a cache validator by calling one of the following `HttpCachePolicy` methods:
  - Call `SetLastModified` to set a timestamp value for the `Last-Modified` header.
  - Call `SetETag` to set a value for the `ETag` header.
3. Optionally, specify a cache expiration time by calling `SetExpires` to set a value for the `Expires` header. Otherwise, the default cache heuristics described previously in this document apply.

For example, to cache content for one hour, add the following C# code:

```
// Set the caching parameters.
Response.Cache.SetExpires(DateTime.Now.AddHours(1));
Response.Cache.SetCacheability(HttpCacheability.Public);
Response.Cache.SetLastModified(DateTime.Now);
```

## Testing the Cache-Control header

You can easily verify the TTL settings of your web content. With your browser's [developer tools](#), test that your web content includes the `Cache-Control` response header. You can also use a tool such as [wget](#), [Postman](#), or [Fiddler](#) to examine the response headers.

## Next Steps

- [Read details about the clientCache element](#)
- [Read the documentation for the HttpResponse.Cache Property](#)
- [Read the documentation for the HttpCachePolicy Class](#)
- [Learn about caching concepts](#)

# Manage expiration of Azure Blob storage in Azure CDN

11/14/2019 • 5 minutes to read • [Edit Online](#)

The [Blob storage service](#) in Azure Storage is one of several Azure-based origins integrated with Azure Content Delivery Network (CDN). Any publicly accessible blob content can be cached in Azure CDN until its time-to-live (TTL) elapses. The TTL is determined by the `Cache-Control` header in the HTTP response from the origin server. This article describes several ways that you can set the `Cache-Control` header on a blob in Azure Storage.

You can also control cache settings from the Azure portal by setting CDN caching rules. If you create a caching rule and set its caching behavior to **Override** or **Bypass cache**, the origin-provided caching settings discussed in this article are ignored. For information about general caching concepts, see [How caching works](#).

## TIP

You can choose to set no TTL on a blob. In this case, Azure CDN automatically applies a default TTL of seven days, unless you have set up caching rules in the Azure portal. This default TTL applies only to general web delivery optimizations. For large file optimizations, the default TTL is one day, and for media streaming optimizations, the default TTL is one year.

For more information about how Azure CDN works to speed up access to blobs and other files, see [Overview of the Azure Content Delivery Network](#).

For more information about Azure Blob storage, see [Introduction to Blob storage](#).

## Setting Cache-Control headers by using CDN caching rules

The preferred method for setting a blob's `Cache-Control` header is to use caching rules in the Azure portal. For more information about CDN caching rules, see [Control Azure CDN caching behavior with caching rules](#).

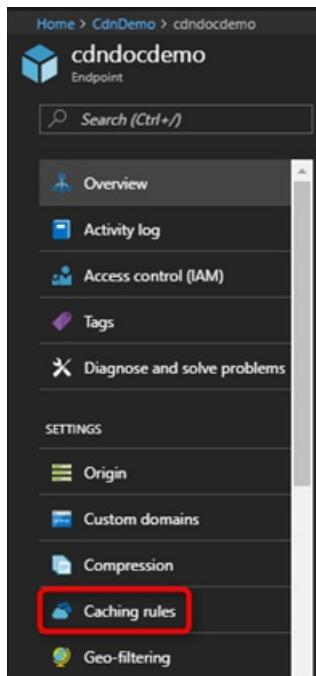
## NOTE

Caching rules are available only for **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles.

For **Azure CDN Premium from Verizon** profiles, you must use the [Azure CDN rules engine](#) in the **Manage** portal for similar functionality.

### To navigate to the CDN caching rules page:

1. In the Azure portal, select a CDN profile, then select the endpoint for the blob.
2. In the left pane under Settings, select **Caching rules**.



The **Caching rules** page appears.

#### To set a Blob storage service's Cache-Control headers by using global caching rules:

- Under **Global caching rules**, set **Query string caching behavior** to **Ignore query strings** and set **Caching behavior** to **Override**.
- For **Cache expiration duration**, enter 3600 in the **Seconds** box or 1 in the **Hours** box.

This global caching rule sets a cache duration of one hour and affects all requests to the endpoint. It overrides any `Cache-Control` or `Expires` HTTP headers that are sent by the origin server specified by the endpoint.

### 3. Select **Save**.

#### To set a blob file's Cache-Control headers by using custom caching rules:

##### 1. Under **Custom caching rules**, create two match conditions:

- A. For the first match condition, set **Match condition** to **Path** and enter `/blobcontainer1/*` for **Match value**. Set **Caching behavior** to **Override** and enter 4 in the **Hours** box.
- B. For the second match condition, set **Match condition** to **Path** and enter `/blobcontainer1/blob1.txt` for **Match value**. Set **Caching behavior** to **Override** and enter 2 in the **Hours** box.

Custom caching rules						
Create caching rules based on specific match conditions. These rules override the default settings above, and are evaluated from top to down. This means that rules lower on the list can override rules above it in the list, as well as the global caching rules and default behavior. Therefore it makes more sense to have more specific rules towards the bottom of the list so they are not overwritten by a general rule under them. For example a rule for path '/folder/images/*' should be below a rule for path '/folder/*'.						
<a href="#">Move up</a>		<a href="#">Move down</a>		<a href="#">Move to top</a>	<a href="#">Move to bottom</a>	<a href="#">Insert</a>
MATCH CONDITION	MATCH VALUE(S)	CACHING BEHAVIOR	DAYS	HOURS	MINUTES	SECONDS
<input type="checkbox"/> UriPath	<code>/blobcontainer1/*</code>	Override	0	4	0	0
<input type="checkbox"/> UriPath	<code>/blobcontainer1/blob1.txt</code>	Override	0	2	0	0
			<input type="button" value="0"/>	<input type="button" value="0"/>	<input type="button" value="0"/>	<input type="button" value="0"/>

The first custom caching rule sets a cache duration of four hours for any blob files in the `/blobcontainer1` folder on the origin server specified by your endpoint. The second rule overrides the first rule for the `blob1.txt` blob file only and sets a cache duration of two hours for it.

### 2. Select **Save**.

## Setting Cache-Control headers by using Azure PowerShell

### NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Azure PowerShell is one of the quickest and most powerful ways to administer your Azure services. Use the `Get-AzStorageBlob` cmdlet to get a reference to the blob, then set the `.ICloudBlob.Properties.CacheControl` property.

For example:

```

# Create a storage context
$context = New-AzStorageContext -StorageAccountName "<storage account name>" -StorageAccountKey "<storage account key>"

# Get a reference to the blob
$blob = Get-AzStorageBlob -Context $context -Container "<container name>" -Blob "<blob name>"

# Set the CacheControl property to expire in 1 hour (3600 seconds)
$blob.ICloudBlob.Properties.CacheControl = "max-age=3600"

# Send the update to the cloud
$blob.ICloudBlob.SetProperties()

```

**TIP**

You can also use PowerShell to [manage your CDN profiles and endpoints](#).

## Setting Cache-Control headers by using .NET

To specify a blob's `Cache-Control` header by using .NET code, use the [Azure Storage Client Library for .NET](#) to set the `CloudBlob.Properties.CacheControl` property.

For example:

```

class Program
{
    const string connectionString = "<storage connection string>";
    static void Main()
    {
        // Retrieve storage account information from connection string
        CloudStorageAccount storageAccount = CloudStorageAccount.Parse(connectionString);

        // Create a blob client for interacting with the blob service.
        CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

        // Create a reference to the container
        CloudBlobContainer <container name> = blobClient.GetContainerReference("<container name>");

        // Create a reference to the blob
        CloudBlob <blob name> = container.GetBlobReference("<blob name>");

        // Set the CacheControl property to expire in 1 hour (3600 seconds)
        blob.Properties.CacheControl = "max-age=3600";

        // Update the blob's properties in the cloud
        blob.SetProperties();
    }
}

```

**TIP**

There are more .NET code samples available in [Azure Blob Storage Samples for .NET](#).

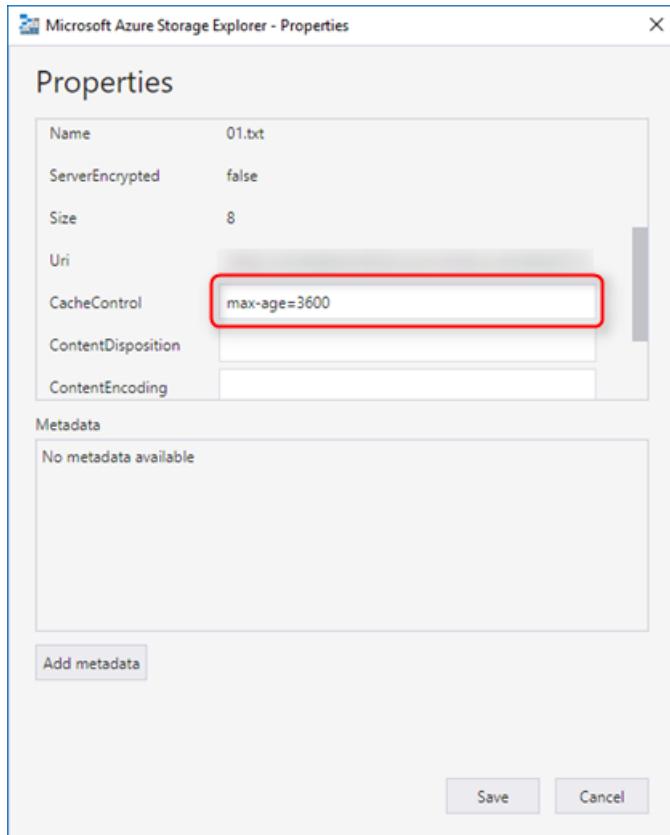
## Setting Cache-Control headers by using other methods

### Azure Storage Explorer

With [Azure Storage Explorer](#), you can view and edit your blob storage resources, including properties such as the `CacheControl` property.

To update the *CacheControl* property of a blob with Azure Storage Explorer:

1. Select a blob, then select **Properties** from the context menu.
2. Scroll down to the *CacheControl* property.
3. Enter a value, then select **Save**.



## Azure Command-Line Interface

With the [Azure Command-Line Interface](#) (CLI), you can manage Azure blob resources from the command line. To set the cache-control header when you upload a blob with the Azure CLI, set the *cacheControl* property by using the `-p` switch. The following example shows how to set the TTL to one hour (3600 seconds):

```
azure storage blob upload -c <connectionstring> -p cacheControl="max-age=3600" .\<blob name> <container name> <blob name>
```

## Azure storage services REST API

You can use the [Azure storage services REST API](#) to explicitly set the *x-ms-blob-cache-control* property by using the following operations on a request:

- [Put Blob](#)
- [Put Block List](#)
- [Set Blob Properties](#)

## Testing the Cache-Control header

You can easily verify the TTL settings of your blobs. With your browser's [developer tools](#), test that your blob includes the `Cache-Control` response header. You can also use a tool such as [Wget](#), [Postman](#), or [Fiddler](#) to examine the response headers.

## Next Steps

- [Learn how to manage expiration of Cloud Service content in Azure CDN](#)
- [Learn about caching concepts](#)

# Securing Azure CDN assets with token authentication

7/5/2019 • 8 minutes to read • [Edit Online](#)

## IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

## Overview

Token authentication is a mechanism that allows you to prevent the Azure Content Delivery Network (CDN) from serving assets to unauthorized clients. Token authentication is typically done to prevent *hotlinking* of content, in which a different website, such as a message board, uses your assets without permission. Hotlinking can have an impact on your content delivery costs. By enabling token authentication on CDN, requests are authenticated by CDN edge server before the CDN delivers the content.

## How it works

Token authentication verifies that requests are generated by a trusted site by requiring requests to contain a token value that holds encoded information about the requester. Content is served to a requester only if the encoded information meets the requirements; otherwise, requests are denied. You can set up the requirements by using one or more of the following parameters:

- Country: Allow or deny requests that originate from the countries/regions specified by their [country code](#).
- URL: Allow only requests that match the specified asset or path.
- Host: Allow or deny requests that use the specified hosts in the request header.
- Referrer: Allow or deny request from the specified referrer.
- IP address: Allow only requests that originated from specific IP address or IP subnet.
- Protocol: Allow or deny requests based on the protocol used to request the content.
- Expiration time: Assign a date and time period to ensure that a link remains valid only for a limited time period.

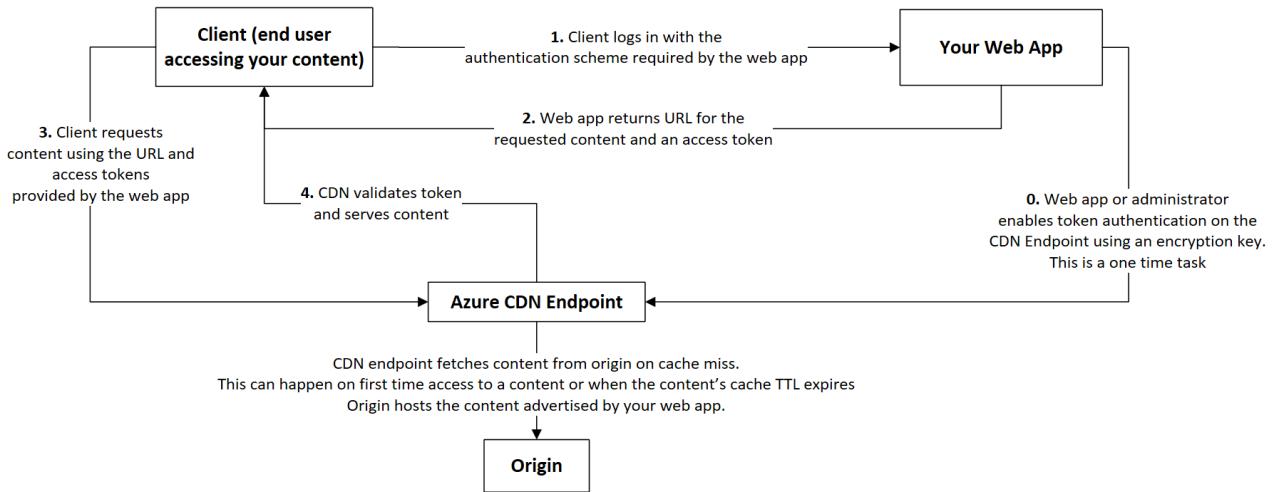
For more information, see the detailed configuration examples for each parameter in [Setting up token authentication](#).

## IMPORTANT

If token authorization is enabled for any path on this account, standard-cache mode is the only mode that can be used for query string caching. For more information, see [Control Azure CDN caching behavior with query strings](#).

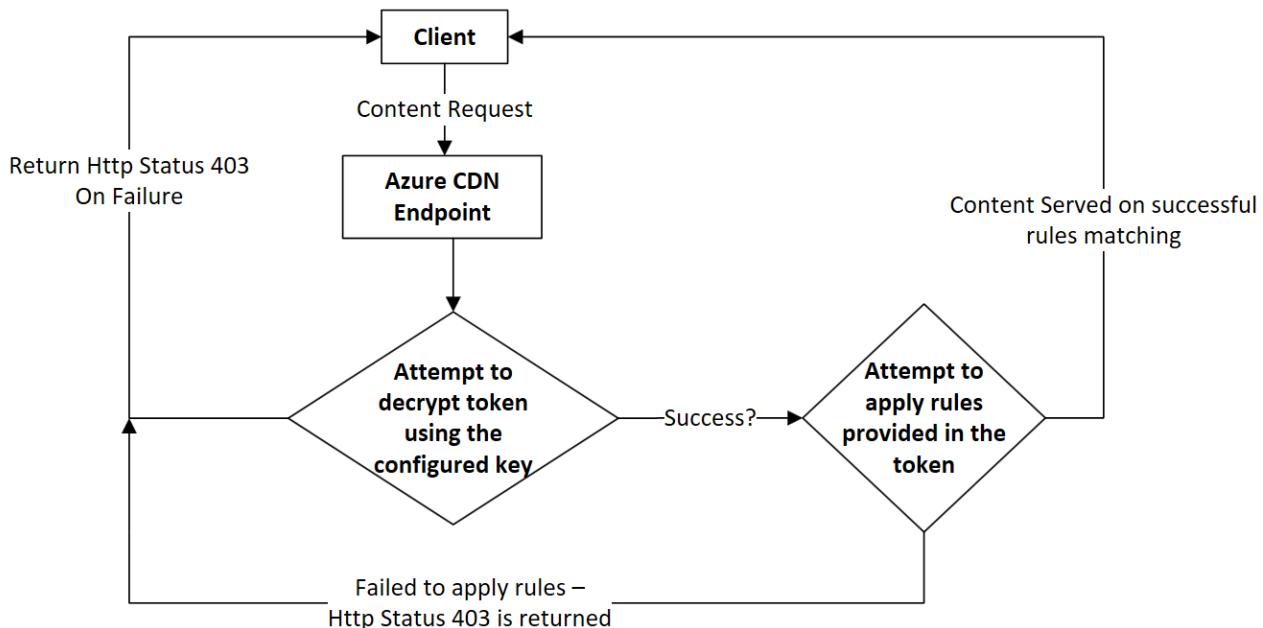
## Reference architecture

The following workflow diagram describes how the CDN uses token authentication to work with your web app.



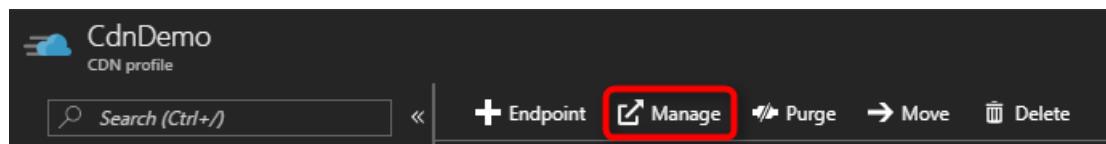
## Token validation logic on CDN endpoint

The following flowchart describes how Azure CDN validates a client request when token authentication is configured on CDN endpoint.



## Setting up token authentication

- From the [Azure portal](#), browse to your CDN profile, then select **Manage** to launch the supplemental portal.



- Hover over **HTTP Large**, then select **Token Auth** in the flyout. You can then set up the encryption key and encryption parameters as follows:
  - Create one or more encryption keys. An encryption key is case-sensitive and can contain any combination of alphanumeric characters. Any other types of characters, including spaces, are not allowed. The maximum length is 250 characters. To ensure that your encryption keys are random, it is recommended that you create them by using the [OpenSSL tool](#).

The OpenSSL tool has the following syntax:

```
rand -hex <key length>
```

For example:

```
OpenSSL> rand -hex 32
```

To avoid downtime, create both a primary and a backup key. A backup key provides uninterrupted access to your content when your primary key is being updated.

- b. Enter a unique encryption key in the **Primary Key** box and optionally enter a backup key in the **Backup Key** box.
- c. Select the minimum encryption version for each key from its **Minimum Encryption Version** list, then select **Update**:
  - **V2**: Indicates that the key can be used to generate version 2.0 and 3.0 tokens. Use this option only if you are transitioning from a legacy version 2.0 encryption key to a version 3.0 key.
  - **V3**: (Recommended) Indicates that the key can only be used to generate version 3.0 tokens.

The screenshot shows a software interface titled "HTTP Large Object Token-Based Authentication". It has a "Primary Key" field containing "12343" and a "Backup Key" field below it. To the right of these are two dropdown menus labeled "Minimum Encryption Version" with options "V3" and "V2". At the bottom right of the window is a "Update" button.

- d. Use the encrypt tool to set up encryption parameters and generate a token. With the encrypt tool, you can allow or deny requests based on expiration time, country/region, referrer, protocol, and client IP (in any combination). Although there is no limit to the number and combination of parameters that can be combined to form a token, the total length of a token is limited to 512 characters.

The screenshot shows a software interface titled "Encrypt Tool". It contains several input fields for encryption parameters: "ec\_expire", "ec\_country\_allow", "ec\_ref\_allow", "ec\_proto\_allow", "ec\_clientip" on the left, and "ec\_url\_allow", "ec\_country\_deny", "ec\_ref\_deny", "ec\_proto\_deny" on the right. At the bottom, there are dropdown menus for "Key To Encrypt" (set to "12343") and "Encryption Version" (set to "V3"), and a large "Encrypt" button.

Enter values for one or more of the following encryption parameters in the **Encrypt Tool** section:

PARAMETER NAME	DESCRIPTION
<b>ec_expire</b>	<p>Assigns an expiration time to a token, after which the token expires. Requests submitted after the expiration time are denied. This parameter uses a Unix timestamp, which is based on the number of seconds since the standard Unix epoch of `1/1/1970 00:00:00 GMT`. (You can use online tools to convert between standard time and Unix time.)</p> <p>For example, if you want the token to expire at <b>12/31/2016 12:00:00 GMT</b>, enter the Unix timestamp value, <b>1483185600</b>.</p>

<p><b>ec_url_allow</b></p>	<p>Allows you to tailor tokens to a particular asset or path. It restricts access to requests whose URL start with a specific relative path. URLs are case-sensitive. Input multiple paths by separating each path with a comma; do not add spaces. Depending on your requirements, you can set up different values to provide different level of access.</p> <p>For example, for the URL  <code>http://www.mydomain.com/pictures/city/strasbourg.png</code>, these requests are allowed for the following input values:</p> <ul style="list-style-type: none"> <li>• Input value ` `/ All requests are allowed.</li> <li>• Input value `/pictures`, the following requests are allowed: <ul style="list-style-type: none"> <li>◦ `http://www.mydomain.com/pictures.png`</li> <li>◦ `http://www.mydomain.com/pictures/city/strasbourg.png`</li> <li>◦ `http://www.mydomain.com/pictures/city/strasbourg.png`</li> </ul> </li> <li>• Input value `/pictures/`: Only requests containing the `/pictures/` path are allowed. For example, `http://www.mydomain.com/pictures/city/strasbourg.png`.</li> <li>• Input value `/pictures/city/strasbourg.png`: Only requests for this specific path and asset are allowed.</li> </ul>
<p><b>ec_country_allow</b></p>	<p>Only allows requests that originate from one or more specified countries/regions. Requests that originate from all other countries/regions are denied. Use a two-letter [ISO 3166 country code](/previous-versions/azure/mt761717(v=azure.100)) for each country and separate each one with a comma; do not add a space. For example, if you want to allow access from only the United States and France, enter `US,FR`.</p>
<p><b>ec_country_deny</b></p>	<p>Denies requests that originate from one or more specified countries/regions. Requests that originate from all other countries/regions are allowed. The implementation is the same as the <b>ec_country_allow</b> parameter. If a country code is present in both the <b>ec_country_allow</b> and <b>ec_country_deny</b> parameters, the <b>ec_country_allow</b> parameter takes precedence.</p>

#### **ec\_ref\_allow**

Only allows requests from the specified referrer. A referrer identifies the URL of the web page that is linked to the resource being requested. Do not include the protocol in the parameter value.

The following types of input are allowed:

- A hostname or a hostname and a path.
- Multiple referrers. To add multiple referrers, separate each referrer with a comma; do not add a space. If you specify a referrer value, but the referrer information is not sent in the request due to the browser configuration, the request is denied by default.
- Requests with missing or blank referrer information. By default, the **ec\_ref\_allow** parameter blocks these types of requests. To allow these requests, enter either the text, "missing", or enter a blank value (by using a trailing comma).
- Subdomains. To allow subdomains, enter an asterisk (\*). For example, to allow all subdomains of `contoso.com`, enter `\*.contoso.com`.

For example, to allow access for requests from `www.contoso.com`, all subdomains under `contoso2.com`, and requests with blank or missing referrers, enter  
`www.contoso.com, *.contoso.com, missing`.

#### **ec\_ref\_deny**

Denies requests from the specified referrer. The implementation is the same as the **ec\_ref\_allow** parameter. If a referrer is present in both the **ec\_ref\_allow** and **ec\_ref\_deny** parameters, the **ec\_ref\_allow** parameter takes precedence.

#### **ec\_proto\_allow**

Only allows requests from the specified protocol. Valid values are `http`, `https`, or `http,https`.

#### **ec\_proto\_deny**

Denies requests from the specified protocol. The implementation is the same as the **ec\_proto\_allow** parameter. If a protocol is present in both the **ec\_proto\_allow** and **ec\_proto\_deny** parameters, the **ec\_proto\_allow** parameter takes precedence.

#### **ec\_clientip**

Restricts access to the specified requester's IP address. Both IPV4 and IPV6 are supported. You can specify either a single request IP address or IP addresses associated with a specific subnet. For example, `11.22.33.0/22` allows requests from IP addresses 11.22.32.1 to 11.22.35.254.

- After you have finished entering encryption parameter values, select a key to encrypt (if you have created both a primary and a backup key) from the **Key To Encrypt** list.
- Select an encryption version from the **Encryption Version** list: **V2** for version 2 or **V3** for version 3 (recommended).
- Select **Encrypt** to generate the token.

After the token is generated, it is displayed in the **Generated Token** box. To use the token, append it

as a query string to the end of the file in your URL path. For example,

`http://www.domain.com/content.mov?a4fbc3710fd3449a7c99986b`.

- h. Optionally, test your token with the decrypt tool so that you can view your token's parameters. Paste the token value in the **Token to Decrypt** box. Select the encryption key to use from the **Key To Decrypt** list, then select **Decrypt**.

After the token is decrypted, its parameters are displayed in the **Original Parameters** box.

- i. Optionally, customize the type of response code that is returned when a request is denied. Select **Enabled**, then select the response code from the **Response Code** list. **Header Name** is automatically set to **Location**. Select **Save** to implement the new response code. For certain response codes, you must also enter the URL of your error page in the **Header Value** box. The **403** response code (Forbidden) is selected by default.

3. Under **HTTP Large**, select **Rules Engine**. You use the rules engine to define paths to apply the feature, enable the token authentication feature, and enable additional token authentication-related capabilities. For more information, see [Rules engine reference](#).

- a. Select an existing rule or create a new rule to define the asset or path for which you want to apply token authentication.
- b. To enable token authentication on a rule, select **Token Auth** from the **Features** list, then select **Enabled**. Select **Update** if you are updating a rule or **Add** if you are creating a rule.



4. In the rules engine, you can also enable additional token authentication-related features. To enable any of the following features, select it from the **Features** list, then select **Enabled**.

- **Token Auth Denial Code:** Determines the type of response that is returned to a user when a request is denied. Rules set here override the response code set in the **Custom Denial Handling** section on the token-based authentication page.
- **Token Auth Ignore URL Case:** Determines whether the URL used to validate the token is case-sensitive.
- **Token Auth Parameter:** Renames the token auth query string parameter that appears in the requested URL.



5. You can customize your token by accessing source code in [GitHub](#). Available languages include:

- C
- C#
- PHP
- Perl
- Java
- Python

## Azure CDN features and provider pricing

For information about features, see [Azure CDN product features](#). For information about pricing, see [Content](#)

[Delivery Network pricing.](#)

# Using Azure CDN with SAS

8/22/2019 • 7 minutes to read • [Edit Online](#)

When you set up a storage account for Azure Content Delivery Network (CDN) to use to cache content, by default anyone who knows the URLs for your storage containers can access the files that you've uploaded. To protect the files in your storage account, you can set the access of your storage containers from public to private. However, if you do so, no one will be able to access your files.

If you want to grant limited access to private storage containers, you can use the Shared Access Signature (SAS) feature of your Azure storage account. A SAS is a URI that grants restricted access rights to your Azure Storage resources without exposing your account key. You can provide a SAS to clients that you do not trust with your storage account key but to whom you want to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified period of time.

With a SAS, you can define various parameters of access to a blob, such as start and expiry times, permissions (read/write), and IP ranges. This article describes how to use SAS in conjunction with Azure CDN. For more information about SAS, including how to create it and its parameter options, see [Using shared access signatures \(SAS\)](#).

## Setting up Azure CDN to work with storage SAS

The following three options are recommended for using SAS with Azure CDN. All options assume that you have already created a working SAS (see prerequisites).

### Prerequisites

To start, create a storage account and then generate a SAS for your asset. You can generate two types of stored access signatures: a service SAS or an account SAS. For more information, see [Types of shared access signatures](#).

After you've generated a SAS token, you can access your blob storage file by appending `?sv=<SAS token>` to your URL. This URL has the following format:

```
https://<account name>.blob.core.windows.net/<container>/<file>?sv=<SAS token>
```

For example:

```
https://democdnstorage1.blob.core.windows.net/container1/demo.jpg?sv=2017-07-29&ss=b&srt=co&sp=r&se=2038-01-02T21:30:49Z&st=2018-01-02T13:30:49Z&spr=https&sig=QehoetQFWUEd1hU5i0MGrHBmE727xYAbKJ15ohSiWI%3D
```

For more information about setting parameters, see [SAS parameter considerations](#) and [Shared access signature parameters](#).

## Option 1: Using SAS with pass-through to blob storage from Azure CDN

This option is the simplest and uses a single SAS token, which is passed from Azure CDN to the origin server.

1. Select an endpoint, select **Caching rules**, then select **Cache every unique URL** from the **Query string caching** list.

2. After you set up SAS on your storage account, you must use the SAS token with the CDN endpoint and origin server URLs to access the file.

The resulting CDN endpoint URL has the following format:

```
https://<endpoint hostname>.azureedge.net/<container>/<file>?sv=<SAS token>
```

For example:

```
https://demoeendpoint.azureedge.net/container1/demo.jpg?sv=2017-07-29&ss=b&srt=c&sp=r&se=2027-12-19T17:35:58Z&st=2017-12-19T09:35:58Z&spr=https&sig=kquaXsAuCLXomN7R00b8CYM13UpDbAHcsRfGOW3Du1M%3D
```

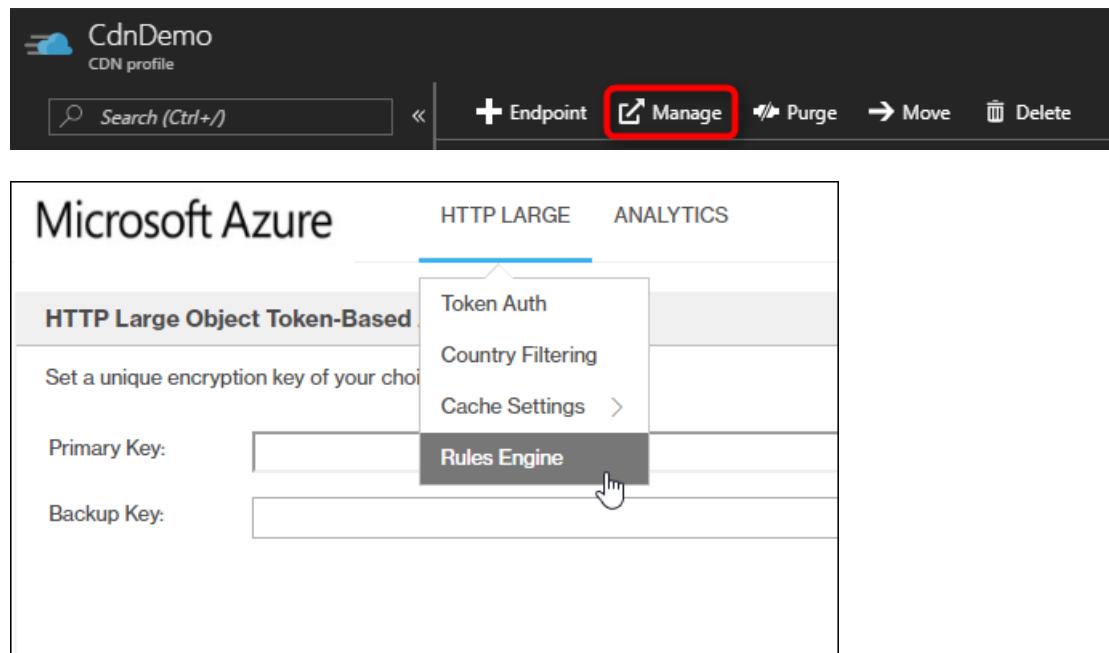
3. Fine-tune the cache duration either by using caching rules or by adding **Cache-Control** headers at the origin server. Because Azure CDN treats the SAS token as a plain query string, as a best practice you should set up a caching duration that expires at or before the SAS expiration time. Otherwise, if a file is

cached for a longer duration than the SAS is active, the file may be accessible from the Azure CDN origin server after the SAS expiration time has elapsed. If this situation occurs, and you want to make your cached file inaccessible, you must perform a purge operation on the file to clear it from the cache. For information about setting the cache duration on Azure CDN, see [Control Azure CDN caching behavior with caching rules](#).

### Option 2: Hidden CDN SAS token using a rewrite rule

This option is available only for **Azure CDN Premium from Verizon** profiles. With this option, you can secure the blob storage at the origin server. You may want to use this option if you don't need specific access restrictions for the file, but want to prevent users from accessing the storage origin directly to improve Azure CDN offload times. The SAS token, which is unknown to the user, is required for anyone accessing files in the specified container of the origin server. However, because of the URL Rewrite rule, the SAS token is not required on the CDN endpoint.

1. Use the [rules engine](#) to create a URL Rewrite rule. New rules take up to 4 hours to propagate.



The screenshot shows the Azure portal interface for managing a CDN profile named "CdnDemo". At the top, there's a navigation bar with "Search (Ctrl+/", "+ Endpoint", "Manage" (which is highlighted with a red box), "Purge", "Move", and "Delete". Below the search bar, there are tabs for "HTTP LARGE" and "ANALYTICS". On the left, under "HTTP Large Object Token-Based", there are fields for "Primary Key" and "Backup Key". A context menu is open over the "Rules Engine" section, with options like "Token Auth", "Country Filtering", "Cache Settings", and "Rules Engine" (which has a hand cursor icon over it).

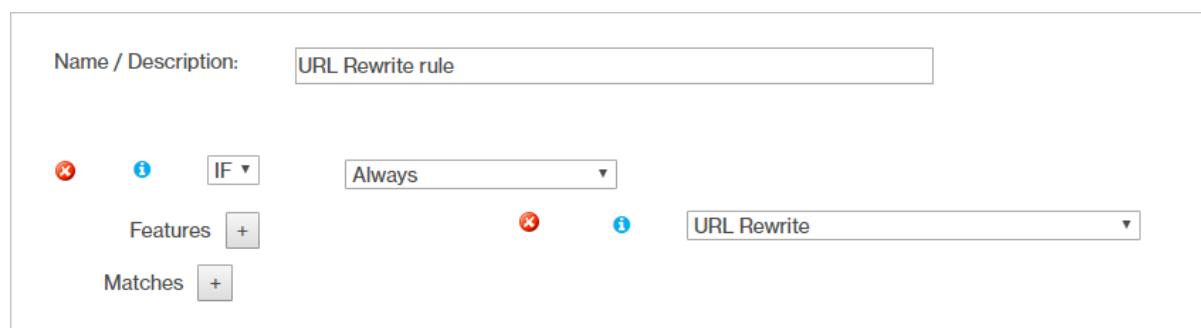
The following sample URL Rewrite rule uses a regular expression pattern with a capturing group and an endpoint named *sasstoragedemo*:

Source:

```
(container1\/.*)
```

Destination:

```
$1?sv=2017-07-29&ss=b&srt=c&sp=r&se=2027-12-19T17:35:58Z&st=2017-12-19T09:35:58Z&spr=https&sig=kquaXsAuCLXomN7R00b8CYM13UpDbAHcsRfGOW3Du1M%3D
```



The screenshot shows the "URL Rewrite rule" configuration dialog. At the top, there's a "Name / Description" field containing "URL Rewrite rule". Below it, there's an "IF" dropdown set to "Always". Underneath the IF dropdown, there are two sections: "Features" and "Matches", each with a "+" button. To the right of the IF dropdown, there's another dropdown set to "URL Rewrite".

2. After the new rule becomes active, anyone can access files in the specified container on the CDN endpoint regardless of whether they're using a SAS token in the URL. Here is the format:

```
https://<endpoint hostname>.azureedge.net/<container>/<file>
```

For example:

```
https://sasstoragedemo.azureedge.net/container1/demo.jpg
```

3. Fine-tune the cache duration either by using caching rules or by adding `Cache-Control` headers at the origin server. Because Azure CDN treats the SAS token as a plain query string, as a best practice you should set up a caching duration that expires at or before the SAS expiration time. Otherwise, if a file is cached for a longer duration than the SAS is active, the file may be accessible from the Azure CDN origin server after the SAS expiration time has elapsed. If this situation occurs, and you want to make your cached file inaccessible, you must perform a purge operation on the file to clear it from the cache. For information about setting the cache duration on Azure CDN, see [Control Azure CDN caching behavior with caching rules](#).

### Option 3: Using CDN security token authentication with a rewrite rule

To use Azure CDN security token authentication, you must have an **Azure CDN Premium from Verizon** profile. This option is the most secure and customizable. Client access is based on the security parameters that you set on the security token. Once you've created and set up the security token, it will be required on all CDN endpoint URLs. However, because of the URL Rewrite rule, the SAS token is not required on the CDN endpoint. If the SAS token later becomes invalid, Azure CDN will no longer be able to revalidate the content from the origin server.

1. [Create an Azure CDN security token](#) and activate it by using the rules engine for the CDN endpoint and path where your users can access the file.

A security token endpoint URL has the following format:

```
https://<endpoint hostname>.azureedge.net/<container>/<file>?<security_token>
```

For example:

```
https://sasstoragedemo.azureedge.net/container1/demo.jpg?
a4fb3710fd3449a7c99986bkquaXsAuCLXomN7R00b8CYM13UpDbAHcsRfG0W3Du1M%3D
```

The parameter options for a security token authentication are different than the parameter options for a SAS token. If you choose to use an expiration time when you create a security token, you should set it to the same value as the expiration time for the SAS token. Doing so ensures that the expiration time is predictable.

2. Use the [rules engine](#) to create a URL Rewrite rule to enable SAS token access to all blobs in the container. New rules take up to 4 hours to propagate.

The following sample URL Rewrite rule uses a regular expression pattern with a capturing group and an endpoint named *sasstoragedemo*:

Source:

```
(container1\/.*)
```

Destination:

3. If you renew the SAS, ensure that you update the Url Rewrite rule with the new SAS token.

## SAS parameter considerations

Because SAS parameters are not visible to Azure CDN, Azure CDN cannot change its delivery behavior based on them. The defined parameter restrictions apply only on requests that Azure CDN makes to the origin server, not for requests from the client to Azure CDN. This distinction is important to consider when you set SAS parameters. If these advanced capabilities are required and you are using [Option 3](#), set the appropriate restrictions on the Azure CDN security token.

SAS PARAMETER NAME	DESCRIPTION
Start	The time that Azure CDN can begin to access the blob file. Due to clock skew (when a clock signal arrives at different times for different components), choose a time 15 minutes earlier if you want the asset to be available immediately.
End	The time after which Azure CDN can no longer access the blob file. Previously cached files on Azure CDN are still accessible. To control the file expiry time, either set the appropriate expiry time on the Azure CDN security token or purge the asset.
Allowed IP addresses	Optional. If you are using <b>Azure CDN from Verizon</b> , you can set this parameter to the ranges defined in <a href="#">Azure CDN from Verizon Edge Server IP Ranges</a> . If you are using <b>Azure CDN from Akamai</b> , you cannot set the IP ranges parameter because the IP addresses are not static.
Allowed protocols	The protocol(s) allowed for a request made with the account SAS. The HTTPS setting is recommended.

## Next steps

For more information about SAS, see the following articles:

- [Using shared access signatures \(SAS\)](#)

- Shared Access Signatures, Part 2: Create and use a SAS with Blob storage

For more information about setting up token authentication, see [Securing Azure Content Delivery Network assets with token authentication](#).

# Using Azure CDN with CORS

11/20/2019 • 4 minutes to read • [Edit Online](#)

## What is CORS?

CORS (Cross Origin Resource Sharing) is an HTTP feature that enables a web application running under one domain to access resources in another domain. In order to reduce the possibility of cross-site scripting attacks, all modern web browsers implement a security restriction known as [same-origin policy](#). This prevents a web page from calling APIs in a different domain. CORS provides a secure way to allow one origin (the origin domain) to call APIs in another origin.

## How it works

There are two types of CORS requests, *simple requests* and *complex requests*.

### For simple requests:

1. The browser sends the CORS request with an additional **Origin** HTTP request header. The value of this header is the origin that served the parent page, which is defined as the combination of *protocol*, *domain*, and *port*. When a page from <https://www.contoso.com> attempts to access a user's data in the [fabrikam.com](https://fabrikam.com) origin, the following request header would be sent to fabrikam.com:

```
Origin: https://www.contoso.com
```

2. The server may respond with any of the following:

- An **Access-Control-Allow-Origin** header in its response indicating which origin site is allowed. For example:

```
Access-Control-Allow-Origin: https://www.contoso.com
```

- An HTTP error code such as 403 if the server does not allow the cross-origin request after checking the Origin header
- An **Access-Control-Allow-Origin** header with a wildcard that allows all origins:

```
Access-Control-Allow-Origin: *
```

### For complex requests:

A complex request is a CORS request where the browser is required to send a *preflight request* (that is, a preliminary probe) before sending the actual CORS request. The preflight request asks the server permission if the original CORS request can proceed and is an **OPTIONS** request to the same URL.

#### TIP

For more details on CORS flows and common pitfalls, view the [Guide to CORS for REST APIs](#).

## Wildcard or single origin scenarios

CORS on Azure CDN will work automatically with no additional configuration when the **Access-Control-Allow-Origin** header is set to wildcard (\*) or a single origin. The CDN will cache the first response and subsequent requests will use the same header.

If requests have already been made to the CDN prior to CORS being set on your origin, you will need to purge content on your endpoint content to reload the content with the **Access-Control-Allow-Origin** header.

## Multiple origin scenarios

If you need to allow a specific list of origins to be allowed for CORS, things get a little more complicated. The problem occurs when the CDN caches the **Access-Control-Allow-Origin** header for the first CORS origin. When a different CORS origin makes a subsequent request, the CDN will serve the cached **Access-Control-Allow-Origin** header, which won't match. There are several ways to correct this.

### Azure CDN standard profiles

On Azure CDN Standard from Microsoft, you can create a rule in the [Standard rules engine](#) to check the **Origin** header on the request. If it's a valid origin, your rule will set the **Access-Control-Allow-Origin** header with the desired value. In this case, the **Access-Control-Allow-Origin** header from the file's origin server is ignored and the CDN's rules engine completely manages the allowed CORS origins.

The screenshot shows the Azure CDN Standard rules engine interface. A single rule is defined:

Name *	CORS	✓		
+ Add condition ✓ + Add action ✓   ↴ ↑ ↓ ↵   ⏎				
If	Header name *	Operator *	Header value *	Case transform
Request header	Origin	Equals ✓	https://contoso.com ✓	No transform ✓
Then	Action *	HTTP header name *	HTTP header value	
Modify response header	Overwrite ✓	Access-Control-Allow-... ✓	https://contoso.com ✓	✓

#### TIP

You can add additional actions to your rule to modify additional response headers, such as [Access-Control-Allow-Methods](#).

On **Azure CDN Standard from Akamai**, the only mechanism to allow for multiple origins without the use of the wildcard origin is to use [query string caching](#). Enable the query string setting for the CDN endpoint and then use a unique query string for requests from each allowed domain. Doing so will result in the CDN caching a separate object for each unique query string. This approach is not ideal, however, as it will result in multiple copies of the same file cached on the CDN.

### Azure CDN Premium from Verizon

Using the Verizon Premium rules engine, You'll need to [create a rule](#) to check the **Origin** header on the request. If it's a valid origin, your rule will set the **Access-Control-Allow-Origin** header with the origin provided in the request. If the origin specified in the **Origin** header is not allowed, your rule should omit the **Access-Control-Allow-Origin** header, which will cause the browser to reject the request.

There are two ways to do this with the Premium rules engine. In both cases, the **Access-Control-Allow-Origin** header from the file's origin server is ignored and the CDN's rules engine completely manages the allowed CORS origins.

#### One regular expression with all valid origins

In this case, you'll create a regular expression that includes all of the origins you want to allow:

```
https?:\/\/(www\.contoso\.com|contoso\.com|www\.microsoft\.com|microsoft\.com\.com)$
```

#### TIP

Azure CDN Premium from Verizon uses Perl Compatible Regular Expressions as its engine for regular expressions. You can use a tool like [Regular Expressions 101](#) to validate your regular expression. Note that the "/" character is valid in regular expressions and doesn't need to be escaped, however, escaping that character is considered a best practice and is expected by some regex validators.

If the regular expression matches, your rule will replace the **Access-Control-Allow-Origin** header (if any) from the origin with the origin that sent the request. You can also add additional CORS headers, such as **Access-Control-Allow-Methods**.

The screenshot shows the Azure CDN rules editor interface. At the top, there is a condition section labeled "IF" with a dropdown menu set to "Request Header Regex". Below it, the "Name" field is set to "Origin" and the "Matches" dropdown is set to "Value". The "Value" field contains the regular expression "https://www\\.contoso\\.com|contoso". The "Ignore Case" checkbox is checked. Below the condition, there is a "Features" section containing four "Modify Client Response Header" actions. Each action has "Overwrite" selected for "Action". The "Name" field for all four actions is "Access-Control-Allow-Origin". The "Value" field for the first action is "{http\_origin}", for the second is "\*", for the third is "GET, HEAD, OPTIONS", and for the fourth is "\*". A "Matches" button is located at the bottom left of the feature section.

#### Request header rule for each origin.

Rather than regular expressions, you can instead create a separate rule for each origin you wish to allow using the **Request Header Wildcard** match condition. As with the regular expression method, the rules engine alone sets the CORS headers.

The screenshot shows the Azure CDN rules editor interface. At the top, there is a condition section labeled "IF" with a dropdown menu set to "Request Header Wildcard". Below it, the "Name" field is set to "Origin" and the "Matches" dropdown is set to "Value(s)". The "Value(s)" field contains the wildcard value "\*://contoso.com". The "Ignore Case" checkbox is checked. Below the condition, there is a "Features" section containing one "Modify Client Response Header" action. The "Action" is set to "Overwrite". The "Name" field is "Access-Control-Allow-Origin" and the "Value" field is "{http\_origin}". A "Matches" button is located at the bottom left of the feature section.

#### TIP

In the example above, the use of the wildcard character \* tells the rules engine to match both HTTP and HTTPS.

# Monitor the health of Azure CDN resources

7/5/2019 • 2 minutes to read • [Edit Online](#)

Azure CDN Resource health is a subset of [Azure resource health](#). You can use Azure resource health to monitor the health of CDN resources and receive actionable guidance to troubleshoot problems.

## IMPORTANT

Azure CDN resource health only currently accounts for the health of global CDN delivery and API capabilities. Azure CDN resource health does not verify individual CDN endpoints.

The signals that feed Azure CDN resource health may be up to 15 minutes delayed.

## How to find Azure CDN resource health

1. In the [Azure portal](#), browse to your CDN profile.

2. Click the **Settings** button.

The screenshot shows the Azure portal interface for a CDN profile named 'CdnDemo'. The top navigation bar includes 'Settings' (which is highlighted with a red box), 'Endpoint', 'Manage', 'Purge', and 'Delete'. The left sidebar has sections for 'Overview' (selected), 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'SETTINGS' (with 'Locks' and 'Automation script'), 'GENERAL' (with 'Quickstart', 'Properties', and 'Endpoints'), and 'SUPPORT + TROUBLESHOOTING' (with 'New support request'). The main content area is titled 'Essentials' and shows details like Resource group 'CdnDemoRG', Pricing tier 'Standard Verizon', Status 'Active', Location 'Central US', Subscription name 'msdn', and Subscription ID '376c977e-dcb5-4959-a52f-7d6999058559'. Below this is a section titled 'Endpoints' which lists four entries:

HOSTNAME	STATUS	PROTOCOL
cdndocdemo.azureedge.net	Running	HTTP, HTTPS
cdndocdemoarm.azureedge...	Running	HTTP, HTTPS
cdndocdemarma.azureedge...	Running	HTTP, HTTPS
cdndocdemarbm.azureedge...	Running	HTTP, HTTPS

3. Under *Support + troubleshooting*, click **Resource health**.



Search (Ctrl+ /)



Overview



Activity log



Access control (IAM)



Tags



Diagnose and solve problems

#### SETTINGS



Locks



Automation script

#### GENERAL



Quickstart



Properties



Endpoints

#### SUPPORT + TROUBLESHOOTING



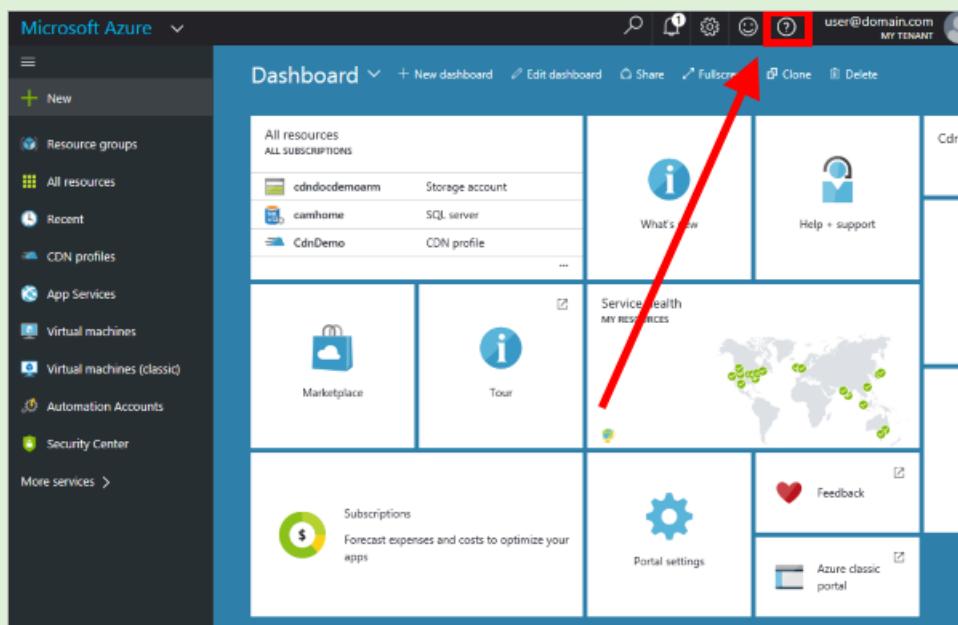
Resource health



New support request

## TIP

You can also find CDN resources listed in the *Resource health* tile in the *Help + support* blade. You can quickly get to *Help + support* by clicking the circled ? in the upper right corner of the portal.



## Azure CDN-specific messages

Statuses related to Azure CDN resource health can be found below.

MESSAGE	RECOMMENDED ACTION
You may have stopped, removed, or misconfigured one or more of your CDN endpoints	You may have stopped, removed, or misconfigured one or more of your CDN endpoints.
We are sorry, the CDN management service is currently unavailable	Check back here for status updates; If your problem persists after the expected resolution time, contact support.
We're sorry, your CDN endpoints may be impacted by ongoing issues with some of our CDN providers	Check back here for status updates; Use the Troubleshoot tool to learn how to test your origin and CDN endpoint; If your problem persists after the expected resolution time, contact support.
We're sorry, CDN endpoint configuration changes are experiencing propagation delays	Check back here for status updates; If your configuration changes are not fully propagated in the expected time, contact support.
We're sorry, we are experiencing issues loading the supplemental portal	Check back here for status updates; If your problem persists after the expected resolution time, contact support.
We are sorry, we are experiencing issues with some of our CDN providers	Check back here for status updates; If your problem persists after the expected resolution time, contact support.

## Next steps

- [Read an overview of Azure resource health](#)
- [Troubleshoot issues with CDN compression](#)
- [Troubleshoot issues with 404 errors](#)



# Override HTTP behavior using the Azure CDN from Verizon Premium rules engine

11/14/2019 • 2 minutes to read • [Edit Online](#)

## IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

## Overview

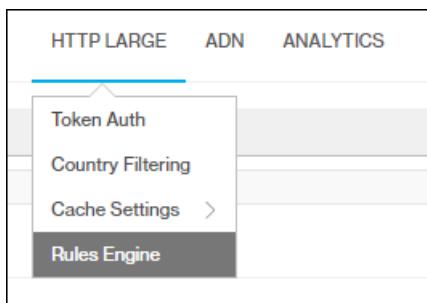
The Azure CDN rules engine allows you to customize how HTTP requests are handled. For example, blocking the delivery of certain content types, defining a caching policy, or modifying an HTTP header. This tutorial demonstrates how to create a rule that changes the caching behavior of CDN assets. For more information about the rules engine syntax, see [Azure CDN rules engine reference](#).

## Access

To access the rules engine, you must first select **Manage** from the top of the **CDN profile** page to access the Azure CDN management page. Depending on whether your endpoint is optimized for dynamic site acceleration (DSA), you then access the rules engine with the set of rules appropriate for your type of endpoint:

- Endpoints optimized for general web delivery or other non-DSA optimization:

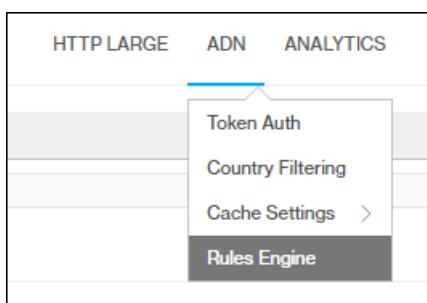
Select the **HTTP Large** tab, then select **Rules Engine**.



- Endpoints optimized for DSA:

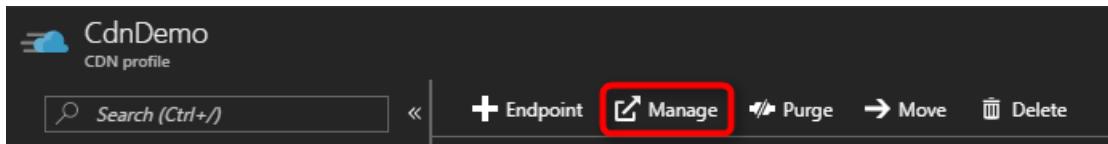
Select the **ADN** tab, then select **Rules Engine**.

ADN is a term used by Verizon to specify DSA content. Any rules you create here are ignored by any endpoints in your profile that are not optimized for DSA.



# Tutorial

- From the **CDN profile** page, select **Manage**.



The CDN management portal opens.

- Select the **HTTP Large** tab, then select **Rules Engine**.

The options for a new rule are displayed.

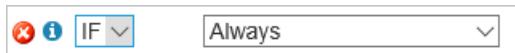
A screenshot of the 'New Rule' dialog box. It includes a note about rule approval, a 'Name / Description' input field, and 'Add' and 'Cancel' buttons. Below these are dropdown menus for 'IF' (set to 'Always') and 'Features' (with a '+' button) and 'Matches' (with a '+' button).

## IMPORTANT

The order in which multiple rules are listed affects how they are handled. A subsequent rule may override the actions specified by a previous rule.

- Enter a name in the **Name / Description** textbox.

- Identify the type of requests the rule applies to. Use the default match condition, **Always**.



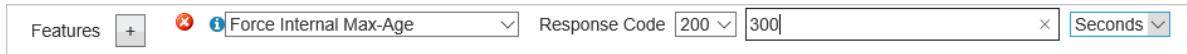
## NOTE

Multiple match conditions are available in the dropdown list. For information about the currently selected match condition, select the blue informational icon to its left.

For a detailed list of conditional expressions, see [Rules engine conditional expressions](#).

For a detailed list of match conditions, see [Rules engine match conditions](#).

- To add a new feature, select the **+** button next to **Features**. In the dropdown on the left, select **Force Internal Max-Age**. In the textbox that appears, enter **300**. Do not change the remaining default values.



#### NOTE

Multiple features are available in the dropdown list. For information about the currently selected feature, select the blue informational icon to its left.

For **Force Internal Max-Age**, the asset's `Cache-Control` and `Expires` headers are overridden to control when the CDN edge node refreshes the asset from the origin. In this example, the CDN edge node caches the asset for 300 seconds, or 5 minutes, before it refreshes the asset from its origin.

For a detailed list of features, see [Rules engine features](#).

6. Select **Add** to save the new rule. The new rule is now awaiting approval. After it has been approved, the status changes from **Pending XML** to **Active XML**.

#### IMPORTANT

Rules changes can take up to 10 minutes to propagate through Azure CDN.

## See also

- [Azure CDN overview](#)
- [Rules engine reference](#)
- [Rules engine match conditions](#)
- [Rules engine conditional expressions](#)
- [Rules engine features](#)
- [Azure Fridays: Azure CDN's powerful new premium features](#) (video)

# Set up failover across multiple Azure CDN endpoints with Azure Traffic Manager

11/14/2019 • 3 minutes to read • [Edit Online](#)

When you configure Azure Content Delivery Network (CDN), you can select the optimal provider and pricing tier for your needs. Azure CDN, with its globally distributed infrastructure, by default creates local and geographic redundancy and global load balancing to improve service availability and performance. If a location is not available to serve content, requests are automatically routed to another location and the optimal POP (based on such factors as request location and server load) is used to serve each client request.

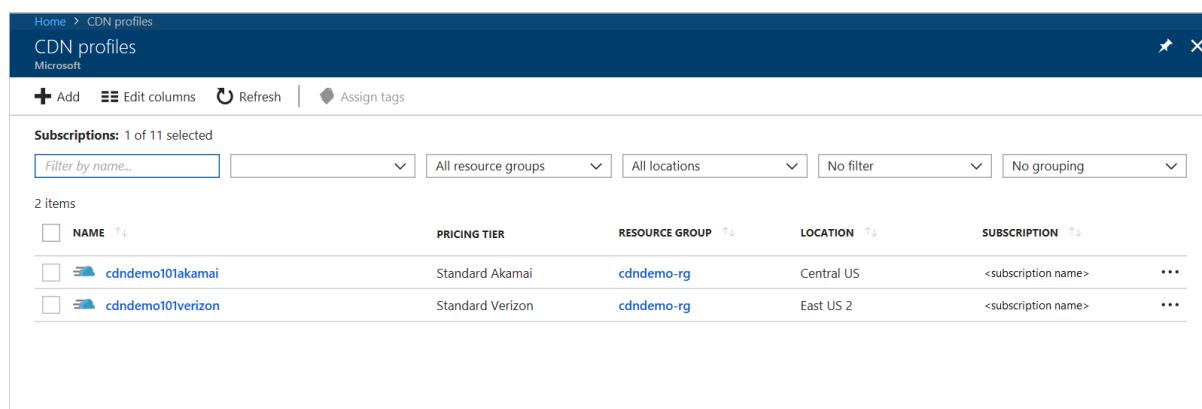
If you have multiple CDN profiles, you can further improve availability and performance with Azure Traffic Manager. You can use Azure Traffic Manager with Azure CDN to load balance among multiple CDN endpoints for failover, geo-load balancing, and other scenarios. In a typical failover scenario, all client requests are first directed to the primary CDN profile; if the profile is not available, requests are then passed to the secondary CDN profile until your primary CDN profile is back online. Using Azure Traffic Manager in this way ensures your web application is always available.

This article provides guidance and an example of how to set up failover with **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profiles.

## Set up Azure CDN

Create two or more Azure CDN profiles and endpoints with different providers.

1. Create an **Azure CDN Standard from Verizon** and **Azure CDN Standard from Akamai** profile by following the steps in [Create a new CDN profile](#).



NAME	PRICING TIER	RESOURCE GROUP	LOCATION	SUBSCRIPTION
cdndemo101akamai	Standard Akamai	cdndemo-rg	Central US	<subscription name>
cdndemo101verizon	Standard Verizon	cdndemo-rg	East US 2	<subscription name>

2. In each of the new profiles, create at least one endpoint by following the steps in [Create a new CDN endpoint](#).

## Set up Azure Traffic Manager

Create an Azure Traffic Manager profile and set up load balancing across your CDN endpoints.

1. Create an Azure Traffic Manager profile by following the steps in [Create a Traffic Manager profile](#).

For **Routing method**, select **Priority**.

2. Add your CDN endpoints in your Traffic Manager profile by following the steps in [Add Traffic Manager endpoints](#)

For **Type**, select **External endpoints**. For **Priority**, enter a number.

For example, create *cdndemo101akamai.azureedge.net* with a priority of 1 and *cdndemo101verizon.azureedge.net* with a priority of 2.

NAME	STATUS	MONITOR STATUS	TYPE	PRIORITY
AkamaiEndpoint	Enabled	Checking endpoint	External endpoint	1
VerizonEndpoint	Enabled	Checking endpoint	External endpoint	2

## Set up custom domain on Azure CDN and Azure Traffic Manager

After you set up your CDN and Traffic Manager profiles, follow these steps to add DNS mapping and register custom domain to the CDN endpoints. For this example, the custom domain name is *cdndemo101.dustydogpetcare.online*.

1. Go to the web site for the domain provider of your custom domain, such as GoDaddy, and create two DNS CNAME entries.
  - a. For the first CNAME entry, map your custom domain, with the *cdnverify* subdomain, to your CDN endpoint. This entry is a required step to register the custom domain to the CDN endpoint that you added to Traffic Manager in step 2.

For example:

```
cdnverify.cdndemo101.dustydogpetcare.online CNAME cdnverify.cdndemo101akamai.azureedge.net
```

- b. For the second CNAME entry, map your custom domain, without the *cdnverify* subdomain, to your CDN endpoint. This entry maps the custom domain to Traffic Manager.

For example:

```
cdndemo101.dustydogpetcare.online CNAME cdndemo101.trafficmanager.net
```

### NOTE

If your domain is currently live and cannot be interrupted, do this step last. Verify that the CDN endpoints and traffic manager domains are live before you update your custom domain DNS to Traffic Manager.

2. From your Azure CDN profile, select the first CDN endpoint (Akamai). Select **Add custom domain** and input *cdndemo101.dustydogpetcare.online*. Verify that the checkmark to validate the custom domain is green.

Azure CDN uses the *cdnverify* subdomain to validate the DNS mapping to complete this registration

process. For more information, see [Create a CNAME DNS record](#). This step enables Azure CDN to recognize the custom domain so that it can respond to its requests.

**NOTE**

To enable SSL on an **Azure CDN from Akamai** profiles, you must directly cname the custom domain to your endpoint. cdnverify for enabling SSL is not yet supported.

3. Return to the web site for the domain provider of your custom domain and update the first DNS mapping you created in so that the custom domain is mapped to your second CDN endpoint.

For example:

```
cdnverify.cdndemo101.dustydogpetcare.online CNAME cdnverify.cdndemo101verizon.azureedge.net
```

4. From your Azure CDN profile, select the second CDN endpoint (Verizon) and repeat step 2. Select **Add custom domain**, and input *cdndemo101.dustydogpetcare.online*.

After you complete these steps, your multi-CDN service with failover capabilities is set up with Azure Traffic Manager. You'll be able to access the test URLs from your custom domain. To test the functionality, disable the primary CDN endpoint and verify that the request is correctly moved over to the secondary CDN endpoint.

## Next steps

You can also set up other routing methods, such as geographic, to balance the load among different CDN endpoints. For more information, see [Configure the geographic traffic routing method using Traffic Manager](#).

# Analyze Azure CDN usage patterns

7/5/2019 • 2 minutes to read • [Edit Online](#)

After you enable CDN for your application, you can monitor CDN usage, check the health of your delivery, and troubleshoot potential issues. Azure CDN provides these capabilities in the following ways:

## Core analytics via Azure diagnostic logs

Core analytics is available for CDN endpoints for all pricing tiers. Azure diagnostics logs allow core analytics to be exported to Azure storage, event hubs, or Azure Monitor logs. Azure Monitor logs offers a solution with graphs that are user-configurable and customizable. For more information about Azure diagnostic logs, see [Azure diagnostic logs](#).

## Verizon core reports

As an Azure CDN user with an **Azure CDN Standard from Verizon** or **Azure CDN Premium from Verizon** profile, you can view Verizon core reports in the Verizon supplemental portal. Verizon core reports is accessible via the **Manage** option from the Azure portal and offers a variety of graphs and views. For more information, see [Core Reports from Verizon](#).

## Verizon custom reports

As an Azure CDN user with an **Azure CDN Standard from Verizon** or **Azure CDN Premium from Verizon** profile, you can view Verizon custom reports in the Verizon supplemental portal. Verizon custom reports is accessible via the **Manage** option from the Azure portal. The Verizon custom reports page shows the number of hits or data transferred for each edge CName belonging to an Azure CDN profile. The data can be grouped by HTTP response code or cache status over any period of time. For more information, see [Custom Reports from Verizon](#).

## Azure CDN Premium from Verizon reports

With **Azure CDN Premium from Verizon**, you can also access the following reports:

- [Advanced HTTP reports](#)
- [Real-time stats](#)
- [Edge node performance](#)

# Azure diagnostic logs

7/5/2019 • 13 minutes to read • [Edit Online](#)

With Azure diagnostic logs, you can view core analytics and save them into one or more destinations including:

- Azure Storage account
- Azure Event Hubs
- [Log Analytics workspace](#)

This feature is available on CDN endpoints for all pricing tiers.

Azure diagnostics logs allow you to export basic usage metrics from your CDN endpoint to a variety of sources so that you can consume them in a customized way. For example, you can do the following types of data export:

- Export data to blob storage, export to CSV, and generate graphs in Excel.
- Export data to Event Hubs and correlate with data from other Azure services.
- Export data to Azure Monitor logs and view data in your own Log Analytics workspace

The following diagram shows a typical CDN core analytics view of data.

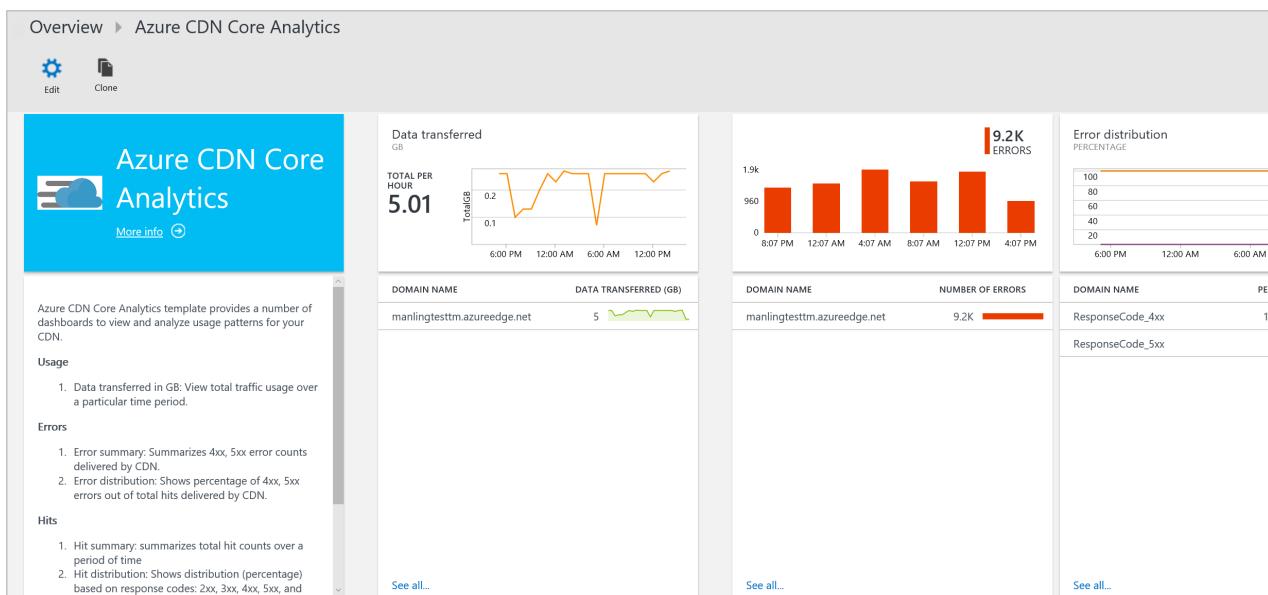


Figure 1 - CDN core analytics view

For more information about diagnostic logs, see [Diagnostic Logs](#).

## NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of logs in Azure Monitor. See [Azure Monitor terminology changes](#) for details.

## Enable logging with the Azure portal

Follow these steps enable logging with CDN core analytics:

Sign in to the [Azure portal](#). If you don't already have enabled CDN for your workflow, [Create an Azure CDN profile and endpoint](#) before you continue.

1. In the Azure portal, navigate to **CDN profile**.
2. In the Azure portal, search for a CDN profile or select one from your dashboard. Then, select the CDN endpoint for which you want to enable diagnostics logs.

The screenshot shows the Azure portal interface for managing a CDN profile. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main area shows 'Essentials' for the 'ManlingAzureCDN' resource group, including status (Active), location (West US), and subscription information. Below that is the 'Endpoints' section, which lists two endpoints with their hostnames, status (Running), protocol (HTTP, HTTPS), origin type (Storage), and custom domains. On the far right, under the 'MONITORING' section, the 'Diagnostics logs' option is highlighted with a red box.

3. Select **Diagnostics logs** in the MONITORING section.

The **Diagnostics logs** page appears.

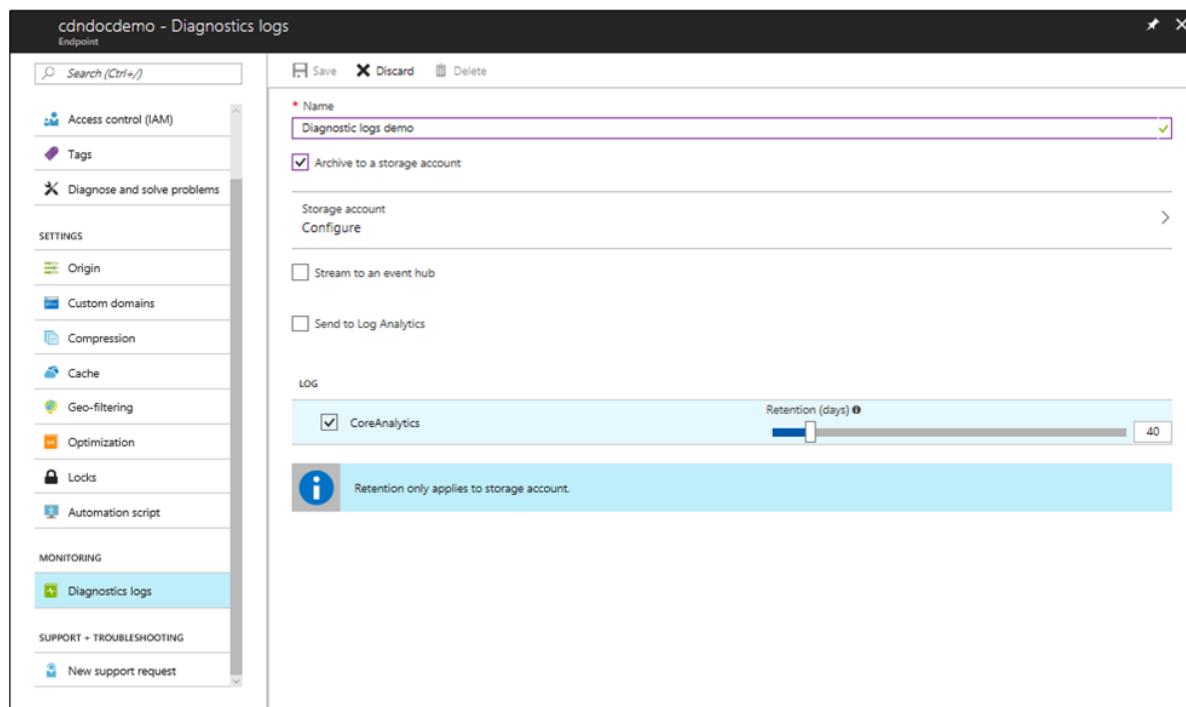
The screenshot shows the 'Diagnostics logs' configuration page. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main area contains fields for 'Name' (with a placeholder 'I'), checkboxes for 'Archive to a storage account', 'Stream to an event hub', and 'Send to Log Analytics', and a 'LOG' section with a checkbox for 'CoreAnalytics'. The 'Diagnostics logs' option in the sidebar is highlighted with a blue box.

## Enable logging with Azure Storage

To use a storage account to store the logs, follow these steps:

1. For **Name**, enter a name for your diagnostic log settings.

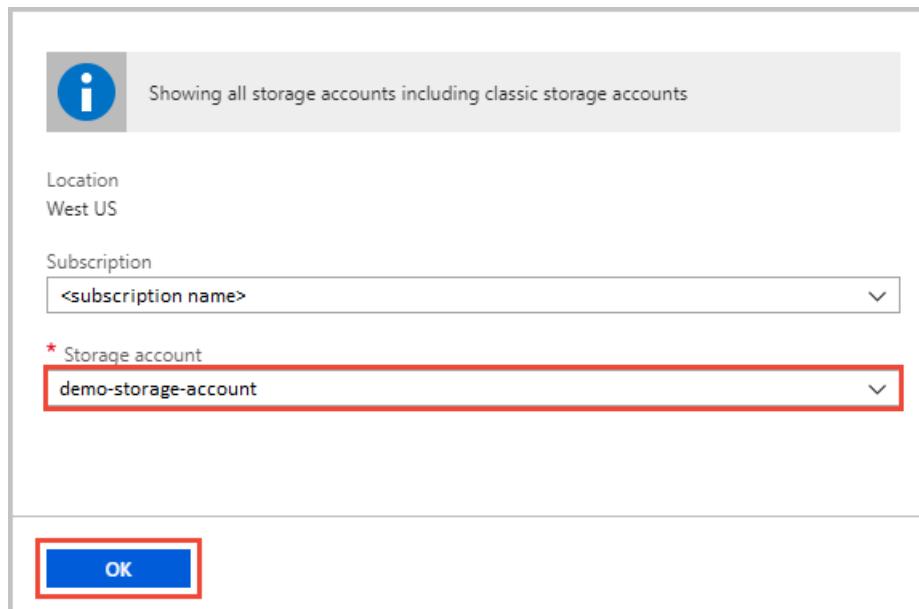
2. Select **Archive to a storage account**, then select **CoreAnalytics**.
3. For **Retention (days)**, choose the number of retention days. A retention of zero days stores the logs indefinitely.



4. Select **Storage account**.

The **Select a storage account** page appears.

5. Select a storage account from the drop-down list, then select **OK**.

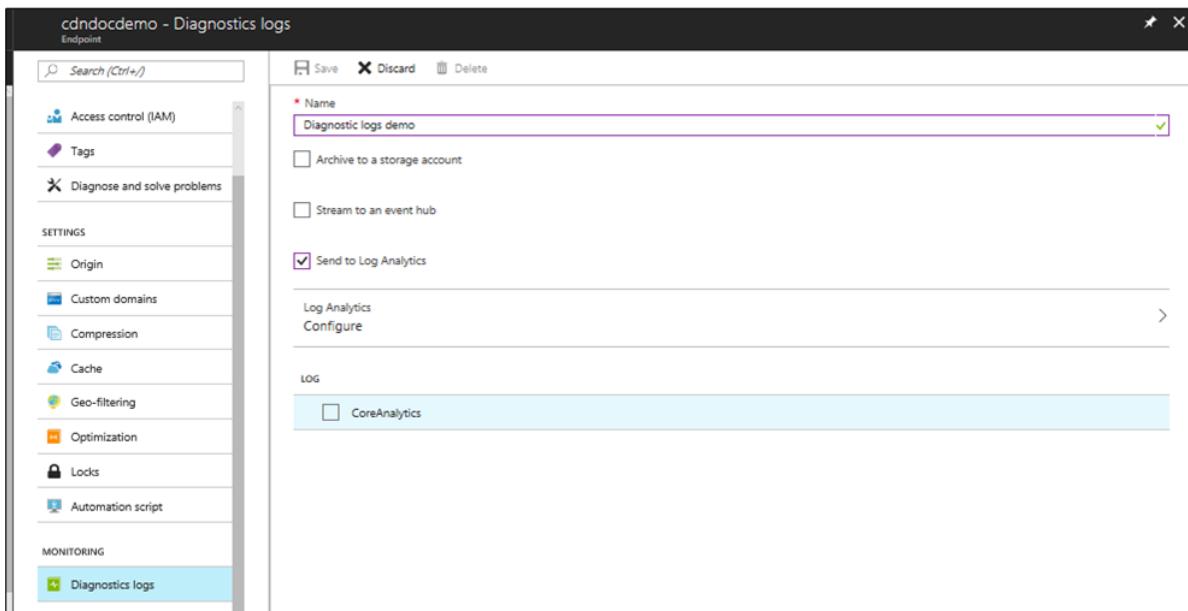


6. After you have finished making your diagnostic log settings, select **Save**.

### Logging with Azure Monitor

To use Azure Monitor to store the logs, follow these steps:

1. From the **Diagnostics logs** page, select **Send to Log Analytics**.

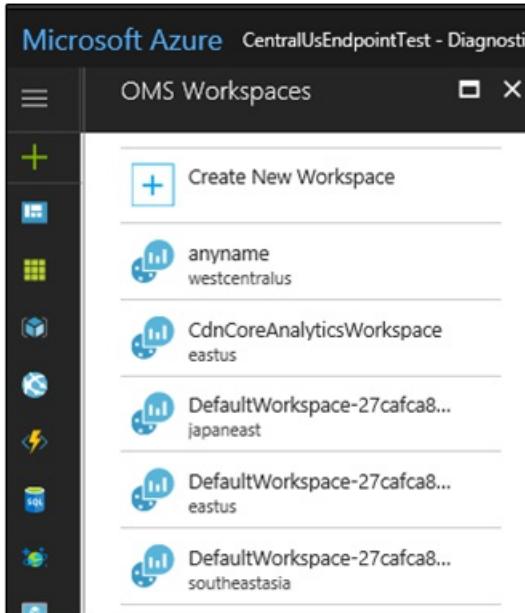


2. Select **Configure** to configure Azure Monitor logging.

The **Log Analytics workspaces** page appears.

**NOTE**

OMS workspaces are now referred to as Log Analytics workspaces.

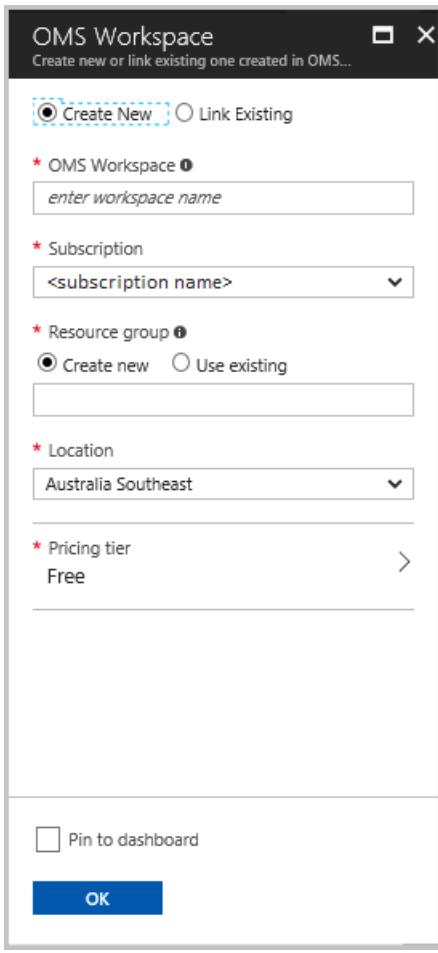


3. Select **Create New Workspace**.

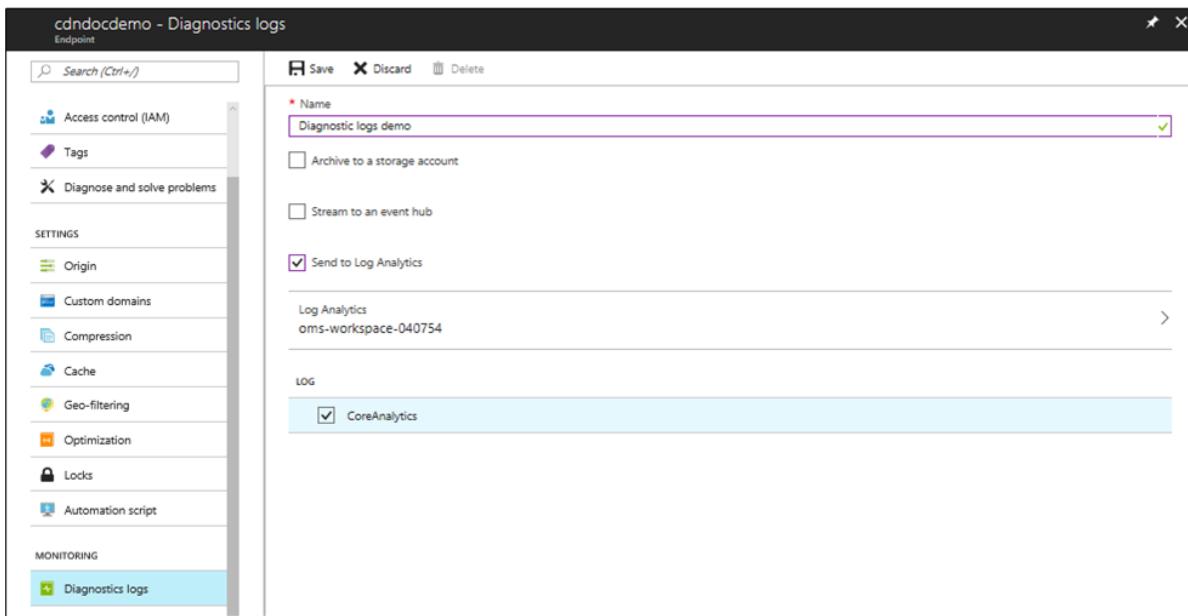
The **Log Analytics workspace** page appears.

**NOTE**

OMS workspaces are now referred to as Log Analytics workspaces.



4. For **Log Analytics workspace**, enter a Log Analytics workspace name. The Log Analytics workspace name must be unique and contain only letters, numbers, and hyphens; spaces and underscores are not allowed.
5. For **Subscription**, select an existing subscription from the drop-down list.
6. For **Resource group**, create a new resource group or select an existing one.
7. For **Location**, select a location from the list.
8. Select **Pin to dashboard** if you want to save the log configuration to your dashboard.
9. Select **OK** to complete the configuration.
10. After your workspace is created, you're returned to the **Diagnostic logs** page. Confirm the name of your new Log Analytics workspace.



11. Select **CoreAnalytics**, then select **Save**.

12. To view the new Log Analytics workspace, select **Core analytics** from your CDN endpoint page.

Your Log Analytics workspace is now ready to log data. In order to consume that data, you must use a [Azure Monitor logs solution](#), covered later in this article.

For more information about log data delays, see [Log data delays](#).

## Enable logging with PowerShell

The following example shows how to enable diagnostic logs via the Azure PowerShell Cmdlets.

### NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

### Enabling diagnostic logs in a storage account

1. Log in and select a subscription:

```
Connect-AzAccount
```

```
Select-AzureSubscription -SubscriptionId
```

2. To enable Diagnostic Logs in a Storage account, enter this command:

```
Set-AzDiagnosticSetting -ResourceId  
"/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/providers/Microsoft.Cdn/profiles/{profileName}/endpoints/{endpointName}" -StorageAccountId  
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.ClassicStorage/storageAccounts/{storageAccountName}" -Enabled $true -Categories CoreAnalytics
```

3. To enable diagnostics logs in a Log Analytics workspace, enter this command:

```
Set-AzDiagnosticSetting -ResourceId "/subscriptions/{subscriptionId}<subscriptionId>.{subscriptionName}" -WorkspaceId "/subscriptions<workspaceId>.<workspaceName>" -Enabled $true -Categories CoreAnalytics
```

## Consuming diagnostics logs from Azure Storage

This section describes the schema of CDN core analytics, how it is organized inside of an Azure storage account, and provides sample code to download the logs in a CSV file.

### Using Microsoft Azure Storage Explorer

Before you can access the core analytics data from an Azure storage account, you first need a tool to access the contents in a storage account. While there are several tools available in the market, the one that we recommend is the Microsoft Azure Storage Explorer. To download the tool, see [Azure Storage Explorer](#). After downloading and installing the software, configure it to use the same Azure storage account that was configured as a destination to the CDN Diagnostics Logs.

1. Open **Microsoft Azure Storage Explorer**
2. Locate the storage account
3. Expand the **Blob Containers** node under this storage account.
4. Select the container named *insights-logs-coreanalytics*.
5. Results show up on the right-hand pane, starting with the first level, as *resourceId=*. Continue selecting each level until you find the file *PT1H.json*. For an explanation of the path, see [Blob path format](#).
6. Each blob *PT1H.json* file represents the analytics logs for one hour for a specific CDN endpoint or its custom domain.
7. The schema of the contents of this JSON file is described in the section schema of the core analytics logs.

### Blob path format

Core analytics logs are generated every hour and the data is collected and stored inside a single Azure blob as a JSON payload. Because the Storage explorer tool interprets '/' as a directory separator and shows the hierarchy, the path to the Azure blob appears as if there is a hierarchical structure and represents the blob name. The name of the blob follows the following naming convention:

```
resourceId=/SUBSCRIPTIONS/{Subscription Id}/RESOURCEGROUPS/{Resource Group Name}/PROVIDERS/MICROSOFT.CDN/PROFILES/{Profile Name}/ENDPOINTS/{Endpoint Name}/ y={Year}/m={Month}/d={Day}/h={Hour}/m={Minutes}/PT1H.json
```

### Description of fields:

VALUE	DESCRIPTION
Subscription ID	ID of the Azure subscription in Guid format.
Resource Group Name	Name of the resource group to which the CDN resources belong.
Profile Name	Name of the CDN Profile
Endpoint Name	Name of the CDN Endpoint
Year	Four-digit representation of the year, for example, 2017
Month	Two-digit representation of the month number. 01=January ... 12=December
Day	Two-digit representation of the day of the month
PT1H.json	Actual JSON file where the analytics data is stored

### Exporting the core analytics data to a CSV file

To make it easy to access core analytics, sample code for a tool is provided. This tool allows downloading the JSON files into a flat comma-separated file format, which can be used to create charts or other aggregations.

Here's how you can use the tool:

1. Visit the GitHub link: <https://github.com/Azure-Samples/azure-cdn-samples/tree/master/CoreAnalytics-ExportToCsv>
2. Download the code.
3. Follow the instructions to compile and configure.
4. Run the tool.
5. The resulting CSV file shows the analytics data in a simple flat hierarchy.

## Consuming diagnostics logs from a Log Analytics workspace

Azure Monitor is an Azure service that monitors your cloud and on-premises environments to maintain their availability and performance. It collects data generated by resources in your cloud and on-premises environments and from other monitoring tools to provide analysis across multiple sources.

To use Azure Monitor, you must [enable logging](#) to the Azure Log Analytics workspace, which is discussed earlier in this article.

### Using the Log Analytics workspace

The following diagram shows the architecture of the inputs and outputs of the repository:

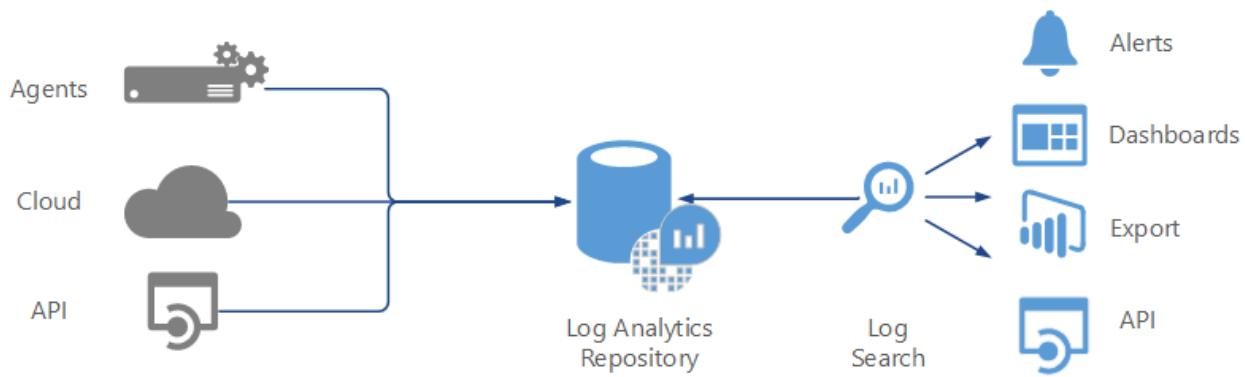


Figure 3 - Log Analytics Repository

You can display the data in a variety of ways by using Management Solutions. You can obtain Management Solutions from the [Azure Marketplace](#).

You can install monitoring solutions from Azure marketplace by selecting the **Get it now** link at the bottom of each solution.

### Add an Azure Monitor CDN monitoring solution

Follow these steps to add an Azure Monitor monitoring solution:

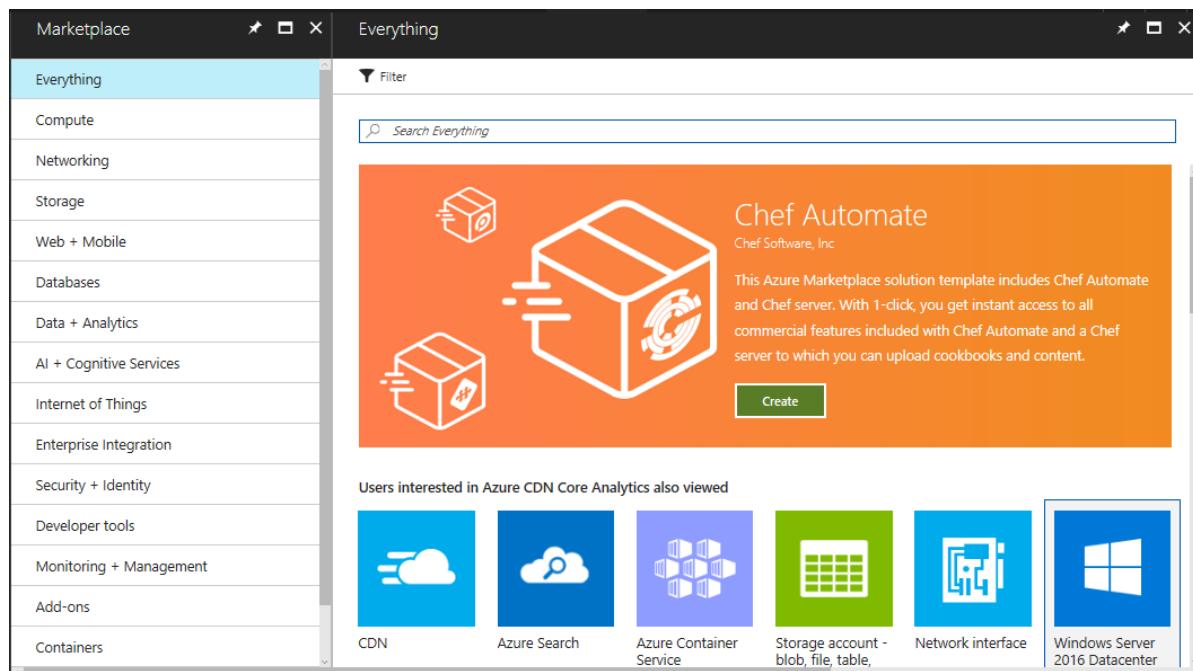
1. Sign in to the Azure portal using your Azure subscription and go to your dashboard.

All resources ALL SUBSCRIPTIONS	
anyname	Log Analytics
OMS-Workspace-040754	Log Analytics
rintest1	Log Analytics
CdnCoreAnalyticsSolution[rintest1]	Solution
CdnCoreAnalyticsSolution[anyname]	Solution
Azurefriday	CDN profile
ahmedh-ec-outage-prem	CDN profile
272522canadacentral	Storage account
<a href="#">See more...</a>	
Service Health	Marketplace

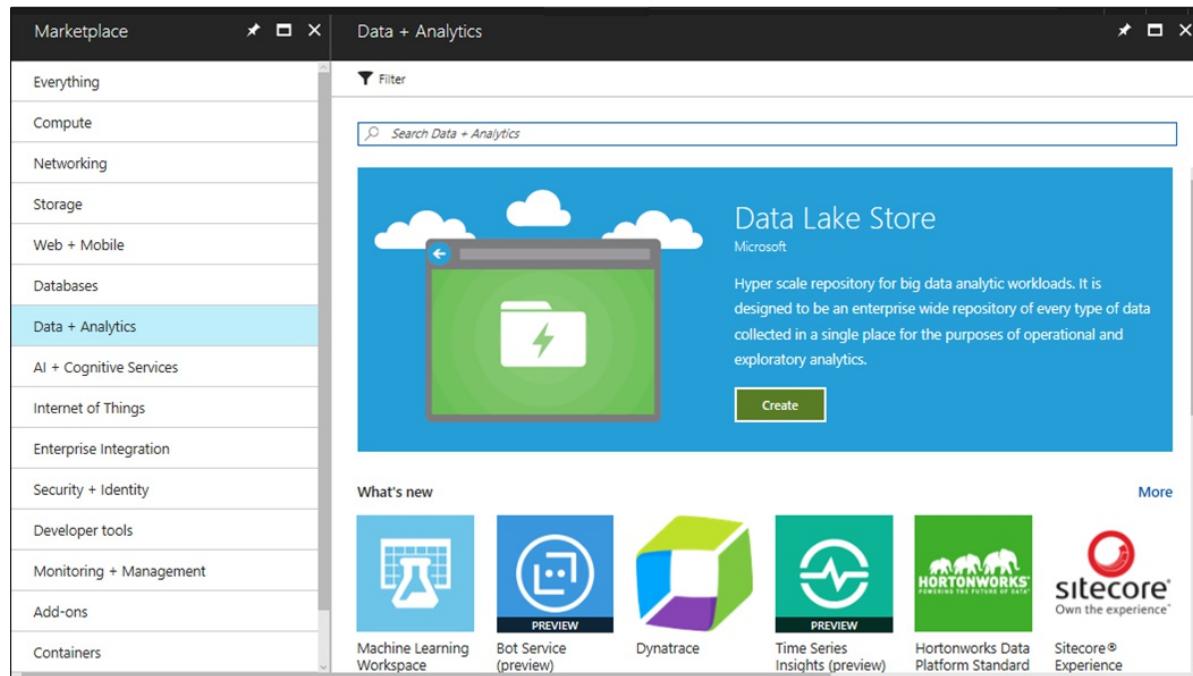
**Quickstart tutorials**

- [Windows Virtual Machines](#) Provision Windows Server, SQL Server, SharePoint VMs
- [Linux Virtual Machines](#) Provision Ubuntu, Red Hat, CentOS, SUSE, CoreOS VMs
- [App Service](#) Create Web Apps using .NET, Java, Node.js, Python, PHP
- [Functions](#) Process events with a serverless code architecture
- [SQL Database](#) Managed relational SQL Database as a Service

2. In the **New** page, under **Marketplace**, select **Monitoring + management**.



3. In the **Monitoring + management** page, select **See all**.



4. Search for CDN in the search box.

Marketplace

Data + Analytics

Filter

NAME

PUBLISHER

CATEGORY

NAME	PUBLISHER	CATEGORY
Azure CDN Core Analytics	Microsoft	Data analytics

5. Select **Azure CDN Core Analytics**.

Azure CDN Core Analytics

Usage:

- **Data transferred in GB:** View total traffic usage over a particular time period.

Errors:

- **Error summary:** Summarizes 4xx, 5xx error counts delivered by CDN.
- **Error distribution:** Shows percentage of 4xx, 5xx errors out of total hits delivered by CDN.

Hits:

- **Hit summary:** summarizes total hit counts over a period of time
- **Hit distribution:** Shows distribution (percentage) based on response codes: 2xx, 3xx, 4xx, 5xx, and other response codes.

Cache efficiency:

- **Cached traffic ratio:** shows usage that's delivered directly from CDN edge POPs over a period of time.

More info

USAGE

Data transferred

TOTAL PER HOUR

4.91

12:00 PM 6:00 PM 12:00 AM 6:00 AM

Create

6. After you select **Create**, you are asked to create a new Log Analytics workspace or use an existing one.

The screenshot shows two windows side-by-side. On the left is the 'Add Solution' dialog, which has a 'Workspace' field containing 'anyname' and a 'Automation account' field containing 'DefaultWorkspace-27cafca8...'. A note at the bottom says: 'Note: It might take up to several minutes to create the solution. Please check portal notifications for the progress.' Below the fields are 'Pin to dashboard' and 'Create' buttons. On the right is the 'OMS Workspaces' list, showing a table with columns for icon, workspace name, and location. The list includes:

	Workspace Name	Location
	Create New Workspace	
	anyname westcentralus	
	CdnCoreAnalyticsWorkspace eastus	
	DefaultWorkspace-27cafca8... japaneast	
	DefaultWorkspace-27cafca8... eastus	
	DefaultWorkspace-27cafca8... southeastasia	
	OMS-Workspace-040754 australiasoutheast	
	rintest1 eastus	

7. Select the workspace you created before. You then need to add an automation account.

The screenshot shows two windows side-by-side. On the left is the 'Add Solution' dialog, which now has 'OMS-Workspace-040754' selected in the 'Workspace' field. The 'Automation account' field is still empty. A note at the bottom says: 'Note: It might take up to several minutes to create the solution. Please check portal notifications for the progress.' Below the fields are 'Pin to dashboard' and 'Create' buttons. On the right is the 'Automation Account' selection dialog, which displays a message: 'Choose an Automation account.' Below it is a 'Create an Automation account' button and a note: 'No Automation accounts found. You may creat...'.

8. The following screen shows the automation account form you must fill out.

The image shows three windows side-by-side:

- Add Solution**: Shows a workspace named "OMS-Workspace-040754" and an automation account named "Automation-040754".
- Automation Account**: A modal window titled "Choose an Automation account." It has a "Create an Automation account" button and a note: "No Automation accounts found. You may creat...".
- Add Automation Account**: A modal window for creating a new Automation account. It includes fields for "Resource group" (set to "AFD"), "Location" (set to "Australia Southeast"), and "Create Azure Run As account" (radio button set to "Use existing"). There are "Yes" and "No" buttons at the bottom.

9. Once you have created the automation account, you are ready to add your solution. Select the **Create** button.

The "Add Solution" window now shows the "Automation-040754" item selected, indicated by a blue dashed border around its row.

Note: It might take up to several minutes to create the solution. Please check portal notifications for the progress.

At the bottom, there is a "Pin to dashboard" checkbox (checked) and a prominent "Create" button.

10. Your solution has now been added to your workspace. Return to your Azure portal dashboard.

The screenshot shows the Azure Dashboard. On the left, under 'All resources ALL SUBSCRIPTIONS', there is a list of resources including 'anyname' (Log Analytics), 'rintest1' (Log Analytics), 'OMS-Workspace-040754' (Log Analytics), 'CdnCoreAnalyticsSolution[rintest1]' (Solution), 'CdnCoreAnalyticsSolution[anyname]' (Solution), 'Azurefriday' (CDN profile), 'ahmedh-ec-outage-prem' (CDN profile), and '272522canadacentral' (Storage account). A 'See more...' link is also present. Below this list are two buttons: 'Service Health' and 'Marketplace'. On the right, under 'Quickstart tutorials', there are five entries: 'Windows Virtual Machines' (Provision Windows Server, SQL Server, SharePoint VMs), 'Linux Virtual Machines' (Provision Ubuntu, Red Hat, CentOS, SUSE, CoreOS VMs), 'App Service' (Create Web Apps using .NET, Java, Node.js, Python, PHP), 'Functions' (Process events with a serverless code architecture), and 'SQL Database' (Managed relational SQL Database as a Service).

Select the Log Analytics workspace you created to go to your workspace.

11. Select the **OMS Portal** tile to see your new solution.

The screenshot shows the Azure OMS Portal dashboard for the workspace 'demo-oms-workspace'. The left sidebar includes links for 'Search (Ctrl+ /)', 'OMS Workspace' (selected), 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Locks', 'Automation script', 'Advanced settings', 'Quick Start', and 'Upgrade Summary Log'. The main content area displays workspace details: Resource group (cdnemorg), Status (Active), Location (East US), Subscription name (<subscription name>), Subscription ID (<subscription ID>), Workspace Name (demo-oms-workplace), Workspace ID (<workspace ID>), Pricing tier (Free), and Management services (Operations logs). Below this, the 'Management' section features four tiles: 'Overview', 'Log Search', 'OMS Portal' (which is highlighted with a blue dashed box), and 'View Designer'.

12. Your portal should now look like the following screen:

The screenshot shows the Microsoft Operations Management Suite dashboard. On the left, there's a vertical navigation bar with icons for Log Search, My Dashboard, Solutions Gallery, Usage, and Settings. The main area has several tiles: one for 'Settings' showing 0 data sources connected; another for 'Azure CDN Core Analytics' with a chart showing 'TOTAL GB TRANSFERRED PER HOUR' at 0.0049; a third for 'HTTP Status Code Distribution'; and a fourth for 'Azure CDN Core Analytics Copy' showing a pie chart with 44 total hits (24 Usage, 20 AzureDiagnostics). A 'View Designer' tile on the right allows users to create personalized views.

Select one of the tiles to see several views into your data.

This screenshot shows the 'AzureCdnCoreAnalytics' view. It includes a timeline from 8 PM to 12 AM, a bar chart of 'PERCENTAGE OF ERRORS' (mostly 0), and a bar chart of 'HITS' for specific times. Below these are tables for 'DOMAIN NAME' (maningtesttm.azureedge.net) and 'NUMBER OF HITS' (10.1K), and 'RESPONSE CODE' distribution. To the right, there's a 'CACHE EFFICIENCY' section with a chart of 'Cached traffic ratio' (0.0) over time, and a 'LIST OF SUGGESTED QUERIES' on the far right.

You can scroll left or right to see further tiles representing individual views into the data.

Select one of the tiles to see more details about your data.

This screenshot shows the 'Log Search' view. It features a histogram of 'Data based on custom time range' (1 bar = 1hr) from Jul 25 to Jul 26, 2017. Below it are three search filters: 'TYPE (1)' (AzureDiagnostics), 'OPERATIONNAME (1)' (Microsoft.Cdn/profiles/endpoints/contentDelivery), and 'CATEGORY (1)' (CoreAnalytics). To the right is a chart titled 'Type=AzureDiagnostics OperationName="Microsoft.Cdn/profiles/endpoints/contentDelivery" Category=CoreAnalytics | measure sum(EgressTotal\_d) as TotalGB interval 1h' showing 'TotalGB' over time (Jul 26 12:00 AM to Jul 26 6:00 PM).

## Offers and pricing tiers

You can see offers and pricing tiers for management solutions [here](#).

## Customizing views

You can customize the view into your data by using the **View Designer**. To begin designing, go to your Log Analytics workspace and select the **View Designer** tile.

Drag-and-drop the types of charts and fill in the data details you want to analyze.

## Log data delays

The following table shows log data delays for **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Akamai**, and **Azure CDN Standard/Premium from Verizon**.

MICROSOFT LOG DATA DELAYS	VERIZON LOG DATA DELAYS	AKAMAI LOG DATA DELAYS
Delayed by 1 hour.	Delayed by 1 hour and can take up to 2 hours to start appearing after endpoint propagation completion.	Delayed by 24 hours; if it was created more than 24 hours ago, it takes up to 2 hours to start appearing. If it was recently created, it can take up to 25 hours for the logs to start appearing.

## Diagnostic log types for CDN core analytics

Microsoft currently offers core analytics logs only, which contain metrics showing HTTP response statistics and egress statistics as seen from the CDN POPs/edges.

### Core analytics metrics details

The following table shows a list of metrics available in the core analytics logs for **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Akamai**, and **Azure CDN Standard/Premium from Verizon**. Not all metrics are available from all providers, although such differences are minimal. The table also displays whether a given metric is available from a provider. The metrics are available for only those CDN endpoints that have traffic on them.

METRIC	DESCRIPTION	MICROSOFT	VERIZON	AKAMAI
RequestCountTotal	Total number of request hits during this period.	Yes	Yes	Yes
RequestCountHttpStatus2xx	Count of all requests that resulted in a 2xx HTTP code (for example, 200, 202).	Yes	Yes	Yes
RequestCountHttpStatus3xx	Count of all requests that resulted in a 3xx HTTP code (for example, 300, 302).	Yes	Yes	Yes
RequestCountHttpStatus4xx	Count of all requests that resulted in a 4xx HTTP code (for example, 400, 404).	Yes	Yes	Yes
RequestCountHttpStatus5xx	Count of all requests that resulted in a 5xx HTTP code (for example, 500, 504).	Yes	Yes	Yes
RequestCountHttpStatusOthers	Count of all other HTTP codes (outside of 2xx-5xx).	Yes	Yes	Yes
RequestCountHttpStatus200	Count of all requests that resulted in a 200 HTTP code response.	Yes	No	Yes

METRIC	DESCRIPTION	MICROSOFT	VERIZON	AKAMAI
RequestCountHttpStatus206	Count of all requests that resulted in a 206 HTTP code response.	Yes	No	Yes
RequestCountHttpStatus302	Count of all requests that resulted in a 302 HTTP code response.	Yes	No	Yes
RequestCountHttpStatus304	Count of all requests that resulted in a 304 HTTP code response.	Yes	No	Yes
RequestCountHttpStatus404	Count of all requests that resulted in a 404 HTTP code response.	Yes	No	Yes
RequestCountCacheHit	Count of all requests that resulted in a Cache hit. The asset was served directly from the POP to the client.	Yes	Yes	No
RequestCountCacheMiss	Count of all requests that resulted in a Cache miss. A Cache miss means the asset was not found on the POP closest to the client, and therefore was retrieved from the Origin.	Yes	Yes	No
RequestCountCacheNoCache	Count of all requests to an asset that are prevented from being cached due to a user configuration on the edge.	Yes	Yes	No
RequestCountCacheUncacheable	Count of all requests to assets that are prevented from being cached by the asset's Cache-Control and Expires headers, which indicate that it should not be cached on a POP or by the HTTP client.	Yes	Yes	No
RequestCountCacheOthers	Count of all requests with cache status not covered by above.	No	Yes	No
EgressTotal	Outbound data transfer in GB	Yes	Yes	Yes

METRIC	DESCRIPTION	MICROSOFT	VERIZON	AKAMAI
EgressHttpStatus2xx	Outbound data transfer* for responses with 2xx HTTP status codes in GB.	Yes	Yes	No
EgressHttpStatus3xx	Outbound data transfer for responses with 3xx HTTP status codes in GB.	Yes	Yes	No
EgressHttpStatus4xx	Outbound data transfer for responses with 4xx HTTP status codes in GB.	Yes	Yes	No
EgressHttpStatus5xx	Outbound data transfer for responses with 5xx HTTP status codes in GB.	Yes	Yes	No
EgressHttpStatusOthers	Outbound data transfer for responses with other HTTP status codes in GB.	Yes	Yes	No
EgressCacheHit	Outbound data transfer for responses that were delivered directly from the CDN cache on the CDN POPs/Edges.	Yes	Yes	No
EgressCacheMiss.	Outbound data transfer for responses that were not found on the nearest POP server, and retrieved from the origin server.	Yes	Yes	No
EgressCacheNoCache	Outbound data transfer for assets that are prevented from being cached due to a user configuration on the edge.	Yes	Yes	No

METRIC	DESCRIPTION	MICROSOFT	VERIZON	AKAMAI
EgressCacheUncacheable	Outbound data transfer for assets that are prevented from being cached by the asset's Cache-Control and/or Expires headers. Indicates that it should not be cached on a POP or by the HTTP client.	Yes	Yes	No
EgressCacheOthers	Outbound data transfers for other cache scenarios.	No	Yes	No

\*Outbound data transfer refers to traffic delivered from CDN POP servers to the client.

### Schema of the core analytics logs

All logs are stored in JSON format and each entry has string fields according to the following schema:

```

"records": [
    {
        "time": "2017-04-27T01:00:00",
        "resourceId": "<ARM Resource Id of the CDN Endpoint>",
        "operationName": "Microsoft.Cdn/profiles/endpoints/contentDelivery",
        "category": "CoreAnalytics",
        "properties": {
            "DomainName": "<Name of the domain for which the statistics is reported>",
            "RequestCountTotal": integer value,
            "RequestCountHttpStatus2xx": integer value,
            "RequestCountHttpStatus3xx": integer value,
            "RequestCountHttpStatus4xx": integer value,
            "RequestCountHttpStatus5xx": integer value,
            "RequestCountHttpStatusOthers": integer value,
            "RequestCountHttpStatus200": integer value,
            "RequestCountHttpStatus206": integer value,
            "RequestCountHttpStatus302": integer value,
            "RequestCountHttpStatus304": integer value,
            "RequestCountHttpStatus404": integer value,
            "RequestCountCacheHit": integer value,
            "RequestCountCacheMiss": integer value,
            "RequestCountCacheNoCache": integer value,
            "RequestCountCacheUncacheable": integer value,
            "RequestCountCacheOthers": integer value,
            "EgressTotal": double value,
            "EgressHttpStatus2xx": double value,
            "EgressHttpStatus3xx": double value,
            "EgressHttpStatus4xx": double value,
            "EgressHttpStatus5xx": double value,
            "EgressHttpStatusOthers": double value,
            "EgressCacheHit": double value,
            "EgressCacheMiss": double value,
            "EgressCacheNoCache": double value,
            "EgressCacheUncacheable": double value,
            "EgressCacheOthers": double value,
        }
    }
]
}

```

Where *time* represents the start time of the hour boundary for which the statistics is reported. When a metric is not supported by a CDN provider, instead of a double or integer value, there is a null value. This null value indicates the absence of a metric, and is different from a value of 0. There is one set of these metrics per domain configured on the endpoint.

Example properties:

```
{  
    "DomainName": "manlingakamaitest2.azureedge.net",  
    "RequestCountTotal": 480,  
    "RequestCountHttpStatus2xx": 480,  
    "RequestCountHttpStatus3xx": 0,  
    "RequestCountHttpStatus4xx": 0,  
    "RequestCountHttpStatus5xx": 0,  
    "RequestCountHttpStatusOthers": 0,  
    "RequestCountHttpStatus200": 480,  
    "RequestCountHttpStatus206": 0,  
    "RequestCountHttpStatus302": 0,  
    "RequestCountHttpStatus304": 0,  
    "RequestCountHttpStatus404": 0,  
    "RequestCountCacheHit": null,  
    "RequestCountCacheMiss": null,  
    "RequestCountCacheNoCache": null,  
    "RequestCountCacheUncacheable": null,  
    "RequestCountCacheOthers": null,  
    "EgressTotal": 0.09,  
    "EgressHttpStatus2xx": null,  
    "EgressHttpStatus3xx": null,  
    "EgressHttpStatus4xx": null,  
    "EgressHttpStatus5xx": null,  
    "EgressHttpStatusOthers": null,  
    "EgressCacheHit": null,  
    "EgressCacheMiss": null,  
    "EgressCacheNoCache": null,  
    "EgressCacheUncacheable": null,  
    "EgressCacheOthers": null  
}
```

## Additional resources

- [Azure Diagnostic logs](#)
- [Core analytics via Azure CDN supplemental portal](#)
- [Azure Monitor logs](#)
- [Azure Log Analytics REST API](#)

# Core Reports from Verizon

7/5/2019 • 6 minutes to read • [Edit Online](#)

## IMPORTANT

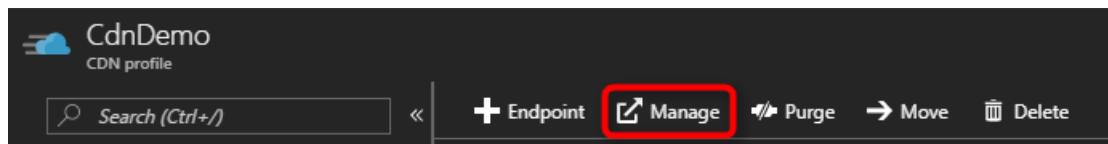
This feature is available only with **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** products. It is not supported on **Azure CDN from Akamai**. For a comparison of CDN features, see [Azure CDN product features](#).

By using Verizon Core Reports via the Manage portal for Verizon profiles, you can view usage patterns for your CDN with the following reports:

- Bandwidth
- Data Transferred
- Hits
- Cache Statuses
- Cache Hit Ratio
- IPV4/IPV6 Data Transferred

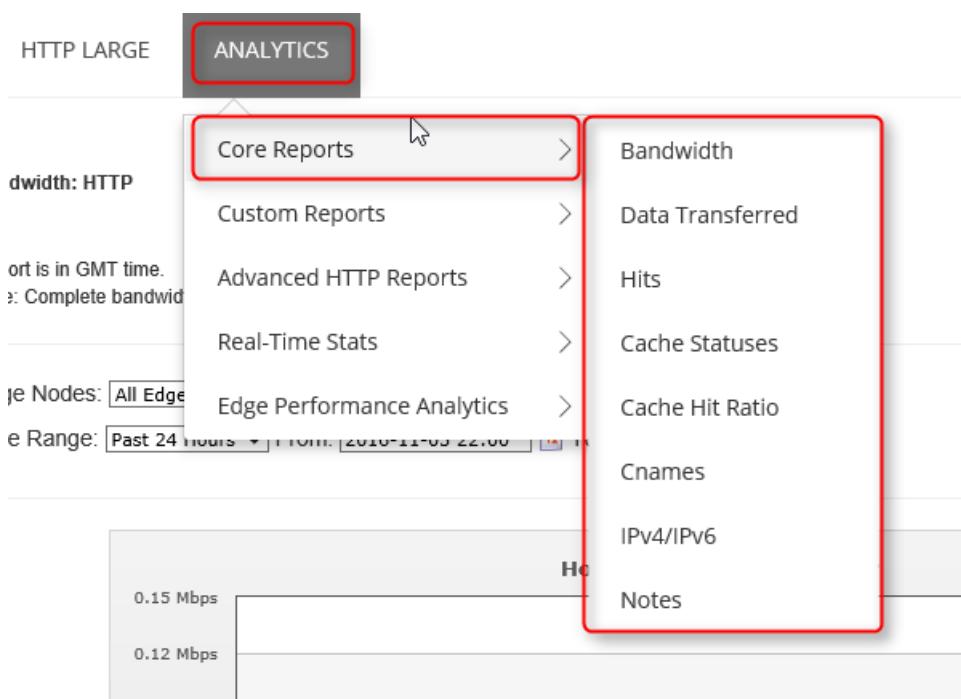
## Accessing Verizon Core Reports

1. From the CDN profile blade, click the **Manage** button.



The CDN management portal opens.

2. Hover over the **Analytics** tab, then hover over the **Core Reports** flyout. Click on a report in the menu.



3. For each report, select a date range from the **Date Range** list. You can either select a pre-defined date

range, such as **Today** or **This Week**, or you can select **Custom** and manually enter a date range by clicking the calendar icons.

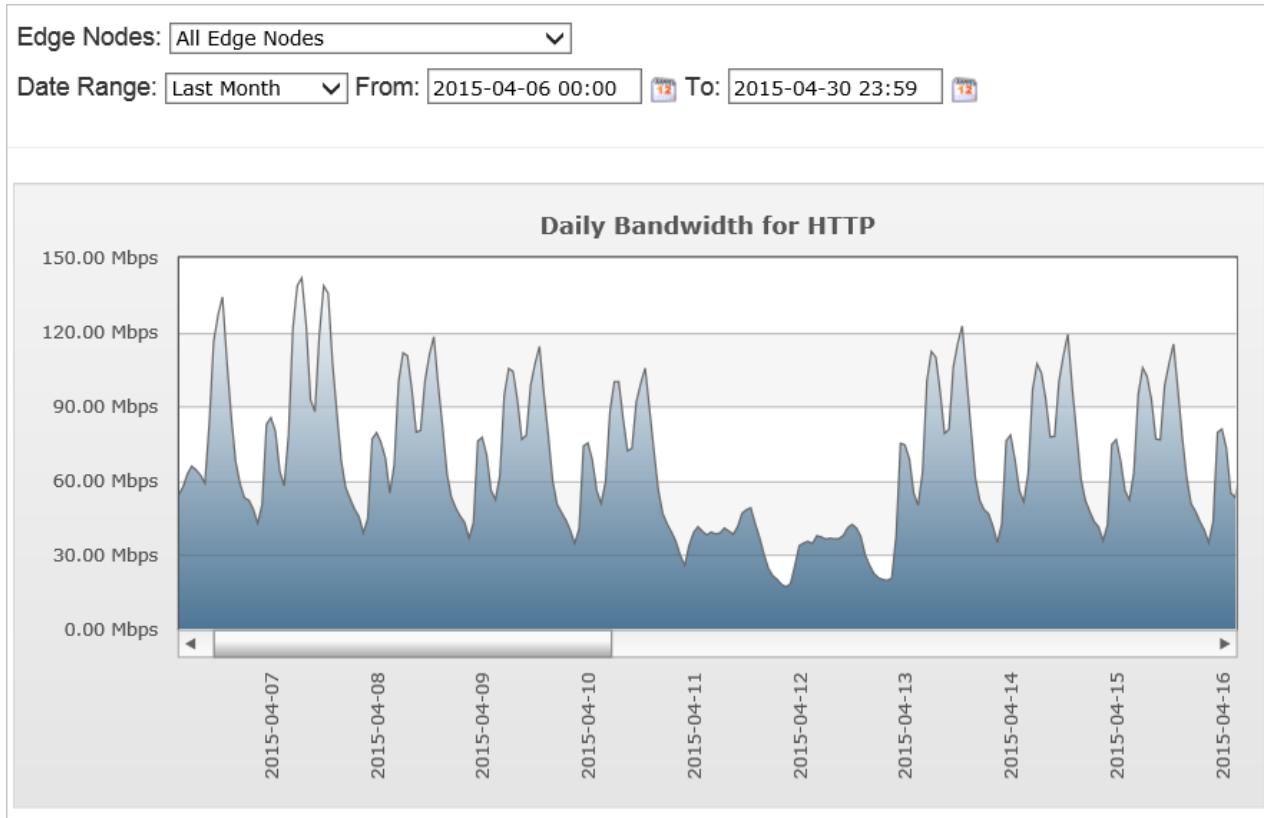
4. After you have selected a date range, click **Go** to generate the report.
5. If you want to export the data in Excel format, click the Excel icon above the **Go** button.

## Bandwidth

The bandwidth report consists of a graph and data table that indicates the CDN bandwidth usage for HTTP and HTTPS over a particular time period, in Mbps. You can view the bandwidth usage across all POPs or for a particular POP. This report allows you to view the traffic spikes and distribution for POPs.

From the **Edge Nodes** list, select **All Edge Nodes** to see traffic from all nodes or select a specific region.

The report is updated every five minutes.



## Data transferred

This report consists of a graph and data table that indicates the CDN traffic usage for HTTP and HTTPS over a particular time period, in GB. You can view the traffic usage across all POPs or for a particular POP. This report allows you to view the traffic spikes and distribution across POPs.

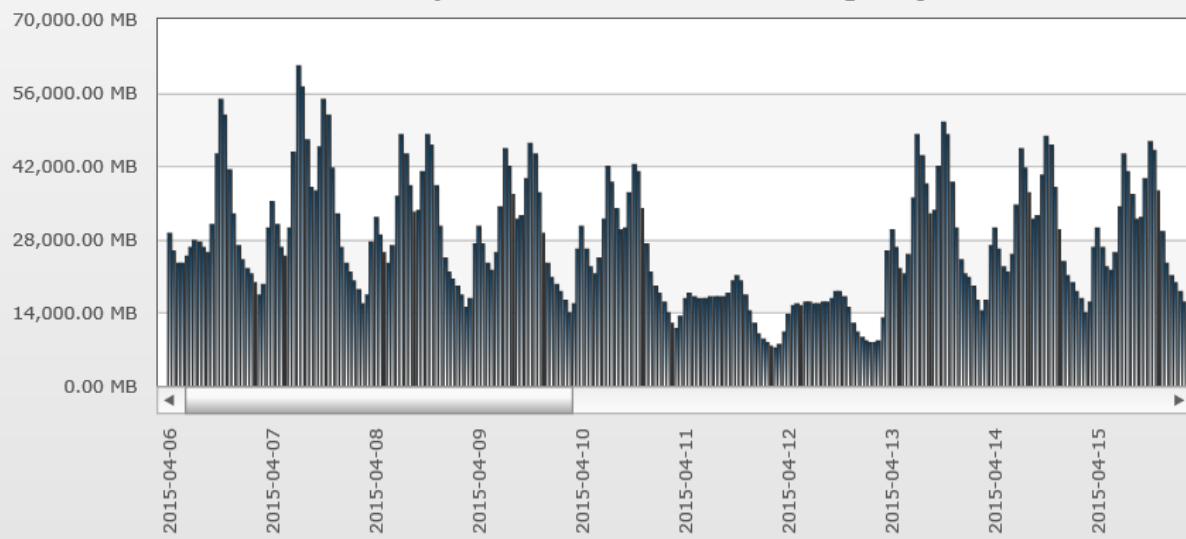
From the **Edge Nodes** list, select **All Edge Nodes** to see traffic from all nodes or select a specific region.

The report is updated every five minutes.

Edge Nodes: All Edge Nodes

Date Range: Custom From: 2015-04-06 00:00 To: 2015-04-30 23:59

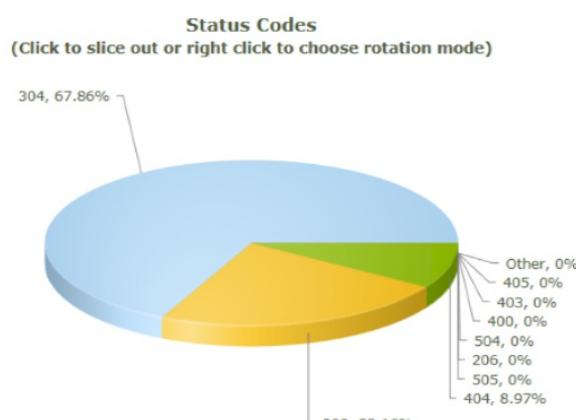
Daily Data Transferred for HTTP Large Object



Total for specified time period: 15,371,907.92 MB / 15,371.91 GB

## Hits (status codes)

This report describes the distribution of request status codes for your content. Every request for content generates an HTTP status code. The status code describes how edge POPs handled the request. For example, a 2xx status code indicates that the request was successfully served to a client, while a 4xx status code indicates that an error occurred. For more information about HTTP status codes, see [List of HTTP status codes](#).



Status Codes: HTTP Large Object

Status Codes	Description	Total Hits	% of Total Hits
304	Not Modified	6553777699	67.86%
200	Ok	2236726432	23.16%

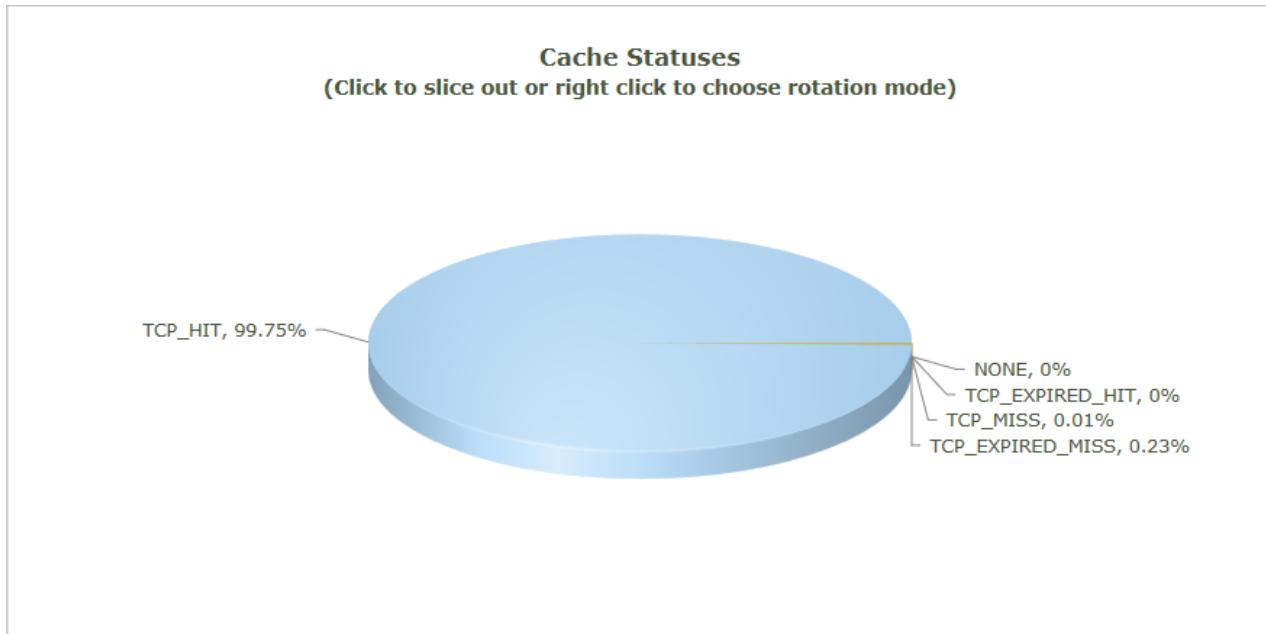
## Cache statuses

This report describes the distribution of cache hits and cache misses for client requests. Because the fastest performance results from cache hits, you can optimize data delivery speeds by minimizing cache misses and expired cache hits.

To reduce cache misses, configure your origin server to minimize the use of the following:

- `no-cache` response headers
- Query-string caching, unless strictly needed
- Non-cacheable response codes

To reduce expired cache hits, set an asset's `max-age` to a long period to minimize the number of requests to the origin server.



### Main cache statuses include:

- TCP\_HIT: Served from edge server. The object was in the cache and has not exceeded its max-age.
- TCP\_MISS: Served from origin server. The object was not in the cache and the response was back to origin.
- TCP\_EXPIRED\_MISS: Served from origin server after revalidation with origin. The object was in the cache, but had exceeded its max-age. A revalidation with origin resulted in the cache object being replaced by a new response from origin.
- TCP\_EXPIRED\_HIT: Served from Edge after revalidation with origin. The object was in cache but had exceeded its max-age. A revalidation with the origin server resulted in the cache object being unmodified.

### Full list of cache statuses

- TCP\_HIT - This status is reported when a request is served directly from the POP to the client. An asset is immediately served from a POP when it is cached on the POP closest to the client and has a valid time-to-live (TTL). TTL is determined by the following response headers:
  - Cache-Control: s-maxage
  - Cache-Control: max-age
  - Expires
- TCP\_MISS: This status indicates that a cached version of the requested asset was not found on the POP closest to the client. The asset is requested from either an origin server or an origin shield server. If the origin server or the origin shield server returns an asset, it is served to the client and cached on both the client and the edge server. Otherwise, a non-200 status code (for example, 403 Forbidden or 404 Not Found) is returned.

- TCP\_EXPIRED\_HIT: This status is reported when a request that targets an asset with an expired TTL was served directly from the POP to the client. For example, when the asset's max-age has expired.

An expired request typically results in a revalidation request to the origin server. For a TCP\_EXPIRED\_HIT status to occur, the origin server must indicate that a newer version of the asset does not exist. This situation typically results in an update of the asset's Cache-Control and Expires headers.

- TCP\_EXPIRED\_MISS: This status is reported when a newer version of an expired cached asset is served from the POP to the client. This status occurs when the TTL for a cached asset is expired (for example, expired max-age) and the origin server returns a newer version of that asset. This new version of the asset is served to the client instead of the cached version. Additionally, it is cached on the edge server and the client.
- CONFIG\_NOCACHE: This status indicates that a customer-specific configuration the edge POP prevented the asset from being cached.
- NONE - This status indicates that a cache content freshness check was not performed.
- TCP\_CLIENT\_REFRESH\_MISS: This status is reported when an HTTP client, such as a browser, forces an edge POP to retrieve a new version of a stale asset from the origin server. By default, the servers prevent an HTTP client from forcing the edge servers to retrieve a new version of the asset from the origin server.
- TCP\_PARTIAL\_HIT: This status is reported when a byte range request results in a hit for a partially cached asset. The requested byte range is immediately served from the POP to the client.
- UNCACHEABLE: This status is reported when an asset's Cache-Control and Expires headers indicate that it should not be cached on a POP or by the HTTP client. These types of requests are served from the origin server.

## Cache Hit Ratio

This report indicates the percentage of cached requests that were served directly from cache.

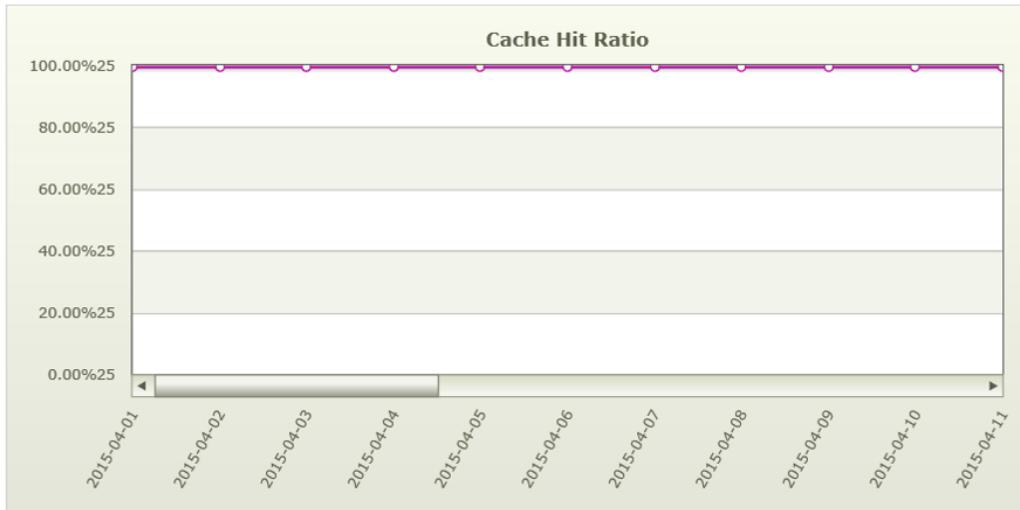
The report provides the following details:

- The requested content was cached on the POP closest to the requester.
- The request was served directly from the edge of our network.
- The request did not require revalidation with the origin server.

The report doesn't include:

- Requests that are denied due to country/region filtering options.
- Requests for assets whose headers indicate that they should not be cached. For example, Cache-Control: private, Cache-Control: no-cache, or Pragma: no-cache headers prevent an asset from being cached.
- Byte range requests for partially cached content.

The formula is:  $(TCP\_HIT / (TCP\_HIT + TCP\_MISS)) * 100$

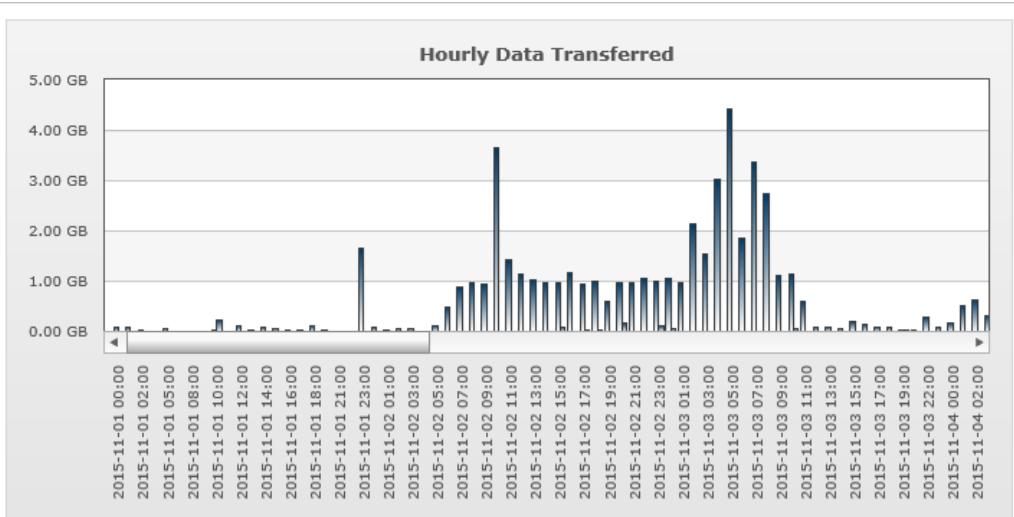


#### Cache Hit Ratio: HTTP Large Object

Date	Hits	Misses	Cache Hit Percentage
2015-04-30	1935989881	2873417	99.85%
2015-04-29	1950331843	3008493	99.85%
2015-04-28	1991619929	2989130	99.85%

## IPv4/IPv6 Data transferred

This report shows the traffic usage distribution in IPv4 vs IPv6.



Total for specified time period: 88.12 GB

#### IPv4/IPv6 Data Transferred

Date	IPv4 (GB)	IPv6 (GB)	Total (GB)
2015-11-01 00:00	0.07 GB	0.00 GB	0.07 GB
2015-11-01 01:00	0.06 GB	0.00 GB	0.06 GB
2015-11-01 02:00	0.02 GB	0.00 GB	0.02 GB

## Considerations

Reports can only be generated within the last 18 months.

# Custom Reports from Verizon

7/5/2019 • 4 minutes to read • [Edit Online](#)

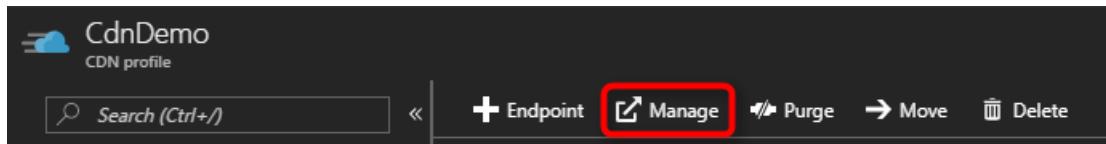
## IMPORTANT

This feature is available only with **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** products. It is not supported on **Azure CDN from Akamai**. For a comparison of CDN features, see [Azure CDN product features](#).

By using Verizon Custom Reports via the Manage portal for Verizon profiles, you can define the type of data to be collected for edge CNAMEs reports.

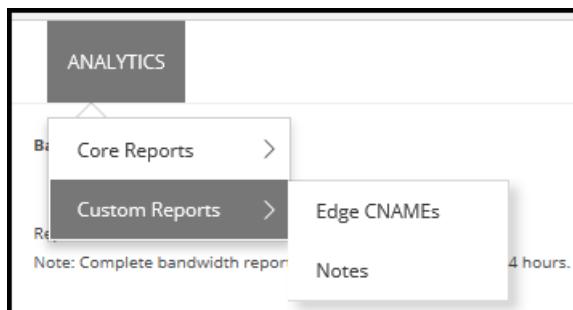
## Accessing Verizon Custom Reports

1. From the CDN profile blade, click the **Manage** button.



The CDN management portal opens.

2. Hover over the **Analytics** tab, then hover over the **Custom Reports** flyout. Click **Edge CNAMEs**.



## Edge CNAMEs custom report

The Edge CNAMEs custom report provides hits and data-transferred statistics for edge CNAMEs on which custom report logging has been enabled. Edge CNAMEs consist of Azure CDN endpoint hostnames and any associated custom domain hostnames.

Custom report data logging begins one hour after you enable an edge CNAME's custom reporting capability. You can view report data by generating an Edge CNAMEs report for a specific platform or for all platforms. The coverage for this report is limited to the edge CNAMEs for which custom report data was collected during the specified time period. The edge CNAMEs report consists of a graph and data table for the top 10 edge CNAMEs according to the metric defined in the Metrics option.

Generate a custom report by defining the following report options:

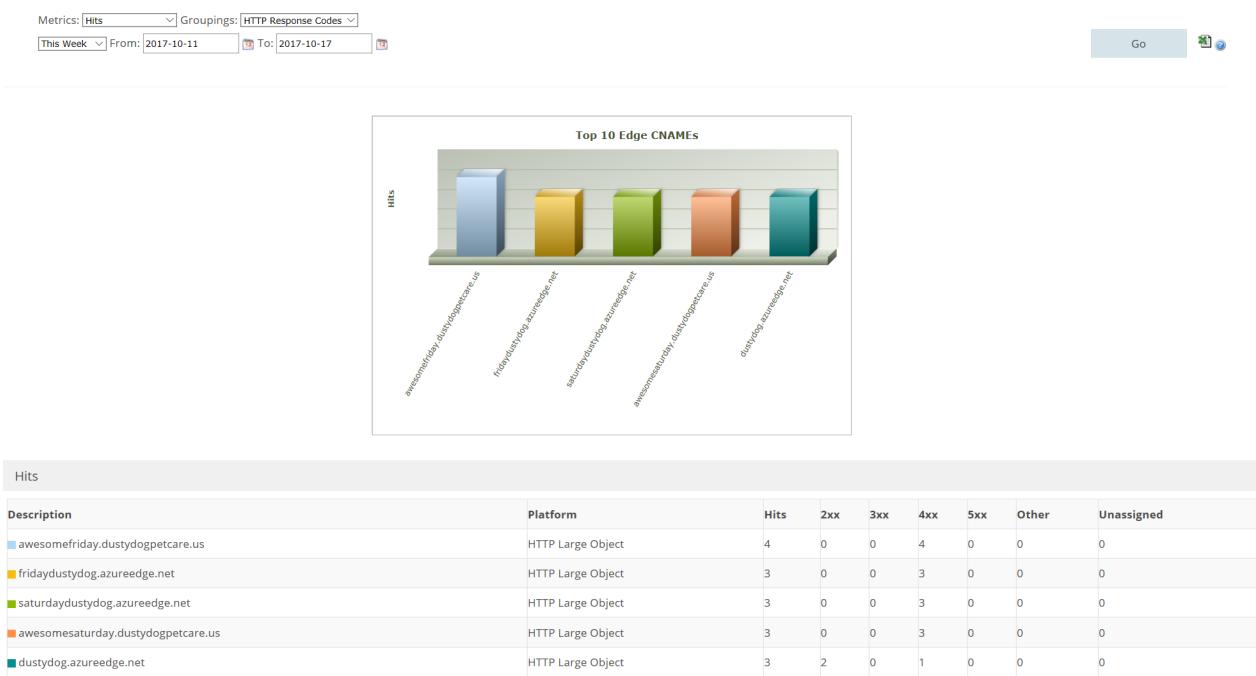
- Metrics: The following options are supported:
  - Hits: Indicates the total number of requests that are directed to an edge CNAME on which the custom reporting capability is enabled. This metric does not include the status code returned to the client.

- Data Transferred: Indicates the total amount of data transferred from the edge servers to the HTTP clients (for example, web browsers) for requests that are directed to an edge CNAME on which the custom reporting capability is enabled. The amount of data transferred is calculated by adding HTTP response headers to the response body. As a result, the amount of data transferred for each asset is greater than its actual file size.
- Groupings: Determines the type of statistics that are shown below the bar chart. The following options are supported:
  - HTTP Response Codes: Organizes statistics by HTTP response code (for example, 200, 403, etc.) returned to the client.
  - Cache Status: Organizes statistics by cache status.

To set the date range for the report, you can either select a pre-defined date range, such as **Today** or **This Week**, from the drop-down list or you can select **Custom** and manually enter a date range by clicking the calendar icons.

After you have selected the date range, click **Go** to generate the report.

You can export the data in Excel format by clicking the Excel symbol to the right of the **Go** button.



## Edge CNAMEs custom report fields

FIELD	DESCRIPTION
2xx	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a 2xx HTTP status code (for example, 200 OK).
3xx	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a 3xx HTTP status code (for example, 302 Found or 304 Not Modified).
4xx	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a 4xx HTTP status code (for example, 400 Bad Request, 403 Forbidden, or 404 Not Found).

FIELD	DESCRIPTION
5xx	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a 5xx HTTP status code (for example, 500 Internal Server Error or 502 Bad Gateway).
Cache Hit %	Indicates the percentage of cacheable requests that were served directly from cache to the requester.
Cache Hits	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a cache hit (for example, TCP_EXPIRED_HIT, TCP_HIT, or TCP_PARTIAL_HIT). A cache hit occurs when a cached version of the requested content is found.
Data Transferred (MB)	Indicates the total amount of data transferred (MB) from the edge servers to HTTP clients (web browsers) for the edge CNAME. The amount of data transferred is calculated by adding the HTTP response headers to the response body. As a result, the amount of data transferred for each asset is greater than its actual file size.
Description	Identifies an edge CNAME by its hostname
Hits	Indicates the total number of requests to the edge CNAME
Misses	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a cache miss (for example, TCP_CLIENT_REFRESH_MISS, TCP_EXPIRED_MISS, or TCP_MISS). A cache miss occurs when the requested content was not cached on the edge server that honored the request.
No Cache	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in a CONFIG_NOCACHE cache status code.
Other	Indicates the total number of requests or data transferred (MB) for the edge CNAME indicated that results in an HTTP status code that falls outside of the 2xx - 5xx range.
Platform	Indicates the platform that handles the edge CNAME's traffic.
Unassigned	Indicates the total number of requests or data transferred (MB) for the edge CNAME for which cache status code or HTTP status code information was not logged.
Uncacheable	Indicates the total number of requests or data transferred (MB) for the edge CNAME that results in an UNCACHEABLE cache status code.

## Considerations

Reports can be generated only within the last 18 months.

# Analyze usage statistics with Azure CDN advanced HTTP reports

7/5/2019 • 15 minutes to read • [Edit Online](#)

## Overview

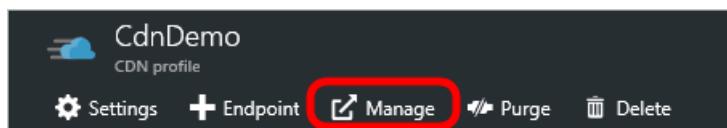
This document explains advanced HTTP reporting in Microsoft Azure CDN. These reports provide detailed information on CDN activity.

### IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

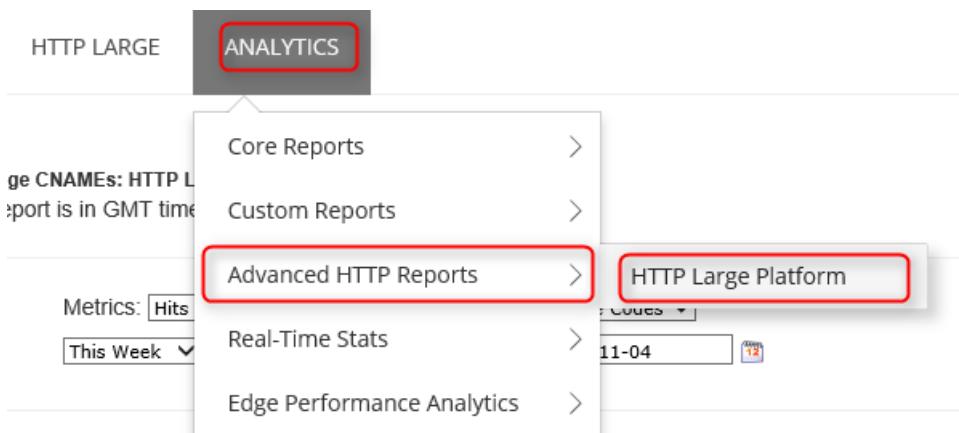
## Accessing advanced HTTP reports

1. From the CDN profile blade, click the **Manage** button.



The CDN management portal opens.

2. Hover over the **Analytics** tab, then hover over the **Advanced HTTP Reports** flyout. Click on **HTTP Large Platform**.



Report options are displayed.

## Geography Reports (Map-Based)

There are five reports that take advantage of a map to indicate the regions from which your content is being requested. These reports are World Map, United States Map, Canada Map, Europe Map, and Asia Pacific Map.

Each map-based report ranks geographic entities (i.e., countries/regions), a map is provided to help you visualize the locations from which your content is being requested. It is able to do so by color-coding each region according

to the amount of demand experienced in that region. Lighter shaded regions indicate lower demand for your content, while darker regions indicate higher levels of demand for your content.

Detailed traffic and bandwidth information for each region is provided directly below the map. This allows you to view the total number of hits, the percentage of hits, the total amount of data transferred (in gigabytes), and the percentage of data transferred for each region. View a description for each of these metrics. Finally, when you hover over a region (i.e., country/region, state, or province), the name and the percentage of hits that occurred in the region will be displayed as a tooltip.

A brief description is provided below for each type of map-based geography report.

REPORT NAME	DESCRIPTION
World Map	This report allows you to view the worldwide demand for your CDN content. Each country/region is color-coded on the world map to indicate the percentage of hits that originated from that region.
United States Map	This report allows you to view the demand for your CDN content in the United States. Each state is color-coded on this map to indicate the percentage of hits that originated from that region.
Canada Map	This report allows you to view the demand for your CDN content in Canada. Each province is color-coded on this map to indicate the percentage of hits that originated from that region.
Europe Map	This report allows you to view the demand for your CDN content in Europe. Each country/region is color-coded on this map to indicate the percentage of hits that originated from that region.
Asia Pacific Map	This report allows you to view the demand for your CDN content in Asia. Each country/region is color-coded on this map to indicate the percentage of hits that originated from that region.

## Geography Reports (Bar Charts)

There are two additional reports that provide statistical information according to geography, which are Top Cities and Top Countries. These reports rank cities and countries/regions, respectively, according to the number of hits that originated from those countries/regions. Upon generating this type of report, a bar chart will indicate the top 10 cities or countries/regions that requested content over a specific platform. This bar chart allows you to quickly assess the regions that generate the highest number of requests for your content.

The left-hand side of the graph (y-axis) indicates how many hits occurred in the specified region. Directly below the graph (x-axis), you will find a label for each of the top 10 regions.

### Using the bar charts

- If you hover over a bar, the name and the total number of hits that occurred in the region will be displayed as a tooltip.
- The tooltip for the Top Cities report identifies a city by its name, state/province, and country/region abbreviation.
- If the city or region (i.e., state/province) from which a request originated could not be determined, then it will indicate that they are unknown. If the country/region is unknown, then two question marks (i.e., ??) will be

displayed.

- A report may include metrics for "Europe" or the "Asia/Pacific Region." Those items are not meant to provide statistical information on all IP addresses in those regions. Rather, they only apply to requests that originate from IP addresses that are spread out over Europe or Asia/Pacific instead of to a specific city or country/region.

The data that was used to generate the bar chart can be viewed below it. There you will find the total number of hits, the percentage of hits, the amount of data transferred (in gigabytes), and the percentage of data transferred for the top 250 regions. View a description for each of these metrics.

A brief description is provided for both types of reports below.

REPORT NAME	DESCRIPTION
Top Cities	This report ranks cities according to the number of hits that originated from that region.
Top Countries	This report ranks countries/regions according to the number of hits that originated from that country/region.

## Daily Summary

The Daily Summary report allows you to view the total number of hits and data transferred over a particular platform on a daily basis. This information can be used to quickly discern CDN activity patterns. For example, this report can help you detect which days experienced higher or lower than expected traffic.

Upon generating this type of report, a bar chart will provide a visual indication as to the amount of platform-specific demand experienced on a daily basis over the time period covered by the report. It will do so by displaying a bar for each day in the report. For example, selecting the time period called "Last Week" will generate a bar chart with seven bars. Each bar will indicate the total number of hits experienced on that day.

The left-hand side of the graph (y-axis) indicates how many hits occurred on the specified date. Directly below the graph (x-axis), you will find a label that indicates the date (Format: YYYY-MM-DD) for each day included in the report.

### TIP

If you hover over a bar, the total number of hits that occurred on that date will be displayed as a tooltip.

The data that was used to generate the bar chart can be viewed below it. There you will find the total number of hits and the amount of data transferred (in gigabytes) for each day covered by the report.

## By Hour

The By Hour report allows you to view the total number of hits and data transferred over a particular platform on an hourly basis. This information can be used to quickly discern CDN activity patterns. For example, this report can help you detect the time periods during the day that experience higher or lower than expected traffic.

Upon generating this type of report, a bar chart will provide a visual indication as to the amount of platform-specific demand experienced on an hourly basis over the time period covered by the report. It will do so by displaying a bar for each hour covered by the report. For example, selecting a 24 hour time period will generate a bar chart with twenty four bars. Each bar will indicate the total number of hits experienced during that hour.

The left-hand side of the graph (y-axis) indicates how many hits occurred on the specified hour. Directly below the graph (x-axis), you will find a label that indicates the date/time (Format: YYYY-MM-DD hh:mm) for each hour

included in the report. Time is reported using 24 hour format and it is specified using the UTC/GMT time zone.

**TIP**

If you hover over a bar, the total number of hits that occurred during that hour will be displayed as a tooltip.

The data that was used to generate the bar chart can be viewed below it. There you will find the total number of hits and the amount of data transferred (in gigabytes) for each hour covered by the report.

## By File

The By File report allows you to view the amount of demand and the traffic incurred over a particular platform for the most requested assets. Upon generating this type of report, a bar chart will be generated on the top 10 most requested assets over the specified time period.

**NOTE**

For the purposes of this report, edge CNAME URLs are converted to their equivalent CDN URLs. This allows an accurate tally for the total number of hits associated with an asset regardless of the CDN or edge CNAME URL used to request it.

The left-hand side of the graph (y-axis) indicates the number of requests for each asset over the specified time period. Directly below the graph (x-axis), you will find a label that indicates the file name for each of the top 10 requested assets.

The data that was used to generate the bar chart can be viewed below it. There you will find the following information for each of the top 250 requested assets: relative path, the total number of hits, the percentage of hits, the amount of data transferred (in gigabytes), and the percentage of data transferred.

## By File Detail

The By File Detail report allows you to view the amount of demand and the traffic incurred over a particular platform for a specific asset. At the very top of this report is the File Details For option. This option provides a list of your most requested assets on the selected platform. In order to generate a By File Detail report, you will need to select the desired asset from the File Details For option. After which, a bar chart will indicate the amount of daily demand that it generated over the specified time period.

The left-hand side of the graph (y-axis) indicates the total number of requests that an asset experienced on a particular day. Directly below the graph (x-axis), you will find a label that indicates the date (Format: YYYY-MM-DD) for which CDN demand for the asset was reported.

The data that was used to generate the bar chart can be viewed below it. There you will find the total number of hits and the amount of data transferred (in gigabytes) for each day covered by the report.

## By File Type

The By File Type report allows you to view the amount of demand and the traffic incurred by file type. Upon generating this type of report, a donut chart will indicate the percentage of hits generated by the top 10 file types.

**TIP**

If you hover over a slice in the donut chart, the Internet media type of that file type will be displayed as a tooltip.

The data that was used to generate the donut chart can be viewed below it. There you will find the file name extension/Internet media type, the total number of hits, the percentage of hits, the amount of data transferred (in

gigabytes), and the percentage of data transferred for each of the top 250 file types.

## By Directory

The By Directory report allows you to view the amount of demand and the traffic incurred over a particular platform for content from a specific directory. Upon generating this type of report, a bar chart will indicate the total number of hits generated by content in the top 10 directories.

### Using the bar chart

- Hover over a bar to view the relative path to the corresponding directory.
- Content stored in a subfolder of a directory does not count when calculating demand by directory. This calculation relies solely on the number of requests generated for content stored in the actual directory.
- For the purposes of this report, edge CNAME URLs are converted to their equivalent CDN URLs. This allows an accurate tally for all statistics associated with an asset regardless of the CDN or edge CNAME URL used to request it.

The left-hand side of the graph (y-axis) indicates the total number of requests for the content stored in your top 10 directories. Each bar on the chart represents a directory. Use the color-coding scheme to match up a bar to a directory listed in the Top 250 Full Directories section.

The data that was used to generate the bar chart can be viewed below it. There you will find the following information for each of the top 250 directories: relative path, the total number of hits, the percentage of hits, the amount of data transferred (in gigabytes), and the percentage of data transferred.

## By Browser

The By Browser report allows you to view which browsers were used to request content. Upon generating this type of report, a pie chart will indicate the percentage of requests handled by the top 10 browsers.

### Using the pie chart

- Hover over a slice in the pie chart to view a browser's name and version.
- For the purposes of this report, each unique browser/version combination is considered a different browser.
- The slice called "Other" indicates the percentage of requests handled by all other browsers and versions.

The data that was used to generate the pie chart can be viewed below it. There you will find the browser type/version number, the total number of hits and the percentage of hits for each of the top 250 browsers.

## By Referrer

The By Referrer report allows you to view the top referrers to content on the selected platform. A referrer indicates the hostname from which a request was generated. Upon generating this type of report, a bar chart will indicate the amount of demand (i.e., hits) generated by the top 10 referrers.

The left-hand side of the graph (y-axis) indicates the total number of requests that an asset experienced for each referrer. Each bar on the chart represents a referrer. Use the color-coding scheme to match up a bar to a referrer listed in the Top 250 Referrer section.

The data that was used to generate the bar chart can be viewed below it. There you will find the URL, the total number of hits, and the percentage of hits generated from each of the top 250 referrers.

## By Download

The By Download report allows you to analyze download patterns for your most requested content. The top of the report contains a bar chart that compares attempted downloads with completed downloads for the top 10 requested assets. Each bar is color-coded according to whether it is an attempted download (blue) or a completed

download (green).

#### NOTE

For the purposes of this report, edge CNAME URLs are converted to their equivalent CDN URLs. This allows an accurate tally for all statistics associated with an asset regardless of the CDN or edge CNAME URL used to request it.

The left-hand side of the graph (y-axis) indicates the file name for each of the top 10 requested assets. Directly below the graph (x-axis), you will find labels that indicate the total number of attempted/completed downloads.

Directly below the bar chart, the following information will be listed for the top 250 requested assets: relative path (including file name), the number of times that it was downloaded to completion, the number of times that it was requested, and the percentage of requests that resulted in a complete download.

#### TIP

Our CDN is not informed by an HTTP client (i.e. browser) when an asset has been completely downloaded. As a result, we have to calculate whether an asset has been completely downloaded according to status codes and byte-range requests. The first thing we look for when making this calculation is whether the request results in a 200 OK status code. If so, then we look at byte-range requests to ensure that they cover the entire asset. Finally, we compare the amount of data transferred to the size of the requested asset. If the data transferred is equal to or greater than the file size and the byte-range requests are appropriate for that asset, then the hit will be counted as a complete download.

Due to the interpretive nature of this report, you should keep in mind the following points that may alter the consistency and accuracy of this report.

- Traffic patterns cannot be accurately predicted when user-agents behave differently. This may produce completed download results that are greater than 100%.
- Assets that take advantage of HTTP Progressive Download may not be accurately represented by this report. This is due to users seeking to different positions in a video.

## By 404 Errors

The By 404 Errors report allows you to identify the type of content that generates the most number of 404 Not Found status codes. The top of the report contains a bar chart for the top 10 assets for which a 404 Not Found status code was returned. This bar chart compares the total number of requests with requests that resulted in a 404 Not Found status code for those assets. Each bar is color-coded. A yellow bar is used to indicate that the request resulted in a 404 Not Found status code. A red bar is used to indicate the total number of requests for the asset.

#### NOTE

For the purposes of this report, note the following:

- A hit represents any request for an asset regardless of status code.
- Edge CNAME URLs are converted to their equivalent CDN URLs. This allows an accurate tally for all statistics associated with an asset regardless of the CDN or edge CNAME URL used to request it.

The left-hand side of the graph (y-axis) indicates the file name for each of the top 10 requested assets that resulted in a 404 Not Found status code. Directly below the graph (x-axis), you will find labels that indicate the total number of requests and the number of requests that resulted in a 404 Not Found status code.

Directly below the bar chart, the following information will be listed for the top 250 requested assets: relative path (including file name), the number of requests that resulted in a 404 Not Found status code, the total number of times that the asset was requested, and the percentage of requests that resulted in a 404 Not Found status code.

## See also

- [Azure CDN Overview](#)
- [Real-time stats in Microsoft Azure CDN](#)
- [Overriding default HTTP behavior using the rules engine](#)
- [Analyze Edge Performance](#)

# Real-time stats in Microsoft Azure CDN

7/5/2019 • 3 minutes to read • [Edit Online](#)

## IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

## Overview

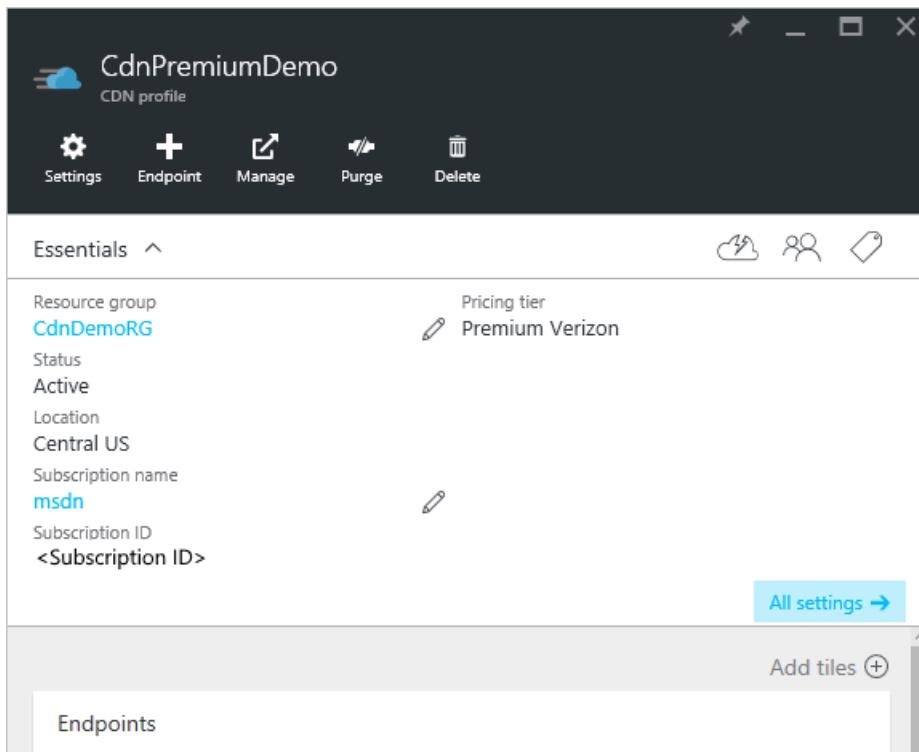
This document explains real-time stats in Microsoft Azure CDN. This functionality provides real-time data, such as bandwidth, cache statuses, and concurrent connections to your CDN profile when delivering content to your clients. This enables continuous monitoring of the health of your service at any time, including go-live events.

The following graphs are available:

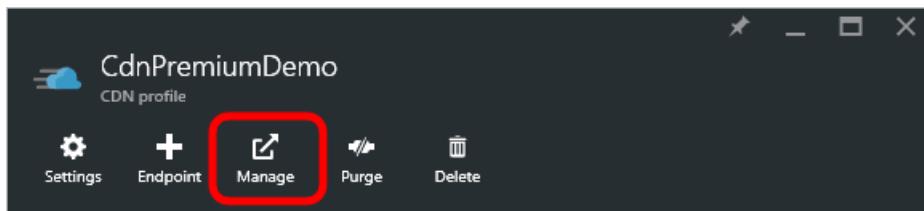
- [Bandwidth](#)
- [Status Codes](#)
- [Cache Statuses](#)
- [Connections](#)

## Accessing real-time stats

1. In the [Azure Portal](#), browse to your CDN profile.



2. From the CDN profile blade, click the **Manage** button.



The CDN management portal opens.

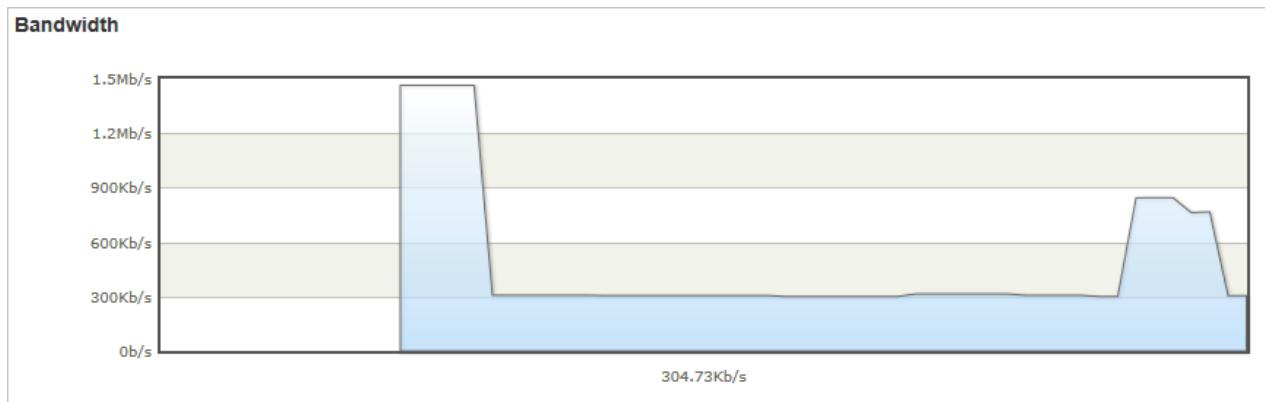
3. Hover over the **Analytics** tab, then hover over the **Real-Time Stats** flyout. Click on **HTTP Large Object**.

A screenshot of the Microsoft Azure Analytics interface. At the top, there are two tabs: 'HTTP LARGE' and 'ANALYTICS' (which has a red box around it). Below the tabs is a 'Country Filtering' section. Under the 'ANALYTICS' tab, there's a flyout menu with several options: 'Core Reports', 'Custom Reports', 'Advanced HTTP Reports', 'Real-Time Stats' (which has a red box around it), 'Edge Performance Analytics', and 'Real-Time Alerts'. A hand cursor icon is pointing at the 'HTTP Large Object' button next to 'Real-Time Stats'.

The real-time stats graphs are displayed.

Each of the graphs displays real-time statistics for the selected time span, starting when the page loads. The graphs update automatically every few seconds. The **Refresh Graph** button, if present, will clear the graph, after which it will only display the selected data.

## Bandwidth



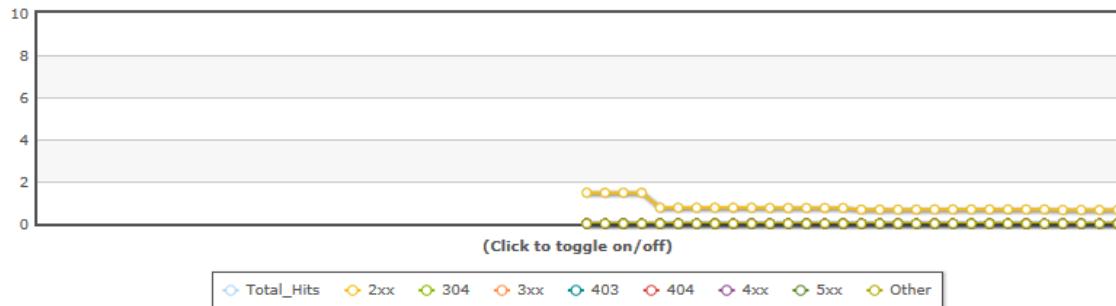
The **Bandwidth** graph displays the amount of bandwidth used for the current platform over the selected time span. The shaded portion of the graph indicates bandwidth usage. The exact amount of bandwidth currently being used is displayed directly below the line graph.

## Status Codes

## Status Codes

Select and refresh to re-scale graph: [Refresh Graph](#)

- Total Hits per second - **1**
- 3xx per second - **0**
- 4xx per second - **0**
- 2xx per second - **1**
- 403 per second - **0**
- 5xx per second - **0**
- 304 per second - **0**
- 404 per second - **0**
- Other per second - **0**



The **Status Codes** graph indicates how often certain HTTP response codes are occurring over the selected time span.

### TIP

For a description of each HTTP status code option, see [Azure CDN HTTP Status Codes](#).

A list of HTTP status codes is displayed directly above the graph. This list indicates each status code that can be included in the line graph and the current number of occurrences per second for that status code. By default, a line is displayed for each of these status codes in the graph. However, you can choose to only monitor the status codes that have special significance for your CDN configuration. To do this, check the desired status codes and clear all other options, then click **Refresh Graph**.

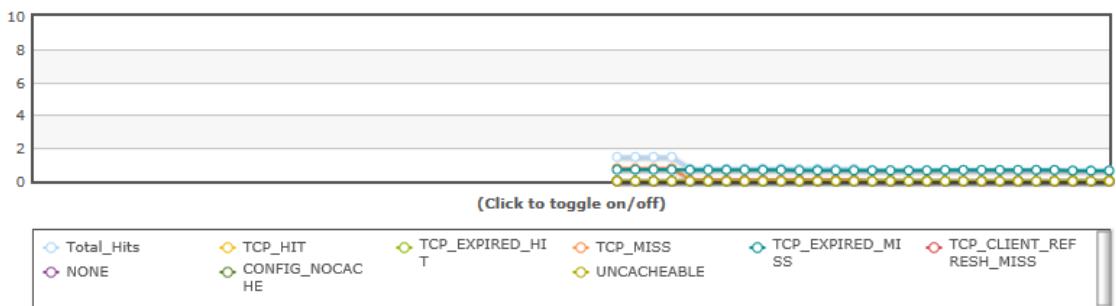
You can temporarily hide logged data for a particular status code. From the legend directly below the graph, click the status code you want to hide. The status code will be immediately hidden from the graph. Clicking that status code again will cause that option to be displayed again.

## Cache Statuses

### Cache Statuses

Select and refresh to re-scale graph: [Refresh Graph](#)

- Total Hits per second - **1**
- TCP\_MISS per second - **0**
- NONE per second - **0**
- TCP\_HIT per second - **0**
- TCP\_EXPIRED\_MISS per second - **1**
- CONFIG\_NOCACHE per second - **0**
- TCP\_EXPIRED\_HIT per second - **0**
- TCP\_CLIENT\_REFRESH\_MISS per second - **0**
- UNCACHEABLE per second - **0**



The **Cache Statuses** graph indicates how often certain types of cache statuses are occurring over the selected time span.

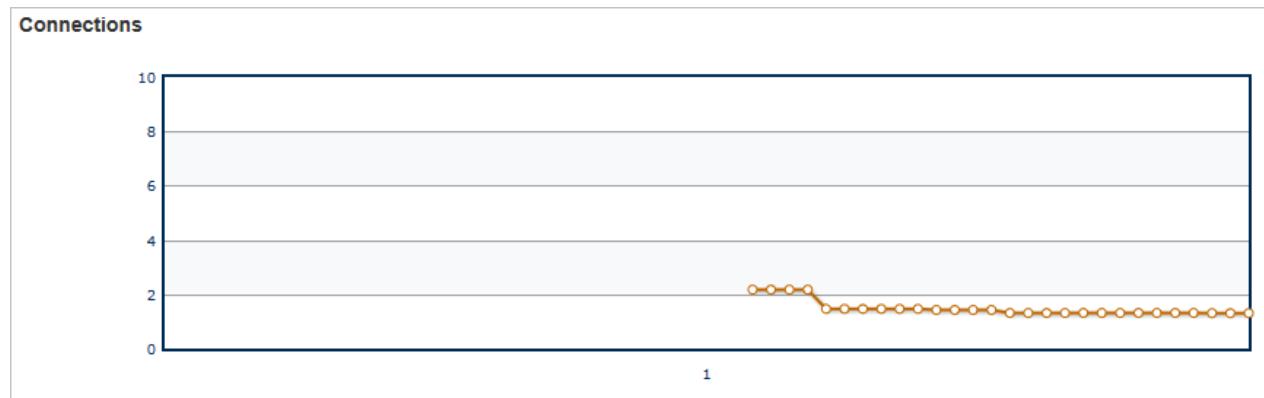
**TIP**

For a description of each cache status code option, see [Azure CDN Cache Status Codes](#).

A list of cache status codes is displayed directly above the graph. This list indicates each status code that can be included in the line graph and the current number of occurrences per second for that status code. By default, a line is displayed for each of these status codes in the graph. However, you can choose to only monitor the status codes that have special significance for your CDN configuration. To do this, check the desired status codes and clear all other options, then click **Refresh Graph**.

You can temporarily hide logged data for a particular status code. From the legend directly below the graph, click the status code you want to hide. The status code will be immediately hidden from the graph. Clicking that status code again will cause that option to be displayed again.

## Connections



This graph indicates how many connections have been established to your edge servers. Each request for an asset that passes through our CDN results in a connection.

## Next Steps

- Get notified with [Real-time alerts in Azure CDN](#)
- Dig deeper with [advanced HTTP reports](#)
- Analyze [usage patterns](#)

# Analyze edge node performance in Microsoft Azure CDN

7/5/2019 • 14 minutes to read • [Edit Online](#)

## IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

## Overview

Edge performance analytics provides granular information traffic and bandwidth usage for the CDN. This information can then be used to generate trending statistics, which allow you to gain insight on how your assets are being cached and delivered to your clients. In turn, this allows you to form a strategy on how to optimize the delivery of your content and to determine what issues should be tackled to better leverage the CDN. As a result, not only will you be able to improve data delivery performance, but you will also be able to reduce your CDN costs.

## NOTE

All reports use UTC/GMT notation when specifying a date/time.

## Reports and log collection

CDN activity data must be collected by the Edge Performance Analytics module before it can generate reports on it. This collection process occurs once a day and it covers the activity that took place during the previous day. This means that a report's statistics represent a sample of the day's statistics at the time it was processed, and do not necessarily contain the complete set of data for the current day. The primary function of these reports is to assess performance. They should not be used for billing purposes or exact numeric statistics.

## NOTE

The raw data from which Edge Performance Analytic reports are generated is available for at least 90 days.

## Dashboard

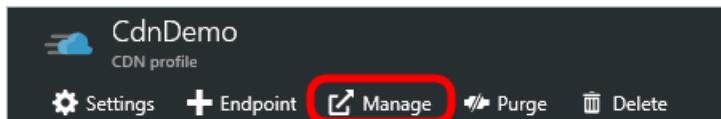
The Edge Performance Analytics dashboard tracks current and historical CDN traffic through a chart and statistics. Use this dashboard to detect recent and long-term trends on the performance of CDN traffic for your account.

This dashboard consists of:

- An interactive chart that allows the visualization of key metrics and trends.
- A timeline that provides a sense of long term patterns for key metrics and trends.
- Key metrics and statistical information on how our CDN network improves site traffic as measured by overall performance, usage, and efficiency.

## Accessing the edge performance dashboard

- From the CDN profile blade, click the **Manage** button.



The CDN management portal opens.

- Hover over the **Analytics** tab, then hover over the **Edge Performance Analytics** flyout. Click on **Dashboard**.

The edge node analytics dashboard is displayed.

### Chart

The dashboard contains a chart that tracks a metric over the time period selected in the timeline that appears directly below it. A timeline that graphs up to the last two years of CDN activity is displayed directly below the chart.

#### Using the chart

- By default, the cache efficiency rate for the last 30 days will be charted.
- This chart is generated from data collated on a daily basis.
- Hovering over a day on the line graph will indicate a date and the value of the metric on that date.
- Click **Highlight Weekends** to toggle an overlay of light gray vertical bars that represent weekends onto the chart. This type of overlay is useful for identifying traffic patterns over weekends.
- Click **View One Year Ago** to toggle an overlay of the previous year's activity over the same time period onto the chart. This type of comparison provides insight into long-term CDN usage patterns. The upper-right hand corner of the chart contains a legend that indicates the color code for each line graph.

#### Updating the chart

- Time Range:** Perform one of the following:
  - Select the desired region in the timeline. The chart will be updated with data that corresponds to the selected time period.
  - Double-click the chart to display all available historical data up to a maximum of two years.
- Metric:** Click the chart icon that appears next to the desired metric. The chart and the timeline will be refreshed with data for the corresponding metric.

### Key metrics and statistics

#### Efficiency metrics

The purpose of these metrics is to see whether cache efficiency can be improved. The main benefits derived from cache efficiency are:

- Reduced load on the origin server which may lead to:
  - Better web server performance.
  - Reduced operational costs.
- Improved data delivery acceleration since more requests will be served directly from the CDN.

FIELD	DESCRIPTION
Cache Efficiency	Indicates the percentage of data transferred that was served from cache. This metric measures when a cached version of the requested content was served directly from the CDN (edge servers) to requesters (e.g., web browser)

FIELD	DESCRIPTION
Hit Rate	Indicates the percentage of requests that were served from cache. This metric measures when a cached version of the requested content was served directly from the CDN (edge servers) to requesters (e.g., web browser).
% of Remote Bytes - No Cache Config	Indicates the percentage of traffic that was served from origin servers to the CDN (edge servers) that will not be cached as a result of the Bypass Cache feature (HTTP Rules Engine).
% of Remote Bytes - Expired Cache	Indicates the percentage of traffic that was served from origin servers to the CDN (edge servers) as a result of stale content revalidation.

#### Usage metrics

The purpose of these metrics is to provide insight into the following cost-cutting measures:

- Minimizing operational costs through the CDN.
- Reducing CDN expenditures through cache efficiency and compression.

#### NOTE

Traffic volume numbers represent traffic that was used in calculations of ratios and percentages, and may only show a portion of the total traffic for high-volume customers.

FIELD	DESCRIPTION
Ave Bytes Out	Indicates the average number of bytes transferred for each request served from the CDN (edge servers) to the requester (e.g., web browser).
No Cache Config Byte Rate	Indicates the percentage of traffic served from the CDN (edge servers) to the requester (e.g., web browser) that will not be cached due to the Bypass Cache feature.
Compressed Byte Rate	Indicates the percentage of traffic sent from the CDN (edge servers) to requesters (e.g., web browser) in a compressed format.
Bytes Out	Indicates the amount of data, in bytes, that were delivered from the CDN (edge servers) to the requester (e.g., web browser).
Bytes In	Indicates the amount of data, in bytes, sent from requesters (e.g., web browser) to the CDN (edge servers).
Bytes Remote	Indicates the amount of data, in bytes, sent from CDN and customer origin servers to the CDN (edge servers).

#### Performance Metrics

The purpose of these metrics is to track overall CDN performance for your traffic.

FIELD	DESCRIPTION
Transfer Rate	Indicates the average rate at which content was transferred from the CDN to a requester.
Duration	Indicates the average time, in milliseconds, it took to deliver an asset to a requester (e.g., web browser).
Compressed Request Rate	Indicates the percentage of hits that were delivered from the CDN (edge servers) to the requester (e.g., web browser) in a compressed format.
4xx Error Rate	Indicates the percentage of hits that generated a 4xx status code.
5xx Error Rate	Indicates the percentage of hits that generated a 5xx status code.
Hits	Indicates the number of requests for CDN content.

#### Secure Traffic Metrics

The purpose of these metrics is to track CDN performance for HTTPS traffic.

FIELD	DESCRIPTION
Secure Cache Efficiency	Indicates the percentage of data transferred for HTTPS requests that were served from cache. This metric measures when a cached version of the requested content was served directly from the CDN (edge servers) to requesters (e.g., web browser) over HTTPS.
Secure Transfer Rate	Indicates the average rate at which content was transferred from the CDN (edge servers) to requesters (e.g., web servers) over HTTPS.
Average Secure Duration	Indicates the average time, in milliseconds, it took to deliver an asset to a requester (e.g., web browser) over HTTPS.
Secure Hits	Indicates the number of HTTPS requests for CDN content.
Secure Bytes Out	Indicates the amount of HTTPS traffic, in bytes, that were delivered from the CDN (edge servers) to the requester (e.g., web browser).

## Reports

Each report in this module contains a chart and statistics on bandwidth and traffic usage for different types of metrics (e.g., HTTP status codes, cache status codes, request URL, etc.). This information may be used to delve deeper into how content is being served to your clients and to fine-tune CDN behavior to improve data delivery performance.

#### Accessing the edge performance reports

- From the CDN profile blade, click the **Manage** button.

The screenshot shows a dark-themed interface for managing a CDN profile named 'CdnDemo'. At the top, there are tabs for 'Settings', 'Endpoint', 'Manage' (which is highlighted with a red circle), 'Purge', and 'Delete'. Below the tabs, the main content area displays the edge node analytics reports screen.

The CDN management portal opens.

2. Hover over the **Analytics** tab, then hover over the **Edge Performance Analytics** flyout. Click on **HTTP Large Object**.

The edge node analytics reports screen is displayed.

REPORT	DESCRIPTION
Daily Summary	Allows you to view daily traffic trends over a specified time period. Each bar on this graph represents a particular date. The size of the bar indicates the total number of hits that occurred on that date.
Hourly Summary	Allows you to view hourly traffic trends over a specified time period. Each bar on this graph represents a single hour on a particular date. The size of the bar indicates the total number of hits that occurred during that hour.
Protocols	Displays the breakdown of traffic between the HTTP and HTTPS protocols. A donut chart indicates the percentage of hits that occurred for each type of protocol.
HTTP Methods	Allows you to get a quick sense of which HTTP methods are being used to request your data. Typically, the most common HTTP request methods are GET, HEAD, and POST. A donut chart indicates the percentage of hits that occurred for each type of HTTP request method.
URLs	Contains a graph that displays the top 10 requested URLs. A bar is displayed for each URL. The height of the bar indicates how many hits that particular URL has generated over the time span covered by the report. Statistics for the top 100 requested URLs are displayed directly below this graph.
CNAMEs	Contains a graph that displays the top 10 CNAMEs used to request assets over the time span of a report. Statistics for the top 100 requested CNAMEs are displayed directly below this graph.
Origins	Contains a graph that displays the top 10 CDN or customer origin servers from which assets were requested over a specified period of time. Statistics for the top 100 requested CDN or customer origin servers are displayed directly below this graph. Customer origin servers are identified by the name defined in the Directory Name option.
Geo POPs	Shows how much of your traffic is being routed through a particular point-of-presence (POP). The three-letter abbreviation represents a POP in our CDN network.

Report	Description
Clients	Contains a graph that displays the top 10 clients that requested assets over a specified period of time. For the purposes of this report, all requests that originate from the same IP address are considered to be from the same client. Statistics for the top 100 clients are displayed directly below this graph. This report is useful for determining download activity patterns for your top clients.
Cache Statuses	Gives a detailed breakdown of cache behavior, which may reveal approaches for improving the overall end-user experience. Since the fastest performance comes from cache hits, you can optimize data delivery speeds by minimizing cache misses and expired cache hits.
NONE Details	Contains a graph that displays the top 10 URLs for assets for which cache content freshness was not checked over a specified period of time. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
CONFIG_NOCACHE Details	Contains a graph that displays the top 10 URLs for assets that were not cached due to the customer's CDN configuration. These types of assets were served directly from the origin server. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
UNCACHEABLE Details	Contains a graph that displays the top 10 URLs for assets that could not be cached due to request header data. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
TCP_HIT Details	Contains a graph that displays the top 10 URLs for assets that are served immediately from cache. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
TCP_MISS Details	Contains a graph that displays the top 10 URLs for assets that have a cache status of TCP_MISS. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
TCP_EXPIRED_HIT Details	Contains a graph that displays the top 10 URLs for stale assets that were served directly from the POP. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
TCP_EXPIRED_MISS Details	Contains a graph that displays the top 10 URLs for stale assets for which a new version had to be retrieved from the origin server. Statistics for the top 100 URLs for these types of assets are displayed directly below this graph.
TCP_CLIENT_REFRESH_MISS Details	Contains a bar chart that displays the top 10 URLs for assets were retrieved from an origin server due to a no-cache request from the client. Statistics for the top 100 URLs for these types of requests are displayed directly below this chart.

REPORT	DESCRIPTION
Client Request Types	Indicates the type of requests that were made by HTTP clients (e.g., browsers). This report includes a donut chart that provides a sense as to how requests are being handled. Bandwidth and traffic information for each request type is displayed below the chart.
User Agent	Contains a bar graph displaying the top 10 user agents to request your content through our CDN. Typically, a user agent is a web browser, media player, or a mobile phone browser. Statistics for the top 100 user agents are displayed directly below this chart.
Referrers	Contains a bar graph displaying the top 10 referrers to content accessed through our CDN. Typically, a referrer is the URL of the web page or resource that links to your content. Detailed information is provided below the graph for the top 100 referrers.
Compression Types	Contains a donut chart that breaks down requested assets by whether they were compressed by our edge servers. The percentage of compressed assets is broken down by the type of compression used. Detailed information is provided below the graph for each compression type and status.
File Types	Contains a bar graph that displays the top 10 file types that have been requested through our CDN for your account. For the purposes of this report, a file type is defined by the asset's file name extension and Internet media type (e.g., .html [text/html], .htm [text/html], .aspx [text/html], etc.). Detailed information is provided below the graph for the top 100 file types.
Unique Files	Contains a graph that plots the total number of unique assets that were requested on a particular day over a specified period of time.
Token Auth Summary	Contains a pie chart that provides a quick overview on whether requested assets were protected by Token-Based Authentication. Protected assets are displayed in the chart according to the results of their attempted authentication.
Token Auth Deny Details	Contains a bar graph that allows you to view the top 10 requests that were denied due to Token-Based Authentication.
HTTP Response Codes	Provides a breakdown of the HTTP status codes (e.g., 200 OK, 403 Forbidden, 404 Not Found, etc.) that were delivered to your HTTP clients by our edge servers. A pie chart allows you to quickly assess how your assets were served. Detailed statistical data is provided for each response code below the graph.
404 Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a 404 Not Found response code.

REPORT	DESCRIPTION
403 Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a 403 Forbidden response code. A 403 Forbidden response code occurs when a request is denied by a customer origin server or an edge server on our POP.
4xx Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a response code in the 400 range. Excluded from this report are 403 Not Found and 404 Forbidden response codes. Typically, a 4xx response code occurs when a request is denied as a result of a client error.
504 Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a 504 Gateway Timeout response code. A 504 Gateway Timeout response code occurs when a timeout occurs when an HTTP proxy is trying to communicate with another server. In the case of our CDN, a 504 Gateway Timeout response code typically occurs when an edge server is unable to establish communication with a customer origin server.
502 Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a 502 Bad Gateway response code. A 502 Bad Gateway response code occurs when an HTTP protocol failure occurs between a server and an HTTP proxy. In the case of our CDN, a 502 Bad Gateway response code typically occurs when a customer origin server returns an invalid response to an edge server. A response is invalid if it cannot be parsed or if it is incomplete.
5xx Errors	Contains a bar graph that allows you to view the top 10 requests that resulted in a response code in the 500 range. Excluded from this report are 502 Bad Gateway and 504 Gateway Timeout response codes.

## See also

- [Azure CDN Overview](#)
- [Real-time stats in Microsoft Azure CDN](#)
- [Overriding default HTTP behavior using the rules engine](#)
- [Advanced HTTP Reports](#)

# Real-time alerts in Microsoft Azure CDN

7/5/2019 • 3 minutes to read • [Edit Online](#)

## IMPORTANT

This is a feature of **Azure CDN Premium from Verizon** only, to configure rules on **Azure CDN from Microsoft** please use the [Standard rules engine](#). Advanced rules are not available for **Azure CDN from Akamai**. For a full comparison of CDN features, see [Azure CDN product features](#).

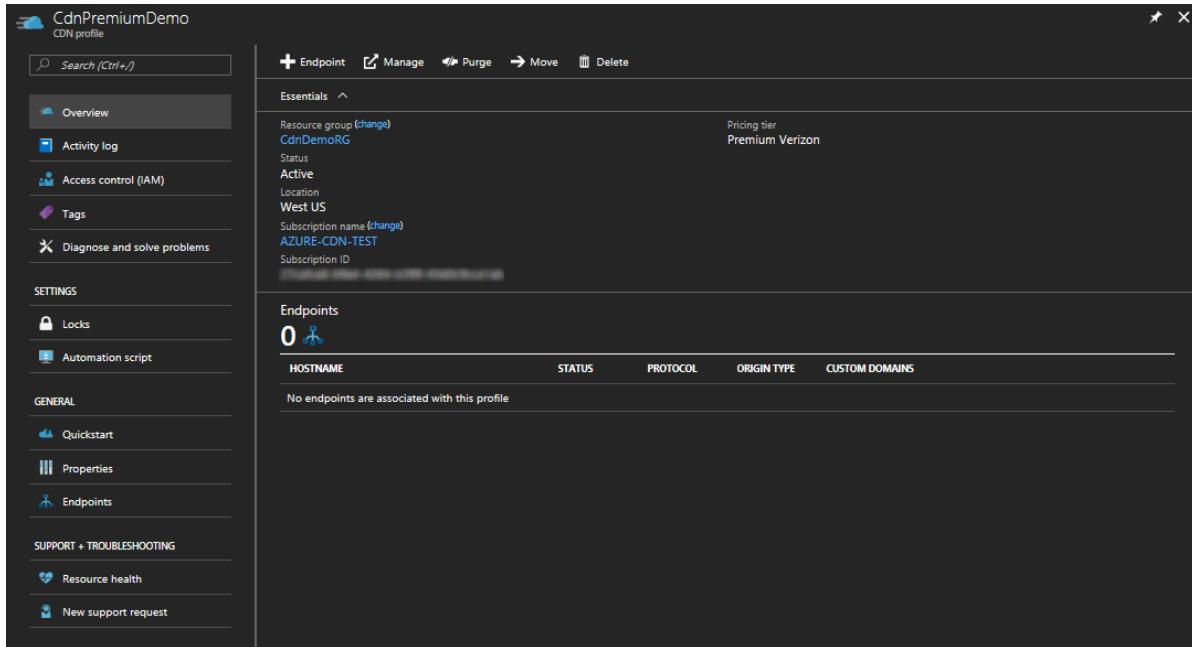
## Overview

This document explains real-time alerts in Microsoft Azure CDN. This functionality provides real-time notifications about the performance of the endpoints in your CDN profile. You can set up email or HTTP alerts based on:

- Bandwidth
- Status Codes
- Cache Statuses
- Connections

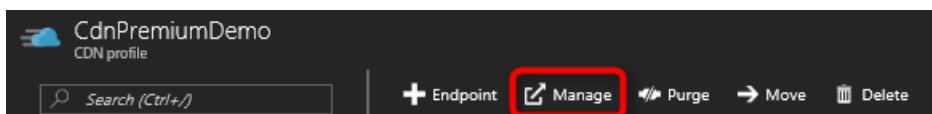
## Creating a real-time alert

1. In the [Azure portal](#), browse to your CDN profile.



The screenshot shows the Azure portal interface for a CDN profile named 'CdnPremiumDemo'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Locks, Automation script, Quickstart, Properties, Endpoints, Resource health, and New support request. The main content area displays the 'Essentials' section with details like Resource group (CdnDemoRG), Status (Active), Location (West US), Pricing tier (Premium Verizon), and Subscription name (AZURE-CDN-TEST). Below this is the 'Endpoints' section, which currently shows 0 endpoints associated with the profile. At the top of the page, there is a navigation bar with icons for Endpoint, Manage, Purge, Move, and Delete, with the 'Manage' icon being the one highlighted by a red box.

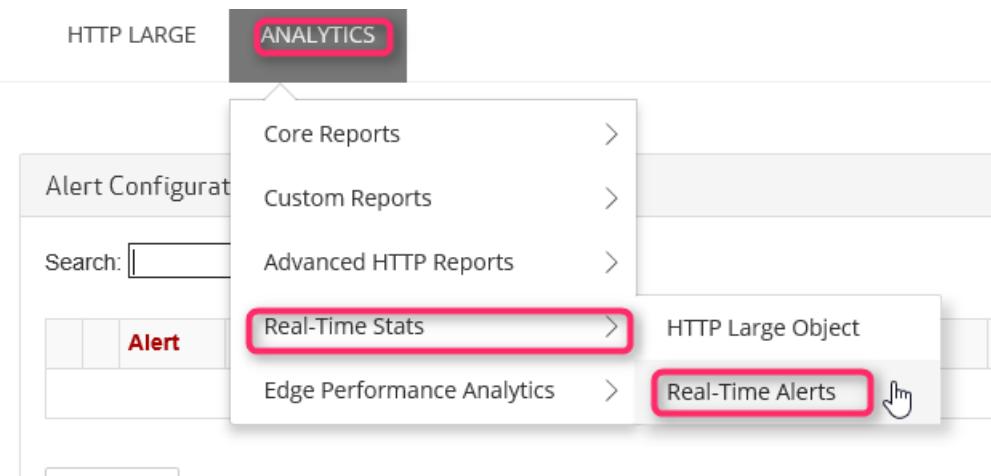
2. From the CDN profile blade, click the **Manage** button.



The screenshot shows the Azure portal interface for managing the 'CdnPremiumDemo' CDN profile. The top navigation bar includes icons for Endpoint, Manage (which is highlighted with a red box), Purge, Move, and Delete. The main content area is currently empty, showing a placeholder message: 'The CDN management portal opens.'

The CDN management portal opens.

3. Hover over the **Analytics** tab, then hover over the **Real-Time Stats** flyout. Click on **Real-Time Alerts**.



The list of existing alert configurations (if any) is displayed.

4. Click the **Add Alert** button.

A screenshot of the "Alert Configurations" page. At the top left, it says "Alert Configurations". Below this is a search bar with a "Search" button. A table header row includes columns for Alert, Media Type, Expression, Notify On, Interval, Enabled, Email, and HTTP. Under the "Enabled" column, it shows "1 | 0 item(s)". At the bottom left, a red-bordered box highlights the "Add Alert" button.

A form for creating a new alert is displayed.

A screenshot of the "Alert Configuration" form. It includes fields for "Name" (with a red-bordered box around the input field), "Media Type" (set to "HTTP Large Object"), "Expression" (set to "Bandwidth Mbps > 1"), "Interval" (set to "Every 5 minutes"), "Metric" (set to "Bandwidth Mbps"), "Operator" (set to ">"), "Trigger value" (set to "1"), and "Notify on" (set to "Condition Start"). There is also a checkbox for "Alert Enabled" which is currently unchecked.

5. If you want this alert to be active when you click **Save**, check the **Alert Enabled** checkbox.
6. Enter a descriptive name for your alert in the **Name** field.
7. In the **Media Type** dropdown, select **HTTP Large Object**.

A screenshot of the "Media Type" dropdown. The option "HTTP Large Object" is selected and highlighted with a red border. Below the dropdown, there is a yellow box containing the word "IMPORTANT" and a note: "You must select **HTTP Large Object** as the **Media Type**. The other choices are not used by **Azure CDN from Verizon**. Failure to select **HTTP Large Object** causes your alert to never be triggered."

8. Create an **Expression** to monitor by selecting a **Metric**, **Operator**, and **Trigger value**.

- For **Metric**, select the type of condition you want monitored. **Bandwidth Mbps** is the amount of bandwidth usage in megabits per second. **Total Connections** is the number of concurrent HTTP connections to our edge servers. For definitions of the various cache statuses and status codes, see [Azure CDN Cache Status Codes](#) and [Azure CDN HTTP Status Codes](#)
- **Operator** is the mathematical operator that establishes the relationship between the metric and the trigger value.
- **Trigger Value** is the threshold value that must be met before a notification is sent.

In the following example, the created expression indicates that a notification is sent when the number of 404 status codes is greater than 25.

The screenshot shows a configuration panel for an alert expression. On the left, it says "Expression:" followed by a dropdown menu containing "Status Code : 404 per second". To the right of the dropdown are three operators: "<", ">", and "=" with a dropdown arrow above them. The operator ">" is selected. To the right of the operators is a text input field containing the value "25". Below the dropdown and operators is the word "Metric". To the right of the value input is the word "Operator" and to its right is "Trigger value".

9. For **Interval**, enter how frequently you would like the expression evaluated.

10. In the **Notify on** dropdown, select when you would like to be notified when the expression is true.

- **Condition Start** indicates that a notification is sent when the specified condition is first detected.
- **Condition End** indicates that a notification is sent when the specified condition is no longer detected. This notification can only be triggered after our network monitoring system detected that the specified condition occurred.
- **Continuous** indicates that a notification is sent each time that the network monitoring system detects the specified condition. Keep in mind that the network monitoring system checks only once per interval for the specified condition.
- **Condition Start and End** indicates that a notification is sent the first time that the specified condition is detected and once again when the condition is no longer detected.

11. If you want to receive notifications by email, check the **Notify by Email** checkbox.

The screenshot shows the "Notify by Email" configuration screen. At the top is a checkbox labeled "Notify by Email". Below it are fields for "From" (no-reply@verizondigitalmedia.com), "To" (empty), and "Subject" (The [Name] alert condition has been detected.). To the right of these fields is a "Body" section and a "Available keywords:" list. The "Available keywords:" list includes: [Name], [MediaType], [Expression], [Interval], [NotificationCondition], [CurrentValue], [TriggerValue], [Metric], [Operator], and [AlertDuration]. At the bottom are "Test Notification" and "Reset" buttons.

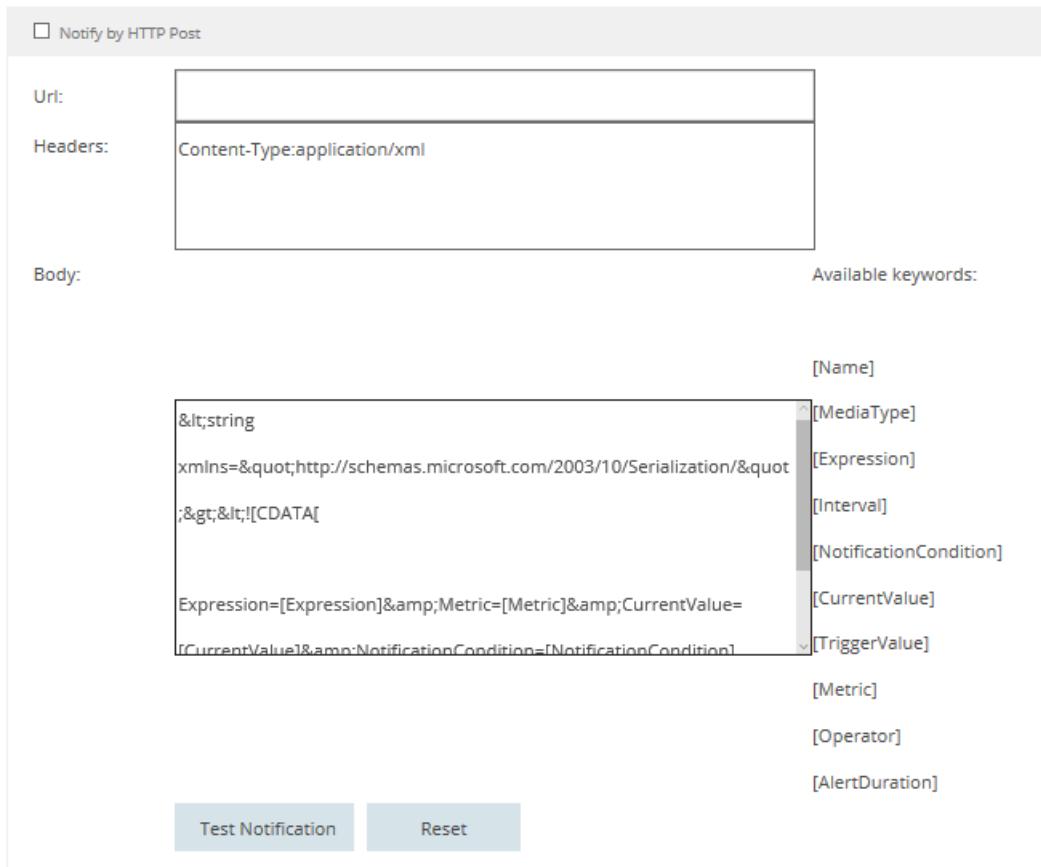
In the **To** field, enter the email address you where you want notifications sent. For **Subject** and **Body**, you may leave the default, or you may customize the message using the **Available keywords** list to

dynamically insert alert data when the message is sent.

**NOTE**

You can test the email notification by clicking the **Test Notification** button, but only after the alert configuration has been saved.

12. If you want notifications to be posted to a web server, check the **Notify by HTTP Post** checkbox.



In the **Url** field, enter the URL you where you want the HTTP message posted. In the **Headers** textbox, enter the HTTP headers to be sent in the request. For **Body**, you may customize the message by using the **Available keywords** list to dynamically insert alert data when the message is sent. **Headers** and **Body** default to an XML payload similar to the following example:

```
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">
  <![CDATA[Expression=Status Code : 404 per second > 25&Metric=Status Code : 404 per
second&CurrentValue=[CurrentValue]&NotificationCondition=Condition Start]]>
</string>
```

**NOTE**

You can test the HTTP Post notification by clicking the **Test Notification** button, but only after the alert configuration has been saved.

13. Click the **Save** button to save your alert configuration. If you checked **Alert Enabled** in step 5, your alert is now active.

## Next Steps

- Analyze [Real-time stats](#) in Azure CDN
- Dig deeper with [advanced HTTP reports](#)
- Analyze [usage patterns](#)

# Get started with Azure CDN development

7/5/2019 • 9 minutes to read • [Edit Online](#)

You can use the [Azure CDN Library for .NET](#) to automate creation and management of CDN profiles and endpoints. This tutorial walks through the creation of a simple .NET console application that demonstrates several of the available operations. This tutorial is not intended to describe all aspects of the Azure CDN Library for .NET in detail.

You need Visual Studio 2015 to complete this tutorial. [Visual Studio Community 2015](#) is freely available for download.

## TIP

The [completed project from this tutorial](#) is available for download on MSDN.

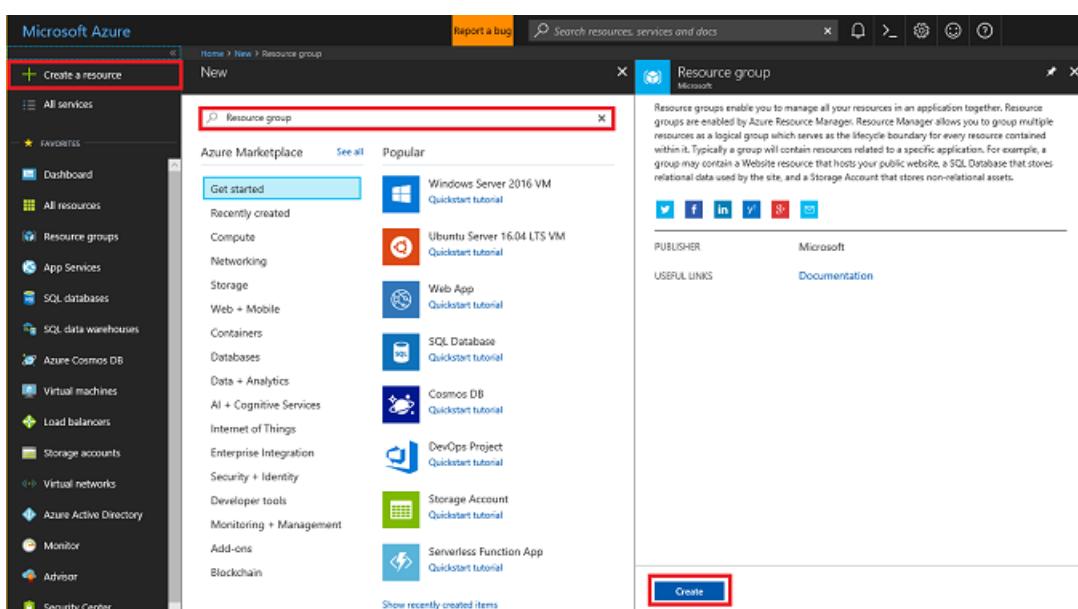
## Prerequisites

Before writing CDN management code, you must do some preparation to enable the code to interact with the Azure Resource Manager. To do this preparation, you need to:

- Create a resource group to contain the CDN profile created in this tutorial
- Configure Azure Active Directory to provide authentication for the application
- Apply permissions to the resource group so that only authorized users from your Azure AD tenant can interact with the CDN profile

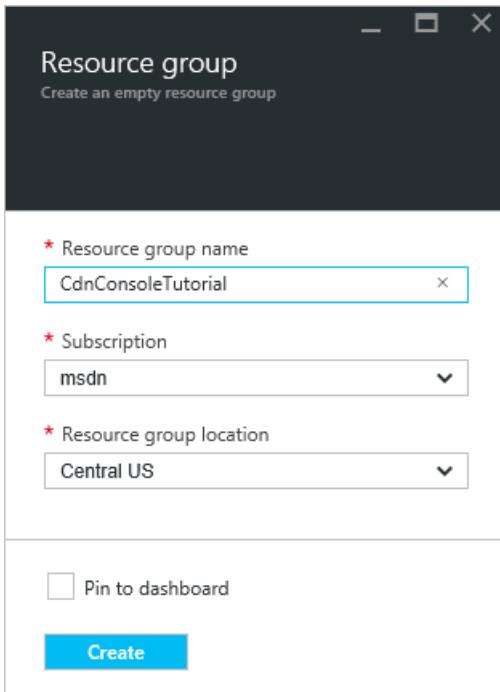
### Creating the resource group

1. Sign in to the [Azure Portal](#).
2. Click **Create a resource**.
3. Search for **Resource group** and in the Resource group pane, click **Create**.

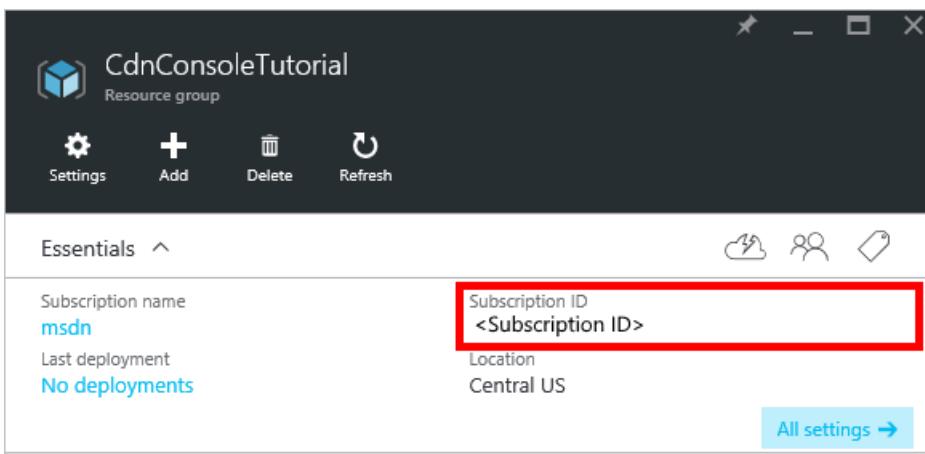


4. Name your resource group *CdnConsoleTutorial*. Select your subscription and choose a location near you. If you wish, you can click the **Pin to dashboard** checkbox to pin the resource group to the dashboard in the

portal. Pinning makes it easier to find later. After you've made your selections, click **Create**.



- After the resource group is created, if you didn't pin it to your dashboard, you can find it by clicking **Browse**, then **Resource Groups**. To open it, click the resource group. Make a note of your **Subscription ID**. We need it later.



### Creating the Azure AD application and applying permissions

There are two approaches to app authentication with Azure Active Directory: Individual users or a service principal. A service principal is similar to a service account in Windows. Instead of granting a particular user permissions to interact with the CDN profiles, permissions are instead granted to the service principal. Service principals are typically used for automated, non-interactive processes. Even though this tutorial is writing an interactive console app, we'll focus on the service principal approach.

Creating a service principal consists of several steps, including creating an Azure Active Directory application. To create it, we're going to [follow this tutorial](#).

## IMPORTANT

Be sure to follow all the steps in the [linked tutorial](#). It is *important* that you complete it exactly as described. Make sure to note your **tenant ID**, **tenant domain name** (commonly a `.onmicrosoft.com` domain unless you've specified a custom domain), **client ID**, and **client authentication key**, as we need this information later. Be careful to guard your **client ID** and **client authentication key**, as these credentials can be used by anyone to execute operations as the service principal.

When you get to the step named Configure multi-tenant application, select **No**.

When you get to the step **Assign the application to a role**, use the resource group created earlier, `CdnConsoleTutorial`, but instead of the **Reader** role, assign the **CDN Profile Contributor** role. After you assign the application the **CDN Profile Contributor** role on your resource group, return to this tutorial.

Once you've created your service principal and assigned the **CDN Profile Contributor** role, the **Users** blade for your resource group should look similar to the following image.

USER	ROLE	ACCESS
CdnConsoleService	CDN Profile Contr...	Assigned
Subscription admins	Owner	Inherited

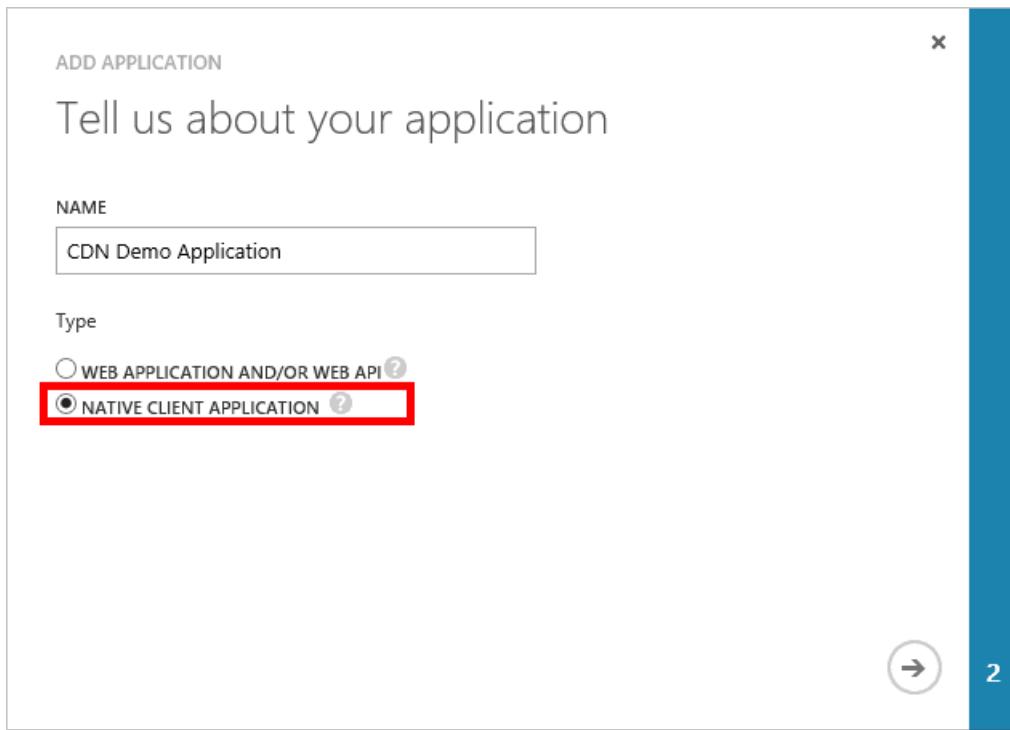
## Interactive user authentication

If, instead of a service principal, you'd rather have interactive individual user authentication, the process is similar to that for a service principal. In fact, you need to follow the same procedure, but make a few minor changes.

## IMPORTANT

Only follow these next steps if you are choosing to use individual user authentication instead of a service principal.

1. When creating your application, instead of **Web Application**, choose **Native application**.



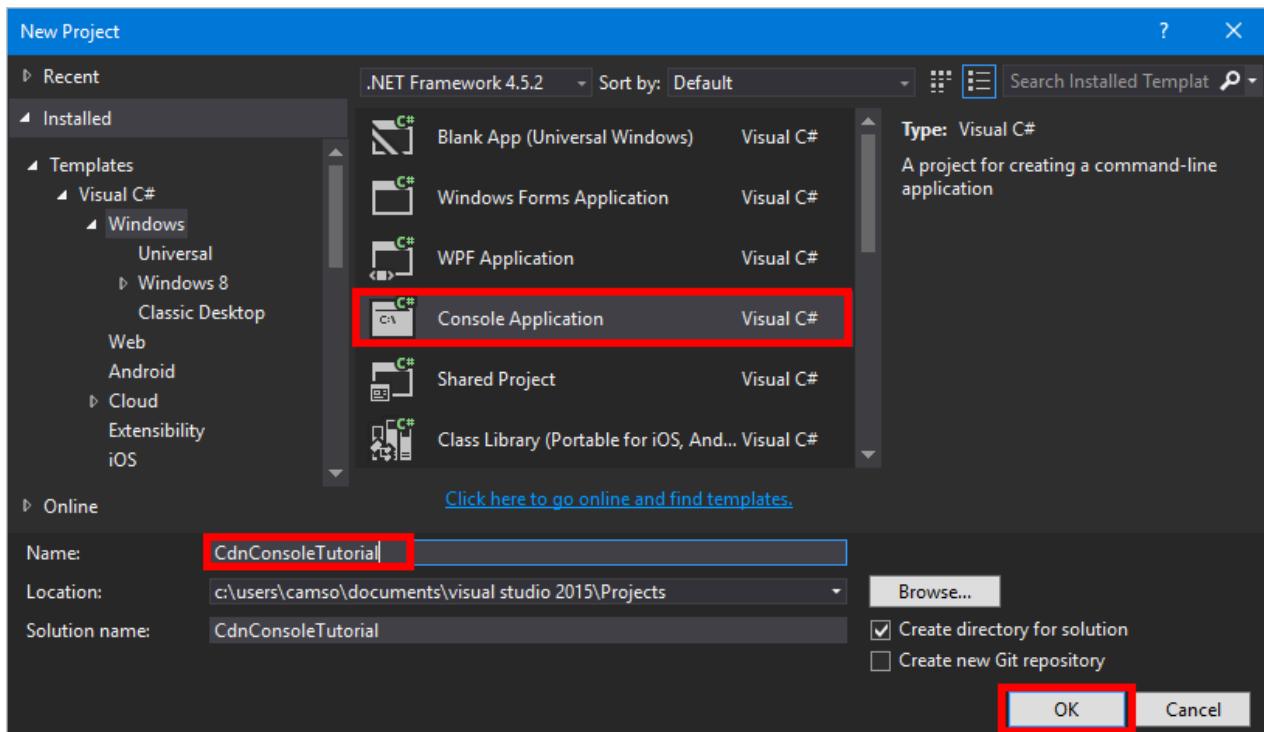
2. On the next page, you are prompted for a **redirect URI**. The URI won't be validated, but remember what you entered. You need it later.
3. There is no need to create a **client authentication key**.
4. Instead of assigning a service principal to the **CDN Profile Contributor** role, we're going to assign individual users or groups. In this example, you can see that I've assigned *CDN Demo User* to the **CDN Profile Contributor** role.

Users			
CdnConsoleTutorial			
	Add	Roles	
 CDN Demo User <user>@<tenant>.onmicrosoft.com		CDN Profile Contr... Assigned	...
 Subscription admins ⓘ		Owner	Inherited
			...

## Create your project and add Nuget packages

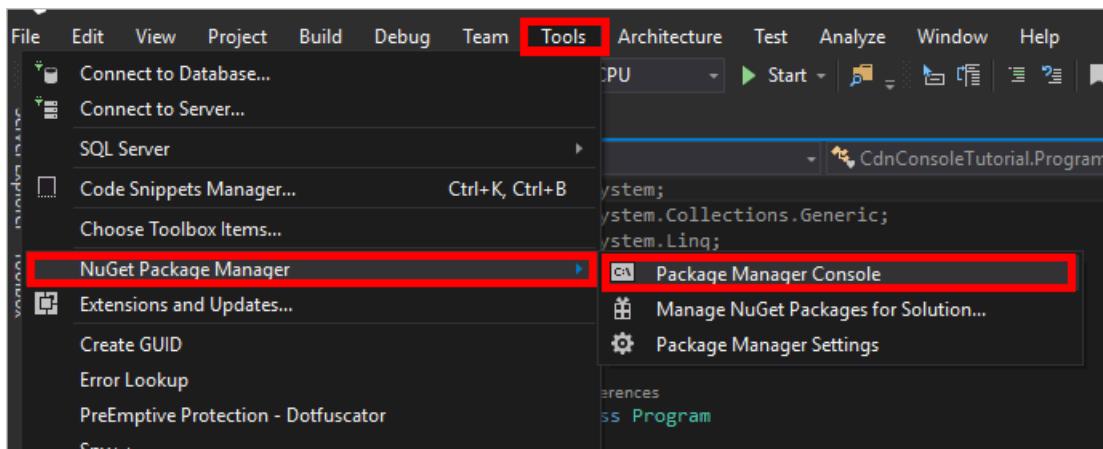
Now that we've created a resource group for our CDN profiles and given our Azure AD application permission to manage CDN profiles and endpoints within that group, we can start creating our application.

From within Visual Studio 2015, click **File, New, Project...** to open the new project dialog. Expand **Visual C#**, then select **Windows** in the pane on the left. Click **Console Application** in the center pane. Name your project, then click **OK**.



Our project is going to use some Azure libraries contained in Nuget packages. Let's add those to the project.

1. Click the **Tools** menu, **NuGet Package Manager**, then **Package Manager Console**.



2. In the Package Manager Console, execute the following command to install the **Active Directory Authentication Library (ADAL)**:

```
Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory
```

3. Execute the following to install the **Azure CDN Management Library**:

```
Install-Package Microsoft.Azure.Management.Cdn
```

## Directives, constants, main method, and helper methods

Let's get the basic structure of our program written.

1. Back in the Program.cs tab, replace the `using` directives at the top with the following:

```
using System;
using System.Collections.Generic;
using Microsoft.Azure.Management.Cdn;
using Microsoft.Azure.Management.Cdn.Models;
using Microsoft.Azure.Management.Resources;
using Microsoft.Azure.Management.Resources.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.Rest;
```

2. We need to define some constants our methods will use. In the `Program` class, but before the `Main` method, add the following. Be sure to replace the placeholders, including the **<angle brackets>**, with your own values as needed.

```
//Tenant app constants
private const string clientID = "<YOUR CLIENT ID>";
private const string clientSecret = "<YOUR CLIENT AUTHENTICATION KEY>"; //Only for service principals
private const string authority = "https://login.microsoftonline.com/<YOUR TENANT ID>/<YOUR TENANT
DOMAIN NAME>";

//Application constants
private const string subscriptionId = "<YOUR SUBSCRIPTION ID>";
private const string profileName = "CdnConsoleApp";
private const string endpointName = "<A UNIQUE NAME FOR YOUR CDN ENDPOINT>";
private const string resourceGroupName = "CdnConsoleTutorial";
private const string resourceLocation = "<YOUR PREFERRED AZURE LOCATION, SUCH AS Central US>";
```

3. Also at the class level, define these two variables. We'll use these later to determine if our profile and endpoint already exist.

```
static bool profileAlreadyExists = false;
static bool endpointAlreadyExists = false;
```

4. Replace the `Main` method as follows:

```

static void Main(string[] args)
{
    //Get a token
    AuthenticationResult authResult = GetAccessToken();

    // Create CDN client
    CdnManagementClient cdn = new CdnManagementClient(new TokenCredentials(authResult.AccessToken))
        { SubscriptionId = subscriptionId };

    ListProfilesAndEndpoints(cdn);

    // Create CDN Profile
    CreateCdnProfile(cdn);

    // Create CDN Endpoint
    CreateCdnEndpoint(cdn);

    Console.WriteLine();

    // Purge CDN Endpoint
    PromptPurgeCdnEndpoint(cdn);

    // Delete CDN Endpoint
    PromptDeleteCdnEndpoint(cdn);

    // Delete CDN Profile
    PromptDeleteCdnProfile(cdn);

    Console.WriteLine("Press Enter to end program.");
    Console.ReadLine();
}

```

5. Some of our other methods are going to prompt the user with "Yes/No" questions. Add the following method to make that a little easier:

```

private static bool PromptUser(string Question)
{
    Console.Write(Question + " (Y/N): ");
    var response = Console.ReadKey();
    Console.WriteLine();
    if (response.Key == ConsoleKey.Y)
    {
        return true;
    }
    else if (response.Key == ConsoleKey.N)
    {
        return false;
    }
    else
    {
        // They pressed something other than Y or N. Let's ask them again.
        return PromptUser(Question);
    }
}

```

Now that the basic structure of our program is written, we should create the methods called by the `Main` method.

## Authentication

Before we can use the Azure CDN Management Library, we need to authenticate our service principal and obtain an authentication token. This method uses ADAL to retrieve the token.

```
private static AuthenticationResult GetAccessToken()
{
    AuthenticationContext authContext = new AuthenticationContext(authority);
    ClientCredential credential = new ClientCredential(clientID, clientSecret);
    AuthenticationResult authResult =
        authContext.AcquireTokenAsync("https://management.core.windows.net/", credential).Result;

    return authResult;
}
```

If you are using individual user authentication, the `GetAccessToken` method will look slightly different.

**IMPORTANT**

Only use this code sample if you are choosing to have individual user authentication instead of a service principal.

```
private static AuthenticationResult GetAccessToken()
{
    AuthenticationContext authContext = new AuthenticationContext(authority);
    AuthenticationResult authResult = authContext.AcquireTokenAsync("https://management.core.windows.net/",
        clientID, new Uri("http://<redirect URI>"), new
        PlatformParameters(PromptBehavior.RefreshSession)).Result;

    return authResult;
}
```

Be sure to replace `<redirect URI>` with the redirect URI you entered when you registered the application in Azure AD.

## List CDN profiles and endpoints

Now we're ready to perform CDN operations. The first thing our method does is list all the profiles and endpoints in our resource group, and if it finds a match for the profile and endpoint names specified in our constants, makes a note of that for later so we don't try to create duplicates.

```

private static void ListProfilesAndEndpoints(CdnManagementClient cdn)
{
    // List all the CDN profiles in this resource group
    var profileList = cdn.Profiles.ListByResourceGroup(resourceGroupName);
    foreach (Profile p in profileList)
    {
        Console.WriteLine("CDN profile {0}", p.Name);
        if (p.Name.Equals(profileName, StringComparison.OrdinalIgnoreCase))
        {
            // Hey, that's the name of the CDN profile we want to create!
            profileAlreadyExists = true;
        }
    }

    //List all the CDN endpoints on this CDN profile
    Console.WriteLine("Endpoints:");
    var endpointList = cdn.Endpoints.ListByProfile(p.Name, resourceGroupName);
    foreach (Endpoint e in endpointList)
    {
        Console.WriteLine("-{0} ({1})", e.Name, e.HostName);
        if (e.Name.Equals(endpointName, StringComparison.OrdinalIgnoreCase))
        {
            // The unique endpoint name already exists.
            endpointAlreadyExists = true;
        }
    }
    Console.WriteLine();
}
}

```

## Create CDN profiles and endpoints

Next, we'll create a profile.

```

private static void CreateCdnProfile(CdnManagementClient cdn)
{
    if (profileAlreadyExists)
    {
        Console.WriteLine("Profile {0} already exists.", profileName);
    }
    else
    {
        Console.WriteLine("Creating profile {0}.", profileName);
        ProfileCreateParameters profileParms =
            new ProfileCreateParameters() { Location = resourceLocation, Sku = new
        Sku(SkuName.StandardVerizon) };
        cdn.Profiles.Create(profileName, profileParms, resourceGroupName);
    }
}

```

Once the profile is created, we'll create an endpoint.

```

private static void CreateCdnEndpoint(CdnManagementClient cdn)
{
    if (endpointAlreadyExists)
    {
        Console.WriteLine("Profile {0} already exists.", profileName);
    }
    else
    {
        Console.WriteLine("Creating endpoint {0} on profile {1}.", endpointName, profileName);
        EndpointCreateParameters endpointParms =
            new EndpointCreateParameters()
        {
            Origins = new List<DeepCreatedOrigin>() { new DeepCreatedOrigin("Contoso", "www.contoso.com") },
            IsHttpAllowed = true,
            IsHttpsAllowed = true,
            Location = resourceLocation
        };
        cdn.Endpoints.Create(endpointName, endpointParms, profileName, resourceGroupName);
    }
}

```

#### NOTE

The example above assigns the endpoint an origin named *Contoso* with a hostname `www.contoso.com`. You should change this to point to your own origin's hostname.

## Purge an endpoint

Assuming the endpoint has been created, one common task that we might want to perform in our program is purging the content in our endpoint.

```

private static void PromptPurgeCdnEndpoint(CdnManagementClient cdn)
{
    if (PromptUser(String.Format("Purge CDN endpoint {0}?", endpointName)))
    {
        Console.WriteLine("Purging endpoint. Please wait...");
        cdn.Endpoints.PurgeContent(endpointName, profileName, resourceGroupName, new List<string>() { /* */ });
        Console.WriteLine("Done.");
        Console.WriteLine();
    }
}

```

#### NOTE

In the example above, the string `/*` denotes that I want to purge everything in the root of the endpoint path. This is equivalent to checking **Purge All** in the Azure portal's "purge" dialog. In the `CreateCdnProfile` method, I created our profile as an **Azure CDN from Verizon** profile using the code `Sku = new Sku(SkuName.StandardVerizon)`, so this will be successful. However, **Azure CDN from Akamai** profiles do not support **Purge All**, so if I was using an Akamai profile for this tutorial, I would need to include specific paths to purge.

## Delete CDN profiles and endpoints

The last methods will delete our endpoint and profile.

```

private static void PromptDeleteCdnEndpoint(CdnManagementClient cdn)
{
    if(PromptUser(String.Format("Delete CDN endpoint {0} on profile {1}?", endpointName, profileName)))
    {
        Console.WriteLine("Deleting endpoint. Please wait...");
        cdn.Endpoints.DeleteIfExists(endpointName, profileName, resourceGroupName);
        Console.WriteLine("Done.");
        Console.WriteLine();
    }
}

private static void PromptDeleteCdnProfile(CdnManagementClient cdn)
{
    if(PromptUser(String.Format("Delete CDN profile {0}?", profileName)))
    {
        Console.WriteLine("Deleting profile. Please wait...");
        cdn.Profiles.DeleteIfExists(profileName, resourceGroupName);
        Console.WriteLine("Done.");
        Console.WriteLine();
    }
}

```

## Running the program

We can now compile and run the program by clicking the **Start** button in Visual Studio.

```

file:///c:/users/camso/documents/visual studio 2015/Projects/CdnConsoleTutorial
Creating profile CdnConsoleApp.
Creating endpoint CdnConsoleEndpoint on profile CdnConsoleApp.

Purge CDN endpoint CdnConsoleEndpoint? (Y/N): -

```

When the program reaches the above prompt, you should be able to return to your resource group in the Azure portal and see that the profile has been created.

NAME	TYPE	RESOURCE GRO...	LOCATION	SUBSCRIPTI...
CdnConsoleApp	CDN profile	CdnConsoleTu...	Central US	msdn

We can then confirm the prompts to run the rest of the program.

```
file:///c:/users/camso/documents/visual studio 2015/Projects/CdnConsoleTutorial/CdnConsoleTutorial/obj/Debug/CdnConsoleApp.exe  
Creating profile CdnConsoleApp.  
Creating endpoint CdnConsoleEndpoint on profile CdnConsoleApp.  
  
Purge CDN endpoint CdnConsoleEndpoint? (Y/N): y  
Purging endpoint. Please wait...  
Done.  
  
Delete CDN endpoint CdnConsoleEndpoint on profile CdnConsoleApp? (Y/N): y  
Deleting endpoint. Please wait...  
Done.  
  
Delete CDN profile CdnConsoleApp? (Y/N): y  
Deleting profile. Please wait...  
Done.  
  
Press Enter to end program.
```

## Next Steps

To see the completed project from this walkthrough, [download the sample](#).

To find additional documentation on the Azure CDN Management Library for .NET, view the [reference on MSDN](#).

Manage your CDN resources with [PowerShell](#).

# Get started with Azure CDN development

7/5/2019 • 9 minutes to read • [Edit Online](#)

You can use the [Azure CDN SDK for Node.js](#) to automate creation and management of CDN profiles and endpoints. This tutorial walks through the creation of a simple Node.js console application that demonstrates several of the available operations. This tutorial is not intended to describe all aspects of the Azure CDN SDK for Node.js in detail.

To complete this tutorial, you should already have [Node.js 4.x.x](#) or higher installed and configured. You can use any text editor you want to create your Node.js application. To write this tutorial, I used [Visual Studio Code](#).

## TIP

The [completed project from this tutorial](#) is available for download on MSDN.

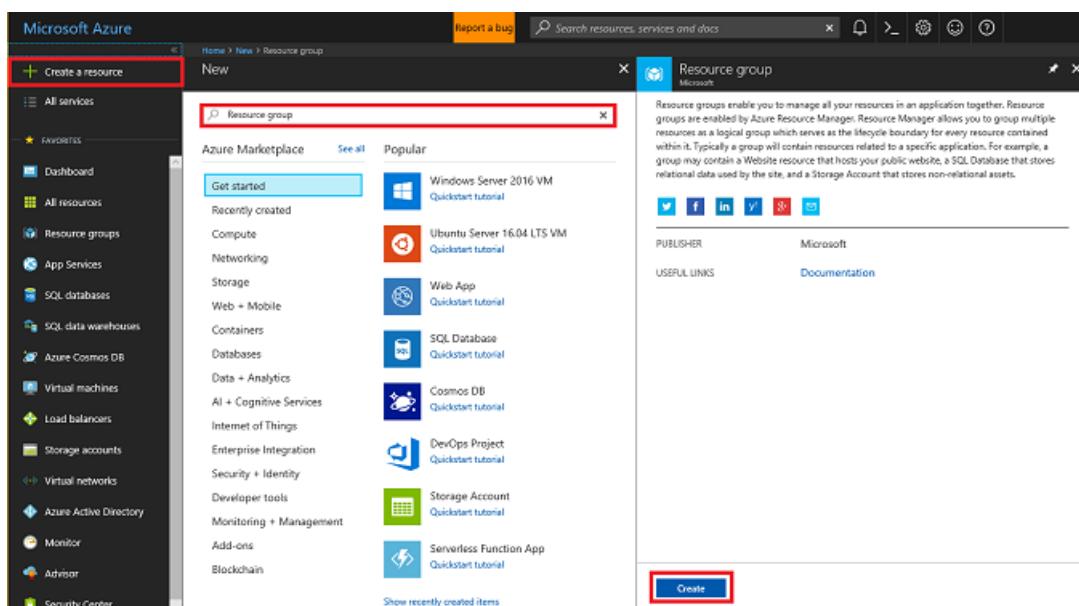
## Prerequisites

Before writing CDN management code, you must do some preparation to enable the code to interact with the Azure Resource Manager. To do this preparation, you need to:

- Create a resource group to contain the CDN profile created in this tutorial
- Configure Azure Active Directory to provide authentication for the application
- Apply permissions to the resource group so that only authorized users from your Azure AD tenant can interact with the CDN profile

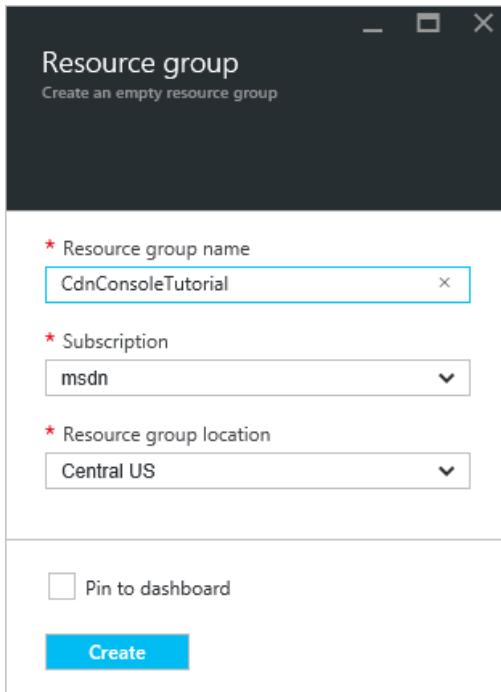
### Creating the resource group

1. Sign in to the [Azure Portal](#).
2. Click **Create a resource**.
3. Search for **Resource group** and in the Resource group pane, click **Create**.

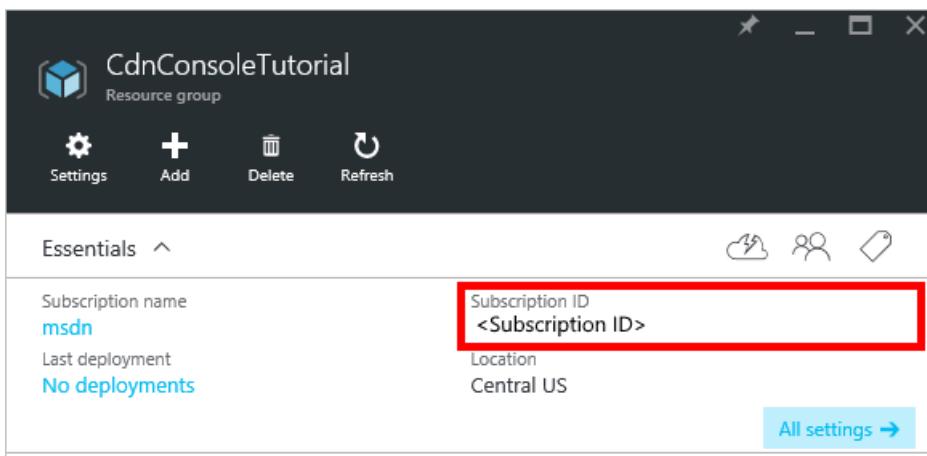


4. Name your resource group *CdnConsoleTutorial*. Select your subscription and choose a location near you. If you wish, you can click the **Pin to dashboard** checkbox to pin the resource group to the dashboard in the

portal. Pinning makes it easier to find later. After you've made your selections, click **Create**.



- After the resource group is created, if you didn't pin it to your dashboard, you can find it by clicking **Browse**, then **Resource Groups**. To open it, click the resource group. Make a note of your **Subscription ID**. We need it later.



### Creating the Azure AD application and applying permissions

There are two approaches to app authentication with Azure Active Directory: Individual users or a service principal. A service principal is similar to a service account in Windows. Instead of granting a particular user permissions to interact with the CDN profiles, permissions are instead granted to the service principal. Service principals are typically used for automated, non-interactive processes. Even though this tutorial is writing an interactive console app, we'll focus on the service principal approach.

Creating a service principal consists of several steps, including creating an Azure Active Directory application. To create it, we're going to [follow this tutorial](#).

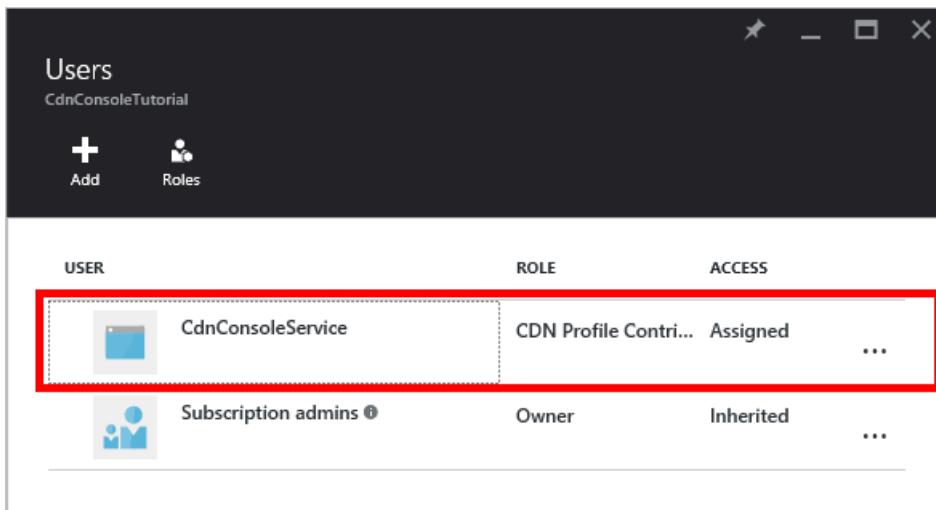
## IMPORTANT

Be sure to follow all the steps in the [linked tutorial](#). It is *important* that you complete it exactly as described. Make sure to note your **tenant ID**, **tenant domain name** (commonly a `.onmicrosoft.com` domain unless you've specified a custom domain), **client ID**, and **client authentication key**, as we need this information later. Be careful to guard your **client ID** and **client authentication key**, as these credentials can be used by anyone to execute operations as the service principal.

When you get to the step named Configure multi-tenant application, select **No**.

When you get to the step **Assign the application to a role**, use the resource group created earlier, `CdnConsoleTutorial`, but instead of the **Reader** role, assign the **CDN Profile Contributor** role. After you assign the application the **CDN Profile Contributor** role on your resource group, return to this tutorial.

Once you've created your service principal and assigned the **CDN Profile Contributor** role, the **Users** blade for your resource group should look similar to the following image.



USER	ROLE	ACCESS
CdnConsoleService	CDN Profile Contr...	Assigned
Subscription admins	Owner	Inherited

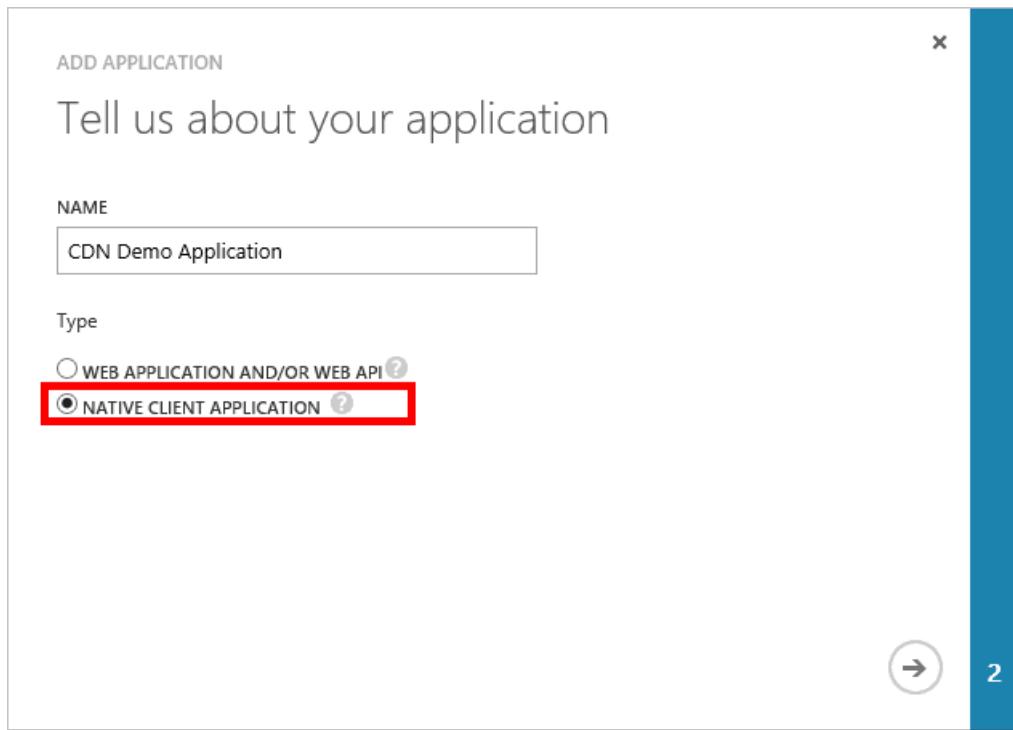
## Interactive user authentication

If, instead of a service principal, you'd rather have interactive individual user authentication, the process is similar to that for a service principal. In fact, you need to follow the same procedure, but make a few minor changes.

## IMPORTANT

Only follow these next steps if you are choosing to use individual user authentication instead of a service principal.

1. When creating your application, instead of **Web Application**, choose **Native application**.



2. On the next page, you are prompted for a **redirect URI**. The URI won't be validated, but remember what you entered. You need it later.
3. There is no need to create a **client authentication key**.
4. Instead of assigning a service principal to the **CDN Profile Contributor** role, we're going to assign individual users or groups. In this example, you can see that I've assigned *CDN Demo User* to the **CDN Profile Contributor** role.

Users		
CdnConsoleTutorial		
	Add	Roles
 CDN Demo User <user>@<tenant>.onmicrosoft.com	CDN Profile Contr... Assigned	...
 Subscription admins ⓘ	Owner	Inherited ...

## Create your project and add NPM dependencies

Now that we've created a resource group for our CDN profiles and given our Azure AD application permission to manage CDN profiles and endpoints within that group, we can start creating our application.

Create a folder to store your application. From a console with the Node.js tools in your current path, set your current location to this new folder and initialize your project by executing:

```
npm init
```

You will then be presented a series of questions to initialize your project. For **entry point**, this tutorial uses *app.js*. You can see my other choices in the following example.

## Command Prompt

```
C:\cdn_node>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (cdn_node)
version: (1.0.0)
description: Azure CDN Node.js tutorial project
entry point: (index.js) app.js
test command:
git repository:
keywords:
author: Cam Soper
license: (ISC) MIT
About to write to C:\cdn_node\package.json:

{
  "name": "cdn_node",
  "version": "1.0.0",
  "description": "Azure CDN Node.js tutorial project",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Cam Soper",
  "license": "MIT"
}

Is this ok? (yes)
C:\cdn_node>
```

Our project is now initialized with a *package.json* file. Our project is going to use some Azure libraries contained in NPM packages. We'll use the Azure Client Runtime for Nodejs (*ms-rest-azure*) and the Azure CDN Client Library for Nodejs (*azure-arm-cdn*). Let's add those to the project as dependencies.

```
npm install --save ms-rest-azure
npm install --save azure-arm-cdn
```

After the packages are done installing, the *package.json* file should look similar to this example (version numbers may differ):

```
{
  "name": "cdn_node",
  "version": "1.0.0",
  "description": "Azure CDN Node.js tutorial project",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Cam Soper",
  "license": "MIT",
  "dependencies": {
    "azure-arm-cdn": "^0.2.1",
    "ms-rest-azure": "^1.14.4"
  }
}
```

Finally, using your text editor, create a blank text file and save it in the root of our project folder as *app.js*. We're

now ready to begin writing code.

## Requires, constants, authentication, and structure

With *app.js* open in our editor, let's get the basic structure of our program written.

1. Add the "requires" for our NPM packages at the top with the following:

```
var msRestAzure = require('ms-rest-azure');
varcdnManagementClient = require('azure-arm-cdn');
```

2. We need to define some constants our methods will use. Add the following. Be sure to replace the placeholders, including the <angle brackets>, with your own values as needed.

```
//Tenant app constants
const clientId = "<YOUR CLIENT ID>";
const clientSecret = "<YOUR CLIENT AUTHENTICATION KEY>"; //Only for service principals
const tenantId = "<YOUR TENANT ID>";

//Application constants
const subscriptionId = "<YOUR SUBSCRIPTION ID>";
const resourceGroupName = "CdnConsoleTutorial";
const resourceLocation = "<YOUR PREFERRED AZURE LOCATION, SUCH AS Central US>";
```

3. Next, we'll instantiate the CDN management client and give it our credentials.

```
var credentials = new msRestAzure.ApplicationTokenCredentials(clientId, tenantId, clientSecret);
var cdnClient = newcdnManagementClient(credentials, subscriptionId);
```

If you are using individual user authentication, these two lines will look slightly different.

### IMPORTANT

Only use this code sample if you are choosing to have individual user authentication instead of a service principal. Be careful to guard your individual user credentials and keep them secret.

```
var credentials = new msRestAzure.UserTokenCredentials(clientId,
    tenantId, '<username>', '<password>', '<redirect URI>');
var cdnClient = newcdnManagementClient(credentials, subscriptionId);
```

Be sure to replace the items in <angle brackets> with the correct information. For <redirect URI>, use the redirect URI you entered when you registered the application in Azure AD.

4. Our Node.js console application is going to take some command-line parameters. Let's validate that at least one parameter was passed.

```

//Collect command-line parameters
var parms = process.argv.slice(2);

//Do we have parameters?
if(parms == null || parms.length == 0)
{
    console.log("Not enough parameters!");
    console.log("Valid commands are list, delete, create, and purge.");
    process.exit(1);
}

```

5. That brings us to the main part of our program, where we branch off to other functions based on what parameters were passed.

```

switch(parms[0].toLowerCase())
{
    case "list":
        cdnList();
        break;

    case "create":
        cdnCreate();
        break;

    case "delete":
        cdnDelete();
        break;

    case "purge":
        cdnPurge();
        break;

    default:
        console.log("Valid commands are list, delete, create, and purge.");
        process.exit(1);
}

```

6. At several places in our program, we'll need to make sure the right number of parameters were passed in and display some help if they don't look correct. Let's create functions to do that.

```

function requireParms(parmCount) {
  if(parms.length < parmCount) {
    usageHelp(parms[0].toLowerCase());
    process.exit(1);
  }
}

function usageHelp(cmd) {
  console.log("Usage for " + cmd + ":");
  switch(cmd)
  {
    case "list":
      console.log("list profiles");
      console.log("list endpoints <profile name>");
      break;

    case "create":
      console.log("create profile <profile name>");
      console.log("create endpoint <profile name> <endpoint name> <origin hostname>");
      break;

    case "delete":
      console.log("delete profile <profile name>");
      console.log("delete endpoint <profile name> <endpoint name>");
      break;

    case "purge":
      console.log("purge <profile name> <endpoint name> <path>");
      break;

    default:
      console.log("Invalid command.");
  }
}

```

- Finally, the functions we'll be using on the CDN management client are asynchronous, so they need a method to call back when they're done. Let's make one that can display the output from the CDN management client (if any) and exit the program gracefully.

```

function callback(err, result, request, response) {
  if (err) {
    console.log(err);
    process.exit(1);
  } else {
    console.log((result == null) ? "Done!" : result);
    process.exit(0);
  }
}

```

Now that the basic structure of our program is written, we should create the functions called based on our parameters.

## List CDN profiles and endpoints

Let's start with code to list our existing profiles and endpoints. My code comments provide the expected syntax so we know where each parameter goes.

```
// list profiles
// list endpoints <profile name>
function cdnList(){
    requireParms(2);
    switch(parms[1].toLowerCase())
    {
        case "profiles":
            console.log("Listing profiles...");
            cdnClient.profiles.listByResourceGroup(resourceGroupName, callback);
            break;

        case "endpoints":
            requireParms(3);
            console.log("Listing endpoints...");
            cdnClient.endpoints.listByProfile(parms[2], resourceGroupName, callback);
            break;

        default:
            console.log("Invalid parameter.");
            process.exit(1);
    }
}
```

## Create CDN profiles and endpoints

Next, we'll write the functions to create profiles and endpoints.

```

function cdnCreate() {
    requireParms(2);
    switch(parms[1].toLowerCase())
    {
        case "profile":
            cdnCreateProfile();
            break;

        case "endpoint":
            cdnCreateEndpoint();
            break;

        default:
            console.log("Invalid parameter.");
            process.exit(1);
    }
}

// create profile <profile name>
function cdnCreateProfile() {
    requireParms(3);
    console.log("Creating profile...");
    var standardCreateParameters = {
        location: resourceLocation,
        sku: {
            name: 'Standard_Verizon'
        }
    };

    cdnClient.profiles.create(parms[2], standardCreateParameters, resourceGroupName, callback);
}

// create endpoint <profile name> <endpoint name> <origin hostname>
function cdnCreateEndpoint() {
    requireParms(5);
    console.log("Creating endpoint...");
    var endpointProperties = {
        location: resourceLocation,
        origins: [
            {
                name: parms[4],
                hostName: parms[4]
            }
        ]
    };

    cdnClient.endpoints.create(parms[3], endpointProperties, parms[2], resourceGroupName, callback);
}

```

## Purge an endpoint

Assuming the endpoint has been created, one common task that we might want to perform in our program is purging content in our endpoint.

```

// purge <profile name> <endpoint name> <path>
function cdnPurge() {
    requireParms(4);
    console.log("Purging endpoint...");
    var purgeContentPaths = [ parms[3] ];
    cdnClient.endpoints.purgeContent(parms[2], parms[1], resourceGroupName, purgeContentPaths, callback);
}

```

## Delete CDN profiles and endpoints

The last function we will include deletes endpoints and profiles.

```
function cdnDelete() {
    requireParms(2);
    switch(parms[1].toLowerCase())
    {
        // delete profile <profile name>
        case "profile":
            requireParms(3);
            console.log("Deleting profile...");
            cdnClient.profiles.deleteIfExists(parms[2], resourceGroupName, callback);
            break;

        // delete endpoint <profile name> <endpoint name>
        case "endpoint":
            requireParms(4);
            console.log("Deleting endpoint...");
            cdnClient.endpoints.deleteIfExists(parms[3], parms[2], resourceGroupName, callback);
            break;

        default:
            console.log("Invalid parameter.");
            process.exit(1);
    }
}
```

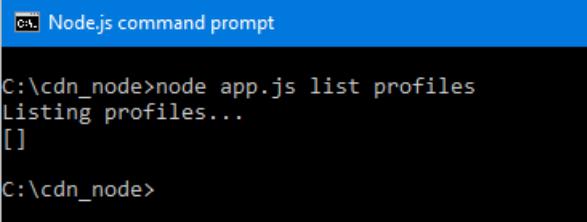
## Running the program

We can now execute our Node.js program using our favorite debugger or at the console.

### TIP

If you're using Visual Studio Code as your debugger, you'll need to set up your environment to pass in the command-line parameters. Visual Studio Code does this in the **launch.json** file. Look for a property named **args** and add an array of string values for your parameters, so that it looks similar to this: `"args": ["list", "profiles"]`.

Let's start by listing our profiles.



```
Node.js command prompt
C:\cdn_node>node app.js list profiles
Listing profiles...
[]
C:\cdn_node>
```

We got back an empty array. Since we don't have any profiles in our resource group, that's expected. Let's create a profile now.

```
Node.js command prompt

C:\cdn_node>node app.js create profile MyNodeCdnProfile
Creating profile...
{ id: '/subscriptions/<subscription ID>/resourcegroups/CdnConsoleTutorial/providers/Microsoft.Cdn/profiles/MyNodeCdnProfile',
  name: 'MyNodeCdnProfile',
  type: 'Microsoft.Cdn/profiles',
  location: 'CentralUs',
  tags: {},
  sku: { name: 'Standard_Verizon' },
  resourceState: 'Active',
  provisioningState: 'Succeeded' }

C:\cdn_node>
```

Now, let's add an endpoint.

```
Node.js command prompt

C:\cdn_node>node app.js create endpoint MyNodeCdnProfile cdnnodedemo www.contoso.com
Creating endpoint...
{ id: '/subscriptions/<subscription ID>/resourcegroups/CdnConsoleTutorial/providers/Microsoft.Cdn/profiles/MyNodeCdnProfile/endpoints/cdnnodedemo',
  name: 'cdnnodedemo',
  type: 'Microsoft.Cdn/profiles/endpoints',
  location: 'CentralUs',
  tags: {},
  hostName: 'cdnnodedemo.azureedge.net',
  contentTypesToCompress: [],
  isCompressionEnabled: false,
  isHttpAllowed: true,
  isHttpsAllowed: true,
  queryStringCachingBehavior: 'IgnoreQueryString',
  origins: [ { name: 'www.contoso.com', hostName: 'www.contoso.com' } ],
  resourceState: 'Running',
  provisioningState: 'Succeeded' }

C:\cdn_node>
```

Finally, let's delete our profile.

```
Node.js command prompt

C:\cdn_node>node app.js delete profile MyNodeCdnProfile
Deleting profile...
Done!

C:\cdn_node>
```

## Next Steps

To see the completed project from this walkthrough, [download the sample](#).

To see the reference for the Azure CDN SDK for Node.js, view the [reference](#).

To find additional documentation on the Azure SDK for Node.js, view the [full reference](#).

Manage your CDN resources with [PowerShell](#).

# Troubleshooting Azure CDN endpoints that return a 404 status code

11/14/2019 • 6 minutes to read • [Edit Online](#)

This article enables you to troubleshoot issues with Azure Content Delivery Network (CDN) endpoints that return 404 HTTP response status codes.

If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and the Stack Overflow forums](#). Alternatively, you can also file an Azure support incident. Go to the [Azure Support site](#) and select **Get Support**.

## Symptom

You've created a CDN profile and an endpoint, but your content doesn't seem to be available on the CDN. Users who attempt to access your content via the CDN URL receive an HTTP 404 status code.

## Cause

There are several possible causes, including:

- The file's origin isn't visible to the CDN.
- The endpoint is misconfigured, causing the CDN to look in the wrong place.
- The host is rejecting the host header from the CDN.
- The endpoint hasn't had time to propagate throughout the CDN.

## Troubleshooting steps

### IMPORTANT

After creating a CDN endpoint, it will not immediately be available for use, as it takes time for the registration to propagate through the CDN:

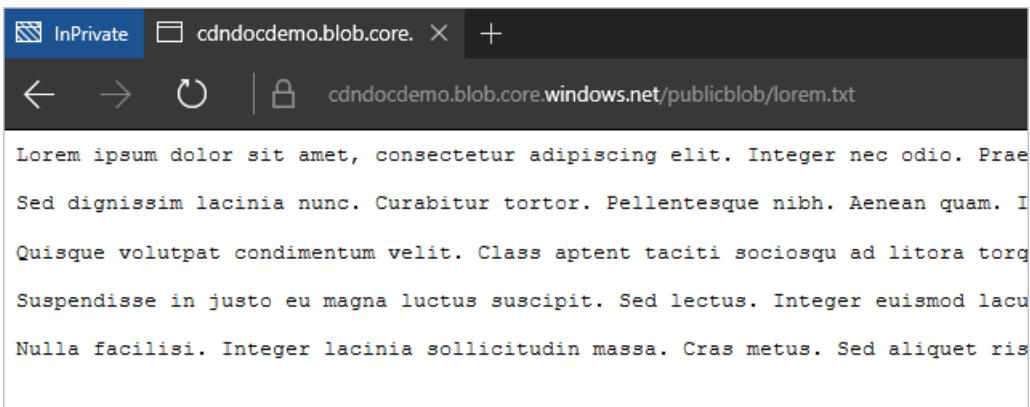
- For **Azure CDN Standard from Microsoft** profiles, propagation usually completes in ten minutes.
- For **Azure CDN Standard from Akamai** profiles, propagation usually completes within one minute.
- For **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles, propagation usually completes within 90 minutes.

If you complete the steps in this document and you're still getting 404 responses, consider waiting a few hours to check again before opening a support ticket.

### Check the origin file

First, verify that the file to cache is available on the origin server and is publicly accessible on the internet. The quickest way to do that is to open a browser in a private or incognito session and browse directly to the file. Type or paste the URL into the address box and verify that it results in the file you expect. For example, suppose you have a file in an Azure Storage account, accessible at

<https://cdndocdemo.blob.core.windows.net/publicblob/lorem.txt>. If you can successfully load this file's contents, it passes the test.



### WARNING

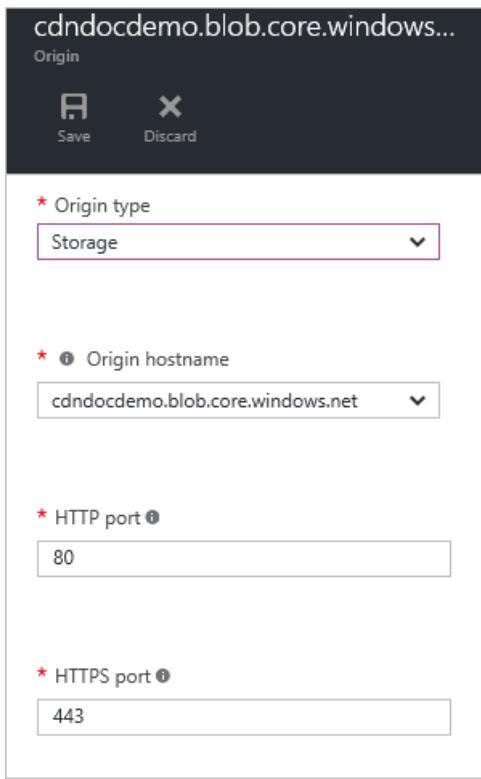
While this is the quickest and easiest way to verify your file is publicly available, some network configurations in your organization could make it appear that a file is publicly available when it is, in fact, only visible to users of your network (even if it's hosted in Azure). To ensure that this isn't the case, test the file with an external browser, such as a mobile device that is not connected to your organization's network, or a virtual machine in Azure.

### Check the origin settings

After you've verified the file is publicly available on the internet, verify your origin settings. In the [Azure Portal](#), browse to your CDN profile and select the endpoint you're troubleshooting. From the resulting **Endpoint** page, select the origin.

HOSTNAME	HTTP PORT	HTTPS PORT
cdndocdemo.blob.core.windows.net	80	443

The **Origin** page appears.



#### Origin type and hostname

Verify that the values of the **Origin type** and **Origin hostname** are correct. In this example, <https://cdndocdemo.blob.core.windows.net/publicblob/lorem.txt>, the hostname portion of the URL is *cdndocdemo.blob.core.windows.net*, which is correct. Because Azure Storage, Web App, and Cloud Service origins use a drop-down list value for the **Origin hostname** field, incorrect spellings aren't an issue. However, if you use a custom origin, ensure that your hostname is spelled correctly.

#### HTTP and HTTPS ports

Check your **HTTP** and **HTTPS ports**. In most cases, 80 and 443 are correct, and you will require no changes. However, if the origin server is listening on a different port, that will need to be represented here. If you're not sure, view the URL for your origin file. The HTTP and HTTPS specifications use ports 80 and 443 as the defaults. In the example URL, <https://cdndocdemo.blob.core.windows.net/publicblob/lorem.txt>, a port is not specified, so the default of 443 is assumed and the settings are correct.

However, suppose the URL for the origin file that you tested earlier is <http://www.contoso.com:8080/file.txt>. Note the :8080 portion at the end of the hostname segment. That number instructs the browser to use port 8080 to connect to the web server at www.contoso.com, therefore you'll need to enter *8080* in the **HTTP port** field. It's important to note that these port settings affect only what port the endpoint uses to retrieve information from the origin.

#### NOTE

**Azure CDN Standard from Akamai** endpoints do not allow the full TCP port range for origins. For a list of origin ports that are not allowed, see [Azure CDN from Akamai Allowed Origin Ports](#).

#### Check the endpoint settings

On the **Endpoint** page, select the **Configure** button.

The screenshot shows the Azure CDN endpoint configuration interface. At the top, there's a header with the endpoint name "cdndocdemo.azureedge.net" and several action buttons: "Custom domain" (with a plus icon), "Configure" (which is highlighted with a red box), "Stop", "Purge", "Load", and "Delete". Below the header is a section titled "Origins" with a table:

HOSTNAME	HTTP PORT	HTTPS PORT
cdndocdemo.blob.core.windows.net	80	443

The CDN endpoint **Configure** page appears.

The screenshot shows the "Configure" page with the following settings:

- Compression:** A toggle switch is set to "Off".
- Query string caching behavior:** A dropdown menu is set to "Ignore query strings".
- Protocols:** Both "HTTP" and "HTTPS" are checked.
- Origin host header:** The input field contains "cdndocdemo.blob.core.windows.net".
- Origin path:** The input field contains "/Path".

#### Protocols

For **Protocols**, verify that the protocol being used by the clients is selected. Because the same protocol used by the client is the one used to access the origin, it's important to have the origin ports configured correctly in the previous section. The CDN endpoint listens only on the default HTTP and HTTPS ports (80 and 443), regardless of the origin ports.

Let's return to our hypothetical example with `http://www.contoso.com:8080/file.txt`. As you'll remember, Contoso specified `8080` as their HTTP port, but let's also assume they specified `44300` as their HTTPS port. If they created an endpoint named `contoso`, their CDN endpoint hostname would be `contoso.azureedge.net`. A request for `http://contoso.azureedge.net/file.txt` is an HTTP request, so the endpoint would use HTTP on port `8080` to retrieve it from the origin. A secure request over HTTPS, `https://contoso.azureedge.net/file.txt`, would cause the endpoint to use HTTPS on port `44300` when retrieving the file from the origin.

#### Origin host header

The **Origin host header** is the host header value sent to the origin with each request. In most cases, this should be the same as the **Origin hostname** we verified earlier. An incorrect value in this field won't generally cause 404 statuses, but is likely to cause other 4xx statuses, depending on what the origin expects.

#### Origin path

Lastly, we should verify our **Origin path**. By default this is blank. You should only use this field if you want to narrow the scope of the origin-hosted resources you want to make available on the CDN.

In the example endpoint, we wanted all resources on the storage account to be available, so **Origin path** was left blank. This means that a request to <https://cdndocdemo.azureedge.net/publicblob/lorem.txt> results in a connection from the endpoint to cdndocdemo.core.windows.net that requests */publicblob/lorem.txt*. Likewise, a request for <https://cdndocdemo.azureedge.net/donotcache/status.png> results in the endpoint requesting */donotcache/status.png* from the origin.

But what if you don't want to use the CDN for every path on your origin? Say you only wanted to expose the *publicblob* path. If we enter */publicblob* in the **Origin path** field, that will cause the endpoint to insert */publicblob* before every request being made to the origin. This means that the request for <https://cdndocdemo.azureedge.net/publicblob/lorem.txt> will now actually take the request portion of the URL, */publicblob/lorem.txt*, and append */publicblob* to the beginning. This results in a request for */publicblob/publicblob/lorem.txt* from the origin. If that path doesn't resolve to an actual file, the origin will return a 404 status. The correct URL to retrieve *lorem.txt* in this example would actually be <https://cdndocdemo.azureedge.net/lorem.txt>. Note that we don't include the */publicblob* path at all, because the request portion of the URL is */lorem.txt* and the endpoint adds */publicblob*, resulting in */publicblob/lorem.txt* being the request passed to the origin.

# Troubleshooting CDN file compression

7/5/2019 • 3 minutes to read • [Edit Online](#)

This article helps you troubleshoot issues with [CDN file compression](#).

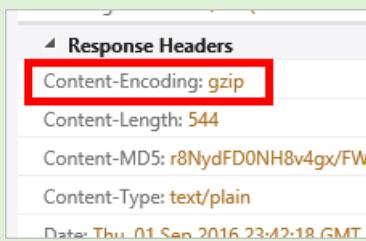
If you need more help at any point in this article, you can contact the Azure experts on [the MSDN Azure and the Stack Overflow forums](#). Alternatively, you can also file an Azure support incident. Go to the [Azure Support site](#) and click **Get Support**.

## Symptom

Compression for your endpoint is enabled, but files are being returned uncompressed.

### TIP

To check whether your files are being returned compressed, you need to use a tool like [Fiddler](#) or your browser's [developer tools](#). Check the HTTP response headers returned with your cached CDN content. If there is a header named `Content-Encoding` with a value of **gzip**, **bzip2**, or **deflate**, your content is compressed.



## Cause

There are several possible causes, including:

- The requested content is not eligible for compression.
- Compression is not enabled for the requested file type.
- The HTTP request did not include a header requesting a valid compression type.

## Troubleshooting steps

### TIP

As with deploying new endpoints, CDN configuration changes take some time to propagate through the network. Usually, changes are applied within 90 minutes. If this is the first time you've set up compression for your CDN endpoint, you should consider waiting 1-2 hours to be sure the compression settings have propagated to the POPs.

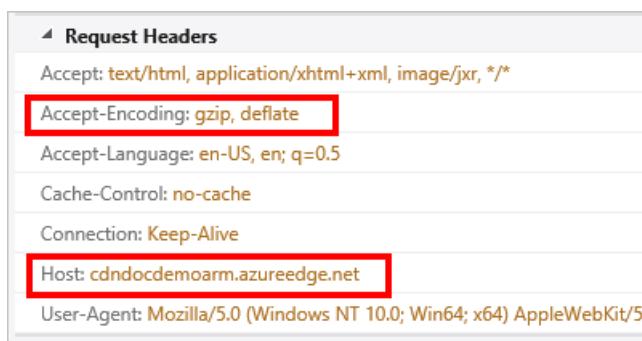
### Verify the request

First, we should do a quick sanity check on the request. You can use your browser's [developer tools](#) to view the requests being made.

- Verify the request is being sent to your endpoint URL, `<endpointname>.azureedge.net`, and not your origin.
- Verify the request contains an **Accept-Encoding** header, and the value for that header contains **gzip**, **deflate**, or **bzip2**.

#### NOTE

Azure CDN from Akamai profiles only support **gzip** encoding.



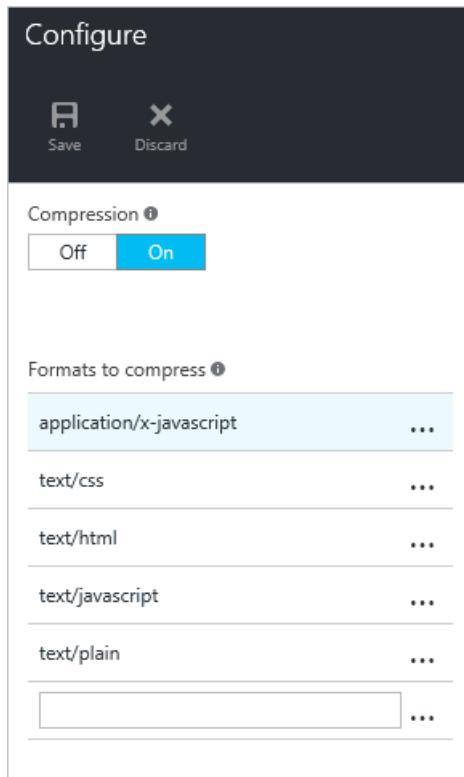
### Verify compression settings (standard CDN profiles)

#### NOTE

This step applies only if your CDN profile is an **Azure CDN Standard from Microsoft**, **Azure CDN Standard from Verizon**, or **Azure CDN Standard from Akamai** profile.

Navigate to your endpoint in the [Azure portal](#) and click the **Configure** button.

- Verify compression is enabled.
- Verify the MIME type for the content to be compressed is included in the list of compressed formats.



### Verify compression settings (Premium CDN profiles)

#### NOTE

This step applies only if your CDN profile is an **Azure CDN Premium from Verizon** profile.

Navigate to your endpoint in the [Azure portal](#) and click the **Manage** button. The supplemental portal will open.

Hover over the **HTTP Large** tab, then hover over the **Cache Settings** flyout. Click **Compression**.

- Verify compression is enabled.
- Verify the **File Types** list contains a comma-separated list (no spaces) of MIME types.
- Verify the MIME type for the content to be compressed is included in the list of compressed formats.

Compression Disabled  
 Compression Enabled

File Types:  
text/plain, text/html, text/css, application/x-javascript, text/javascript

Last Updated: 4/20/2016 8:45:59 PM  
Expected Complete Time (1 Hour): 4/20/2016 9:45:59 PM

### Verify the content is cached (Verizon CDN profiles)

#### NOTE

This step applies only if your CDN profile is an **Azure CDN Standard from Verizon** or **Azure CDN Premium from Verizon** profile.

Using your browser's developer tools, check the response headers to ensure the file is cached in the region where it is being requested.

- Check the **Server** response header. The header should have the format **Platform (POP/Server ID)**, as seen in the following example.
- Check the **X-Cache** response header. The header should read **HIT**.

Response Headers
Accept-Ranges: bytes
Content-Length: 2206
Content-MD5: QZF4+XrFhV9d5rJ4ZZdeIQ==
Content-Type: text/plain
Date: Thu, 21 Apr 2016 01:24:45 GMT
Etag: 0x8D36982CD751E5D
Last-Modified: Thu, 21 Apr 2016 01:18:10 GMT
Server: ECAcc (dfw/565B)
X-Cache: HIT
x-ms-blob-type: BlockBlob

### Verify the file meets the size requirements (Verizon CDN profiles)

#### NOTE

This step applies only if your CDN profile is an **Azure CDN Standard from Verizon** or **Azure CDN Premium from Verizon** profile.

To be eligible for compression, a file must meet the following size requirements:

- Larger than 128 bytes.
- Smaller than 1 MB.

#### **Check the request at the origin server for a `Via` header**

The **Via** HTTP header indicates to the web server that the request is being passed by a proxy server. Microsoft IIS web servers by default do not compress responses when the request contains a **Via** header. To override this behavior, perform the following:

- **IIS 6:** Set `HcNoCompressionForProxies="FALSE"` in the IIS Metabase properties
- **IIS 7 and up:** Set both `noCompressionForHttp10` and `noCompressionForProxies` to False in the server configuration

# Allowed certificate authorities for enabling custom HTTPS on Azure CDN

2/27/2020 • 2 minutes to read • [Edit Online](#)

You must meet specific certificate requirements when you [enable the HTTPS feature by using your own certificate](#) for an Azure Content Delivery Network (CDN) custom domain. The **Azure CDN Standard from Microsoft** profile requires a certificate from one of the approved certificate authorities (CA) in the following list. If a certificate from an unapproved CA or if a self-signed certificate is used, the request is rejected. **Azure CDN Standard from Verizon** and **Azure CDN Premium from Verizon** profiles accept any valid certificate from any valid CA.

## NOTE

The option of using your own certificate to enable the custom domain HTTPS feature is *not* available for **Azure CDN Standard from Akamai** profiles.

The following CAs are allowed when you create your own certificate:

- AddTrust External CA Root
- AlphaSSL Root CA
- AME Infra CA 01
- AME Infra CA 02
- Ameroot
- APCA-DM3P
- Autopilot Root CA
- Baltimore CyberTrust Root
- Class 3 Public Primary Certification Authority
- COMODO RSA Certification Authority
- COMODO RSA Domain Validation Secure Server CA
- D-TRUST Root Class 3 CA 2 2009
- DigiCert Cloud Services CA-1
- DigiCert Global Root CA
- DigiCert High Assurance CA-3
- DigiCert High Assurance EV Root CA
- DigiCert SHA2 Extended Validation Server CA
- DigiCert SHA2 High Assurance Server CA
- DigiCert SHA2 Secure Server CA
- DST Root CA X3
- D-trust Root Class 3 CA 2 2009
- Encryption Everywhere DV TLS CA
- Entrust Root Certification Authority
- Entrust Root Certification Authority - G2
- Entrust.net Certification Authority (2048)
- GeoTrust Global CA
- GeoTrust Primary Certification Authority
- GeoTrust Primary Certification Authority - G2

- Geotrust RSA CA 2018
- GlobalSign
- GlobalSign Extended Validation CA - SHA256 - G2
- GlobalSign Organization Validation CA - G2
- GlobalSign Root CA
- Go Daddy Root Certificate Authority - G2
- Go Daddy Secure Certificate Authority - G2
- QuoVadis Root CA2 G3
- RapidSSL RSA CA 2018
- Symantec Class 3 EV SSL CA - G3
- Symantec Class 3 Secure Server CA - G4
- Symantec Enterprise Mobile Root for Microsoft
- Thawte Primary Root CA
- Thawte Primary Root CA - G2
- Thawte Primary Root CA - G3
- Thawte RSA CA 2018
- Thawte Timestamping CA
- TrustAsia TLS RSA CA
- VeriSign Class 3 Extended Validation SSL CA
- VeriSign Class 3 Extended Validation SSL SGC CA
- VeriSign Class 3 Public Primary Certification Authority - G5
- VeriSign International Server CA - Class 3
- VeriSign Time Stamping Service Root
- VeriSign Universal Root Certification Authority

# Standard rules engine reference for Azure CDN

11/18/2019 • 3 minutes to read • [Edit Online](#)

In the [Standard rules engine](#) for Azure Content Delivery Network (Azure CDN), a rule consists of one or more match conditions and an action. This article provides detailed descriptions of the match conditions and features that are available in the Standard rules engine for Azure CDN.

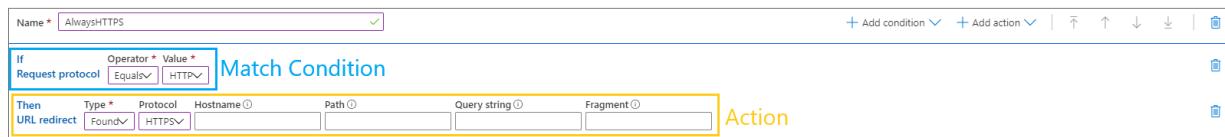
The rules engine is designed to be the final authority on how specific types of requests are processed by Standard Azure CDN.

## Common uses for the rules:

- Override or define a custom cache policy.
- Redirect requests.
- Modify HTTP request and response headers.

## Terminology

To define a rule in the rules engine, set [match conditions](#) and [actions](#):



Each rule can have up to four match conditions and three actions. Each Azure CDN endpoint can have up to five rules.

Included in the current five-rule limit for an Azure CDN endpoint is a default *global rule*. The global rule doesn't have match conditions, and actions that are defined in a global rule always trigger.

## Syntax

How special characters are treated in a rule varies based on how different match conditions and actions handle text values. A match condition or action can interpret text in one of the following ways:

- [Literal values](#)
- [Wildcard values](#)

### Literal values

Text that's interpreted as a literal value treats all special characters *except the % symbol* as part of the value that must be matched in a rule. For example, a literal match condition set to `'*''` is satisfied only when the exact value `'*''` is found.

A percent sign is used to indicate URL encoding (for example, `%20`).

### Wildcard values

Text that's interpreted as a wildcard value assigns additional meaning to special characters. The following table describes how specific special characters are interpreted in the Standard rules engine:

CHARACTER	DESCRIPTION
\	A backslash is used to escape any of the characters specified in this table. A backslash must be specified directly before the special character that should be escaped. For example, the following syntax escapes an asterisk: <code>\*</code>
%	A percent sign is used to indicate URL encoding (for example, <code>%20</code> ).
*	An asterisk is a wildcard that represents one or more characters.
space	A space character indicates that a match condition can be satisfied by either of the specified values or patterns.
single quotation marks	<p>A single quotation mark doesn't have special meaning. However, a set of single quotation marks indicates that a value should be treated as a literal value. Single quotation marks can be used in the following ways:</p> <ul style="list-style-type: none"> <li>To allow a match condition to be satisfied whenever the specified value matches any portion of the comparison value. For example, <code>'ma'</code> would match any of the following strings: <ul style="list-style-type: none"> <li><code>/business/marathon/asset.htm</code></li> <li><code>map.gif</code></li> <li><code>/business/template.map</code></li> </ul> </li> <li>To allow a special character to be specified as a literal character. For example, you can specify a literal space character by enclosing a space character in a set of single quotation marks (<code>' '</code> or <code>'&lt;sample value&gt;'</code>).</li> <li>To allow a blank value to be specified. Specify a blank value by specifying a set of single quotation marks ('').</li> </ul> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>If the specified value doesn't contain a wildcard, the value is automatically considered a literal value. You don't need to specify a set of single quotation marks for a literal value.</li> <li>If a backslash isn't used to escape another character in this table, the backslash is ignored when it's specified in a set of single quotation marks.</li> <li>Another way to specify a special character as a literal character is to escape it by using a backslash (<code>\</code>).</li> </ul>

## Next steps

- [Match conditions in the Standard rules engine](#)
- [Actions in the Standard rules engine](#)
- [Enforce HTTPS by using the Standard rules engine](#)
- [Azure CDN overview](#)

# Match conditions in the Standard rules engine for Azure CDN

12/6/2019 • 5 minutes to read • [Edit Online](#)

In the [Standard rules engine](#) for Azure Content Delivery Network (Azure CDN), a rule consists of one or more match conditions and an action. This article provides detailed descriptions of the match conditions you can use in the Standard rules engine for Azure CDN.

The first part of a rule is a match condition or set of match conditions. In the Standard rules engine for Azure CDN, each rule can have up to four match conditions. A match condition identifies specific types of requests for which defined actions are performed. If you use multiple match conditions, the match conditions are grouped together by using AND logic.

For example, you can use a match condition to:

- Filter requests based on a specific IP address, country, or region.
- Filter requests by header information.
- Filter requests from mobile devices or desktop devices.

## Match conditions

The following match conditions are available to use in the Standard rules engine for Azure CDN.

### Device type

Identifies requests made from a mobile device or desktop device.

#### Required fields

OPERATOR	SUPPORTED VALUES
Equals, Not equals	Mobile, Desktop

### HTTP version

Identifies requests based on the HTTP version of the request.

#### Required fields

OPERATOR	SUPPORTED VALUES
Equals, Not equals	2.0, 1.1, 1.0, 0.9, All

### Request cookies

Identifies requests based on cookie information in the incoming request.

#### Required fields

COOKIE NAME	OPERATOR	COOKIE VALUE	CASE TRANSFORM
String	<a href="#">Standard operator list</a>	String, Int	No transform, to uppercase, to lowercase

#### Key information

- You can't use wildcard values (including asterisks (\*)) when you specify a cookie name; you must use an exact cookie name.
- You can specify only a single cookie name per instance of this match condition.
- Cookie name comparisons are case-insensitive.
- To specify multiple cookie values, use a single space between each cookie value.
- Cookie values can take advantage of wildcard values.
- If a wildcard value hasn't been specified, only an exact match satisfies this match condition. For example, "Value" will match "Value" but not "Value1".

## Post argument

Identifies requests based on arguments defined for the POST request method that's used in the request.

### Required fields

ARGUMENT NAME	OPERATOR	ARGUMENT VALUE	CASE TRANSFORM
String	Standard operator list	String, Int	No transform, to uppercase, to lowercase

## Query string

Identifies requests that contain a specific query string parameter. This parameter is set to a value that matches a specific pattern. Query string parameters (for example, **parameter=value**) in the request URL determine whether this condition is met. This match condition identifies a query string parameter by its name and accepts one or more values for the parameter value.

### Required fields

OPERATOR	QUERY STRING	CASE TRANSFORM
Standard operator list	String, Int	No transform, to uppercase, to lowercase

## Remote address

Identifies requests based on the requester's location or IP address.

### Required fields

OPERATOR	SUPPORTED VALUES
Any	N/A
Geo Match	Country code
IP Match	IP address (space-separated)
Not Any	N/A
Not Geo Match	Country code
Not IP Match	IP address (space-separated)

### Key information

- Use CIDR notation.
- To specify multiple IP addresses and IP address blocks, use a single space between the values:
  - **IPv4 example:** 1.2.3.4 10.20.30.40 matches any requests that arrive from either address 1.2.3.4 or 10.20.30.40.

10.20.30.40.

- **IPv6 example:** 1:2:3:4:5:6:7:8 10:20:30:40:50:60:70:80 matches any requests that arrive from either address 1:2:3:4:5:6:7:8 or 10:20:30:40:50:60:70:80.
- The syntax for an IP address block is the base IP address followed by a forward slash and the prefix size. For example:
  - **IPv4 example:** 5.5.5.64/26 matches any requests that arrive from addresses 5.5.5.64 through 5.5.5.127.
  - **IPv6 example:** 1:2:3:/48 matches any requests that arrive from addresses 1:2:3:0:0:0:0 through 1:2:3:ffff:ffff:ffff:ffff:ffff.

## Request body

Identifies requests based on specific text that appears in the body of the request.

### Required fields

OPERATOR	REQUEST BODY	CASE TRANSFORM
<a href="#">Standard operator list</a>	String, Int	No transform, to uppercase, to lowercase

## Request header

Identifies requests that use a specific header in the request.

### Required fields

HEADER NAME	OPERATOR	HEADER VALUE	CASE TRANSFORM
String	<a href="#">Standard operator list</a>	String, Int	No transform, to uppercase, to lowercase

## Request method

Identifies requests that use the specified request method.

### Required fields

OPERATOR	SUPPORTED VALUES
Equals, Not equals	GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE

### Key information

- Only the GET request method can generate cached content in Azure CDN. All other request methods are proxied through the network.

## Request protocol

Identifies requests that use the specified protocol used.

### Required fields

OPERATOR	SUPPORTED VALUES
Equals, Not equals	HTTP, HTTPS

## Request URL

Identifies requests that match the specified URL.

### Required fields

OPERATOR	REQUEST URL	CASE TRANSFORM
Standard operator list	String, Int	No transform, to uppercase, to lowercase

#### Key information

- When you use this rule condition, be sure to include protocol information. For example: `https://www.<yourdomain>.com.`

#### URL file extension

Identifies requests that include the specified file extension in the file name in the requesting URL.

#### Required fields

OPERATOR	EXTENSION	CASE TRANSFORM
Standard operator list	String, Int	No transform, to uppercase, to lowercase

#### Key information

- For extension, don't include a leading period; for example, use `html` instead of `.html`.

#### URL file name

Identifies requests that include the specified file name in the requesting URL.

#### Required fields

OPERATOR	FILE NAME	CASE TRANSFORM
Standard operator list	String, Int	No transform, to uppercase, to lowercase

#### Key information

- To specify multiple file names, separate each file name with a single space.

#### URL path

Identifies requests that include the specified path in the requesting URL.

#### Required fields

OPERATOR	VALUE	CASE TRANSFORM
Standard operator list	String, Int	No transform, to uppercase, to lowercase

#### Key information

- A file name value can take advantage of wildcard values. For example, each file name pattern can consist of one or more asterisks (\*), where each asterisk matches a sequence of one or more characters.

## Reference for rules engine match conditions

### Standard operator list

For rules that accept values from the standard operator list, the following operators are valid:

- Any
- Equals
- Contains

- Begins with
- Ends with
- Less than
- Less than or equals
- Greater than
- Greater than or equals
- Not any
- Not contains
- Not begins with
- Not ends with
- Not less than
- Not less than or equals
- Not greater than
- Not greater than or equals

For numeric operators like *Less than* and *Greater than or equals*, the comparison used is based on length. In this case, the value in the match condition should be an integer that's equal to the length you want to compare.

## Next steps

- [Azure CDN overview](#)
- [Standard rules engine reference](#)
- [Actions in the Standard rules engine](#)
- [Enforce HTTPS by using the Standard rules engine](#)

# Actions in the Standard rules engine for Azure CDN

11/18/2019 • 4 minutes to read • [Edit Online](#)

In the [Standard rules engine](#) for Azure Content Delivery Network (Azure CDN), a rule consists of one or more match conditions and an action. This article provides detailed descriptions of the actions you can use in the Standard rules engine for Azure CDN.

The second part of a rule is an action. An action defines the behavior that's applied to the request type that a match condition or set of match conditions identifies.

## Actions

The following actions are available to use in the Standard rules engine for Azure CDN.

### Cache expiration

Use this action to overwrite the time to live (TTL) value of the endpoint for requests that the rules match conditions specify.

#### Required fields

CACHE BEHAVIOR	DESCRIPTION
Bypass cache	When this option is selected and the rule matches, the content is not cached.
Override	When this option is selected and the rule matches, the TTL value returned from your origin is overwritten with the value specified in the action.
Set if missing	When this option is selected and the rule matches, if no TTL value was returned from your origin, the rule sets the TTL to the value specified in the action.

#### Additional fields

DAYS	HOURS	MINUTES	SECONDS
Int	Int	Int	Int

### Cache key query string

Use this action to modify the cache key based on query strings.

#### Required fields

BEHAVIOR	DESCRIPTION
Include	When this option is selected and the rule matches, query strings specified in the parameters are included when the cache key is generated.
Cache every unique URL	When this option is selected and the rule matches, each unique URL has its own cache key.

BEHAVIOR	DESCRIPTION
Exclude	When this option is selected and the rule matches, query strings specified in the parameters are excluded when the cache key is generated.
Ignore query strings	When this option is selected and the rule matches, query strings aren't considered when the cache key is generated.

## Modify request header

Use this action to modify headers that are present in requests sent to your origin.

### Required fields

ACTION	HTTP HEADER NAME	VALUE
Append	When this option is selected and the rule matches, the header specified in <b>Header name</b> is added to the request with the specified value. If the header is already present, the value is appended to the existing value.	String
Overwrite	When this option is selected and the rule matches, the header specified in <b>Header name</b> is added to the request with the specified value. If the header is already present, the specified value overwrites the existing value.	String
Delete	When this option is selected, the rule matches, and the header specified in the rule is present, the header is deleted from the request.	String

## Modify response header

Use this action to modify headers that are present in responses returned to your clients.

### Required fields

ACTION	HTTP HEADER NAME	VALUE
Append	When this option is selected and the rule matches, the header specified in <b>Header name</b> is added to the response by using the specified <b>Value</b> . If the header is already present, <b>Value</b> is appended to the existing value.	String
Overwrite	When this option is selected and the rule matches, the header specified in <b>Header name</b> is added to the response by using the specified <b>Value</b> . If the header is already present, <b>Value</b> overwrites the existing value.	String

Action	HTTP Header Name	Value
Delete	When this option is selected, the rule matches, and the header specified in the rule is present, the header is deleted from the response.	String

## URL redirect

Use this action to redirect clients to a new URL.

### Required fields

Field	Description
Type	Select the response type to return to the requestor: Found (302), Moved (301), Temporary redirect (307), and Permanent redirect (308).
Protocol	Match Request, HTTP, HTTPS.
Hostname	Select the host name you want the request to be redirected to. Leave blank to preserve the incoming host.
Path	Define the path to use in the redirect. Leave blank to preserve the incoming path.
Query string	Define the query string used in the redirect. Leave blank to preserve the incoming query string.
Fragment	Define the fragment to use in the redirect. Leave blank to preserve the incoming fragment.

We highly recommend that you use an absolute URL. Using a relative URL might redirect Azure CDN URLs to an invalid path.

## URL rewrite

Use this action to rewrite the path of a request that's en route to your origin.

### Required fields

Field	Description
Source pattern	Define the source pattern in the URL path to replace. Currently, source pattern uses a prefix-based match. To match all URL paths, use a forward slash (/) as the source pattern value.
Destination	Define the destination path to use in the rewrite. The destination path overwrites the source pattern.
Preserve unmatched path	If set to <b>Yes</b> , the remaining path after the source pattern is appended to the new destination path.

## Next steps

- [Azure CDN overview](#)
- [Standard rules engine reference](#)

- Match conditions in the Standard rules engine
- Enforce HTTPS by using the Standard rules engine

# Azure CDN from Verizon Premium rules engine reference

8/23/2019 • 3 minutes to read • [Edit Online](#)

This article lists detailed descriptions of the available match conditions and features for the Azure Content Delivery Network (CDN) [rules engine](#).

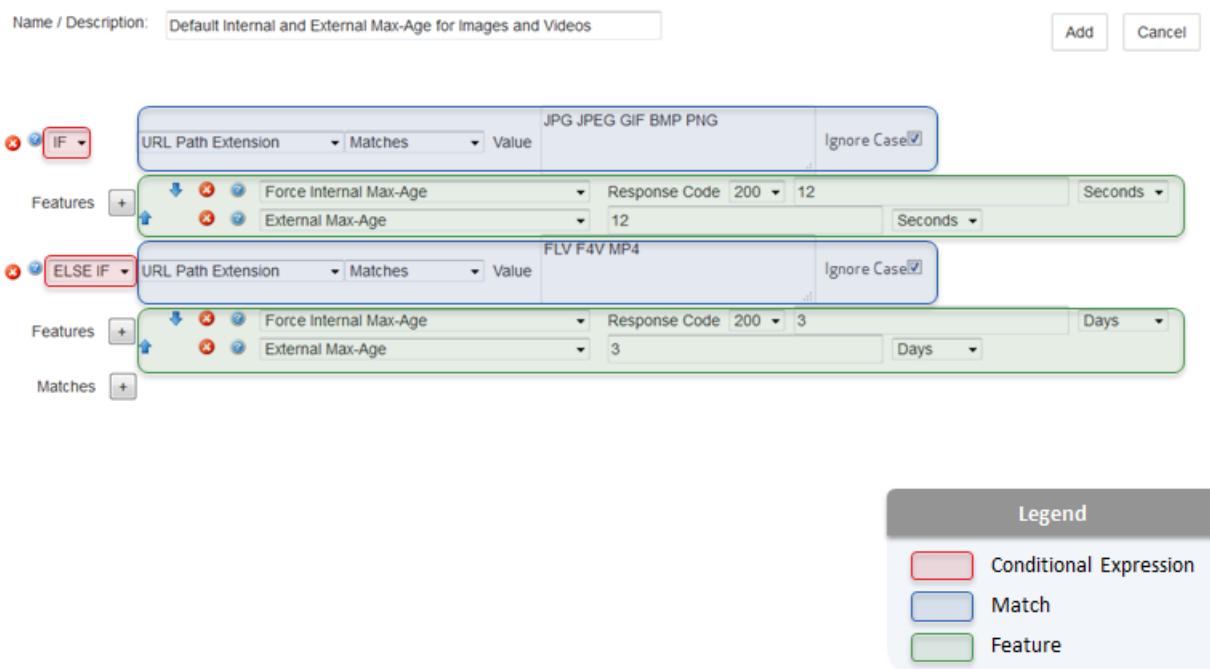
The rules engine is designed to be the final authority on how specific types of requests are processed by the CDN.

## Common uses:

- Override or define a custom cache policy.
- Secure or deny requests for sensitive content.
- Redirect requests.
- Store custom log data.

## Terminology

A rule is defined through the use of **conditional expressions**, **match conditions**, and **features**. These elements are highlighted in the following illustration:



## Syntax

The manner in which special characters are treated varies according to how a match condition or feature handles text values. A match condition or feature may interpret text in one of the following ways:

1. **Literal values**
2. **Wildcard values**
3. **Regular expressions**

### Literal values

Text that is interpreted as a literal value treats all special characters, with the exception of the % symbol, as a part of the value that must be matched. In other words, a literal match condition set to `\'*\'` is only satisfied when that exact value (that is, `\'*\'`) is found.

A percentage symbol is used to indicate URL encoding (for example, `%20`).

## Wildcard values

Text that is interpreted as a wildcard value assigns additional meaning to special characters. The following table describes how the following set of characters is interpreted:

CHARACTER	DESCRIPTION
\	A backslash is used to escape any of the characters specified in this table. A backslash must be specified directly before the special character that should be escaped. For example, the following syntax escapes an asterisk: <code>\*</code>
%	A percentage symbol is used to indicate URL encoding (for example, <code>%20</code> ).
*	An asterisk is a wildcard that represents one or more characters.
Space	A space character indicates that a match condition may be satisfied by either of the specified values or patterns.
'value'	<p>A single quote does not have special meaning. However, a set of single quotes is used to indicate that a value should be treated as a literal value. It can be used in the following ways:</p> <ul style="list-style-type: none"> <li>- It allows a match condition to be satisfied whenever the specified value matches any portion of the comparison value. For example, <code>'ma'</code> would match any of the following strings:</li> </ul> <p style="margin-left: 20px;">/business/<b>marathon</b>/asset.htm  <b>map.gif</b>  /business/template.<b>map</b></p> <ul style="list-style-type: none"> <li>- It allows a special character to be specified as a literal character. For example, you may specify a literal space character by enclosing a space character within a set of single quotes (that is, <code>' '</code> or <code>'sample value'</code>).</li> <li>- It allows a blank value to be specified. Specify a blank value by specifying a set of single quotes (that is, <code>"</code>).</li> </ul> <p><b>Important:</b></p> <ul style="list-style-type: none"> <li>- If the specified value does not contain a wildcard, then it is automatically considered a literal value, which means that it is not necessary to specify a set of single quotes.</li> <li>- If a backslash does not escape another character in this table, it is ignored when it is specified within a set of single quotes.</li> <li>- Another way to specify a special character as a literal character is to escape it using a backslash (that is, <code>\</code>).</li> </ul>

## Regular expressions

Regular expressions define a pattern that is searched for within a text value. Regular expression notation defines specific meanings to a variety of symbols. The following table indicates how special characters are treated by match conditions and features that support regular expressions.

SPECIAL CHARACTER	DESCRIPTION
\	A backslash escapes the character that follows it, which causes that character to be treated as a literal value instead of taking on its regular expression meaning. For example, the following syntax escapes an asterisk: <code>\*</code>
%	The meaning of a percentage symbol depends on its usage.  <code>%{HTTPVariable}</code> : This syntax identifies an HTTP variable. <code>%{HTTPVariable%Pattern}</code> : This syntax uses a percentage symbol to identify an HTTP variable and as a delimiter. <code>\%</code> : Escaping a percentage symbol allows it to be used as a literal value or to indicate URL encoding (for example, <code>\%20</code> ).
*	An asterisk allows the preceding character to be matched zero or more times.
Space	A space character is typically treated as a literal character.
'value'	Single quotes are treated as literal characters. A set of single quotes does not have special meaning.

Match conditions and features that support regular expressions accept patterns defined by Perl Compatible Regular Expressions (PCRE).

## Next steps

- [Rules engine match conditions](#)
- [Rules engine conditional expressions](#)
- [Rules engine features](#)
- [Override HTTP behavior using the rules engine](#)
- [Azure CDN overview](#)

# Azure CDN from Verizon Premium rules engine conditional expressions

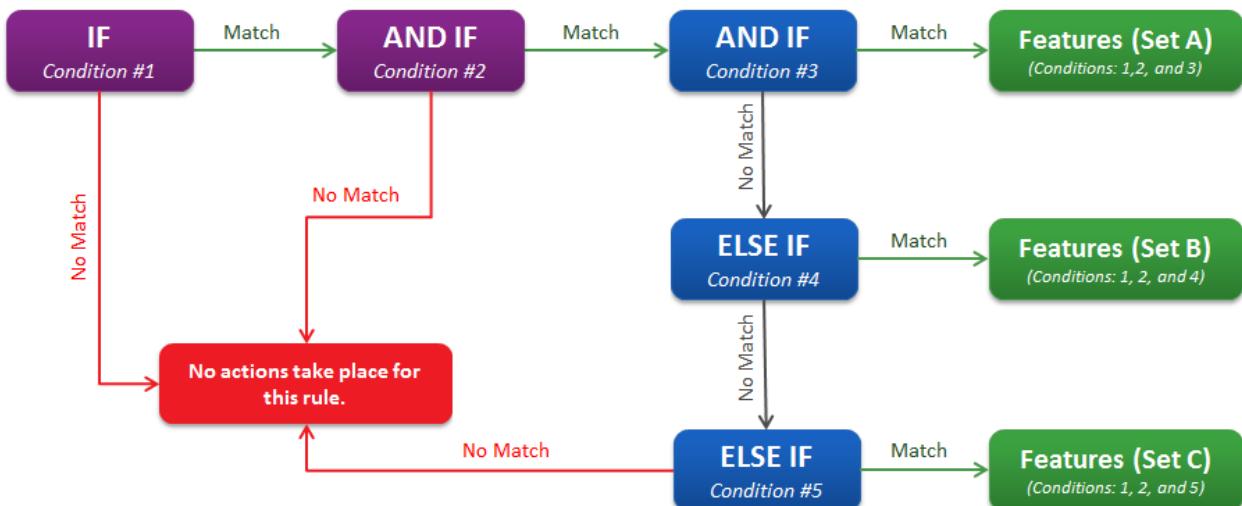
11/14/2019 • 2 minutes to read • [Edit Online](#)

This article lists detailed descriptions of the Conditional Expressions for Azure Content Delivery Network (CDN) Rules Engine.

The first part of a rule is the Conditional Expression.

CONDITIONAL EXPRESSION	DESCRIPTION
IF	An IF expression is always a part of the first statement in a rule. Like all other conditional expressions, this IF statement must be associated with a match. If no additional conditional expressions are defined, this match determines the criterion that must be met before a set of features may be applied to a request.
AND IF	An AND IF expression may only be added after the following types of conditional expressions: IF, AND IF. It indicates that there is another condition that must be met for the initial IF statement.
ELSE IF	An ELSE IF expression specifies an alternative condition that must be met before a set of features specific to this ELSE IF statement takes place. The presence of an ELSE IF statement indicates the end of the previous statement. The only conditional expression that may be placed after an ELSE IF statement is another ELSE IF statement. This means that an ELSE IF statement may only be used to specify a single additional condition that has to be met.

## Example:



**TIP**

A subsequent rule may override the actions specified by a previous rule. Example: A catch-all rule secures all requests via Token-Based Authentication. Another rule may be created directly below it to make an exception for certain types of requests.

## Next steps

- [Azure CDN overview](#)
- [Rules engine reference](#)
- [Rules engine match conditions](#)
- [Rules engine features](#)
- [Overriding default HTTP behavior using the rules engine](#)

# Azure CDN from Verizon Premium rules engine match conditions

7/5/2019 • 32 minutes to read • [Edit Online](#)

This article lists detailed descriptions of the available match conditions for the Azure Content Delivery Network (CDN) from Verizon Premium [rules engine](#).

The second part of a rule is the match condition. A match condition identifies specific types of requests for which a set of features will be performed.

For example, you can use a match condition to:

- Filter requests for content at a particular location.
- Filter requests generated from a particular IP address or country/region.
- Filter requests by header information.

## Always match condition

The Always match condition applies a default set of features to all requests.

NAME	PURPOSE
Always	Applies a default set of features to all requests.

## Device match condition

The Device match condition identifies requests made from a mobile device based on its properties.

NAME	PURPOSE
Device	Identifies requests made from a mobile device based on its properties.

## Location match conditions

The Location match conditions identify requests based on the requester's location.

NAME	PURPOSE
AS Number	Identifies requests that originate from a particular network.
Country	Identifies requests that originate from the specified countries/regions.

## Origin match conditions

The Origin match conditions identify requests that point to Content Delivery Network storage or a customer origin server.

NAME	PURPOSE
CDN Origin	Identifies requests for content stored in Content Delivery Network storage.
Customer Origin	Identifies requests for content stored on a specific customer origin server.

## Request match conditions

The Request match conditions identify requests based on their properties.

NAME	PURPOSE
Client IP Address	Identifies requests that originate from a particular IP address.
Cookie Parameter	Checks the cookies associated with each request for the specified value.
Cookie Parameter Regex	Checks the cookies associated with each request for the specified regular expression.
Edge Cname	Identifies requests that point to a specific edge CNAME.
Referring Domain	Identifies requests that were referred from the specified host names.
Request Header Literal	Identifies requests that contain the specified header set to a specified value.
Request Header Regex	Identifies requests that contain the specified header set to a value that matches the specified regular expression.
Request Header Wildcard	Identifies requests that contain the specified header set to a value that matches the specified pattern.
Request Method	Identifies requests by their HTTP method.
Request Scheme	Identifies requests by their HTTP protocol.

## URL match conditions

The URL match conditions identify requests based on their URLs.

NAME	PURPOSE
URL Path Directory	Identifies requests by their relative path.
URL Path Extension	Identifies requests by their file name extension.
URL Path Filename	Identifies requests by their file name.
URL Path Literal	Compares a request's relative path to the specified value.

NAME	PURPOSE
URL Path Regex	Compares a request's relative path to the specified regular expression.
URL Path Wildcard	Compares a request's relative path to the specified pattern.
URL Query Literal	Compares a request's query string to the specified value.
URL Query Parameter	Identifies requests that contain the specified query string parameter set to a value that matches a specified pattern.
URL Query Regex	Identifies requests that contain the specified query string parameter set to a value that matches a specified regular expression.
URL Query Wildcard	Compares the specified value against the request's query string.

## Reference for rules engine match conditions

### Always

The Always match condition applies a default set of features to all requests.

[Back to top](#)

---

### AS Number

The AS Number network is defined by its autonomous system number (ASN).

The **Matches/Does Not Match** option determines the conditions under which the AS Number match condition is met:

- **Matches:** Requires that the ASN of the client network matches one of the specified ASNs.
- **Does Not Match:** Requires that the ASN of the client network does not match any of the specified ASNs.

Key information:

- Specify multiple ASNs by delimiting each one with a single space. For example, 64514 64515 matches requests that arrive from either 64514 or 64515.
- Certain requests might not return a valid ASN. A question mark (?) will match requests for which a valid ASN could not be determined.
- Specify the entire ASN for the desired network. Partial values will not be matched.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## CDN Origin

The CDN Origin match condition is met when both of the following conditions are met:

- Content from CDN storage was requested.
- The request URI uses the type of content access point (for example, /000001) that's defined in this match condition:
  - CDN URL: The request URI must contain the selected content access point.
  - Edge CNAME URL: The corresponding edge CNAME configuration must point to the selected content access point.

Key information:

- The content access point identifies the service that should serve the requested content.
- Don't use an AND IF statement to combine certain match conditions. For example, combining a CDN Origin match condition with a Customer Origin match condition would create a match pattern that could never be matched. For this reason, two CDN Origin match conditions cannot be combined through an AND IF statement.

[Back to top](#)

---

## Client IP Address

The **Matches/Does Not Match** option determines the conditions under which the Client IP Address match condition is met:

- **Matches:** Requires that the client's IP address matches one of the specified IP addresses.
- **Does Not Match:** Requires that the client's IP address does not match any of the specified IP addresses.

Key information:

- Use CIDR notation.
- Specify multiple IP addresses and/or IP address blocks by delimiting each one with a single space. For example:
  - **IPv4 example:** 1.2.3.4 10.20.30.40 matches any requests that arrive from either address 1.2.3.4 or 10.20.30.40.
  - **IPv6 example:** 1:2:3:4:5:6:7:8 10:20:30:40:50:60:70:80 matches any requests that arrive from either address 1:2:3:4:5:6:7:8 or 10:20:30:40:50:60:70:80.
- The syntax for an IP address block is the base IP address followed by a forward slash and the prefix size. For example:
  - **IPv4 example:** 5.5.5.64/26 matches any requests that arrive from addresses 5.5.5.64 through 5.5.5.127.
  - **IPv6 example:** 1:2:3:/48 matches any requests that arrive from addresses 1:2:3:0:0:0:0 through 1:2:3:ffff:ffff:ffff:ffff.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

---

## Cookie Parameter

The **Matches/Does Not Match** option determines the conditions under which the Cookie Parameter match condition is met.

- **Matches:** Requires a request to contain the specified cookie with a value that matches at least one of the values that are defined in this match condition.
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified cookie.
  - It contains the specified cookie, but its value does not match any of the values that are defined in this match condition.

Key information:

- Cookie name:
  - Because wildcard values, including asterisks (\*), are not supported when you're specifying a cookie name, only exact cookie name matches are eligible for comparison.
  - Only a single cookie name can be specified per instance of this match condition.
  - Cookie name comparisons are case-insensitive.
- Cookie value:
  - Specify multiple cookie values by delimiting each one with a single space.
  - A cookie value can take advantage of [wildcard values](#).
  - If a wildcard value has not been specified, then only an exact match will satisfy this match condition. For example, specifying "Value" will match "Value," but not "Value1" or "Value2."
  - Use the **Ignore Case** option to control whether a case-sensitive comparison is made against the request's cookie value.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

---

## Cookie Parameter Regex

The Cookie Parameter Regex match condition defines a cookie name and value. You can use [regular expressions](#) to define the desired cookie value.

The **Matches/Does Not Match** option determines the conditions under which the Cookie Parameter Regex match condition is met.

- **Matches:** Requires a request to contain the specified cookie with a value that matches the specified regular expression.
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified cookie.
  - It contains the specified cookie, but its value does not match the specified regular expression.

Key information:

- Cookie name:
  - Because regular expressions and wildcard values, including asterisks (\*), are not supported when you're specifying a cookie name, only exact cookie name matches are eligible for comparison.
  - Only a single cookie name can be specified per instance of this match condition.
  - Cookie name comparisons are case-insensitive.
- Cookie value:
  - A cookie value can take advantage of regular expressions.
  - Use the **Ignore Case** option to control whether a case-sensitive comparison is made against the request's cookie value.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## Country

You can specify a country through its country code.

The **Matches/Does Not Match** option determines the conditions under which the Country match condition is met:

- **Matches:** Requires the request to contain the specified country code values.
- **Does Not Match:** Requires that the request does not contain the specified country code values.

Key information:

- Specify multiple country codes by delimiting each one with a single space.
- Wildcards are not supported when you're specifying a country code.
- The "EU" and "AP" country codes do not encompass all IP addresses in those regions.
- Certain requests might not return a valid country code. A question mark (?) will match requests for which a valid country code could not be determined.
- Country codes are case-sensitive.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

### Implementing Country Filtering by using the rules engine

This match condition allows you to perform a multitude of customizations based on the location from which a request originated. For example, the behavior of the Country Filtering feature can be replicated through the following configuration:

- URL Path Wildcard match: Set the [URL Path Wildcard match condition](#) to the directory that will be secured. Append an asterisk to the end of the relative path to ensure that access to all of its children will be restricted by this rule.
- Country match: Set the Country match condition to the desired set of countries.
  - Allow: Set the Country match condition to **Does Not Match** to allow only the specified countries access to content stored in the location defined by the URL Path Wildcard match condition.
  - Block: Set the Country match condition to **Matches** to block the specified countries from accessing content stored in the location defined by the URL Path Wildcard match condition.
- Deny Access (403) Feature: Enable the [Deny Access \(403\) feature](#) to replicate the allow or block portion of the Country Filtering feature.

[Back to top](#)

---

## Customer Origin

Key information:

- The Customer Origin match condition is met regardless of whether content is requested through a CDN URL or an edge CNAME URL that points to the selected customer origin.
- A customer origin configuration that's referenced by a rule cannot be deleted from the Customer Origin page. Before you attempt to delete a customer origin configuration, make sure that the following configurations do not reference it:
  - A Customer Origin match condition
  - An edge CNAME configuration
- Don't use an AND IF statement to combine certain match conditions. For example, combining a Customer Origin match condition with a CDN Origin match condition would create a match pattern that could never be matched. For this reason, two Customer Origin match conditions cannot be combined through an AND IF statement.

[Back to top](#)

---

## Device

The Device match condition identifies requests made from a mobile device based on its properties. Mobile device detection is achieved through [WURFL](#).

The **Matches/Does Not Match** option determines the conditions under which the Device match condition is met:

- **Matches:** Requires the requester's device to match the specified value.
- **Does Not Match:** Requires that the requester's device does not match the specified value.

Key information:

- Use the **Ignore Case** option to specify whether the specified value is case-sensitive.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache

- Internal Max-Stale

#### String Type

A WURFL capability typically accepts any combination of numbers, letters, and symbols. Due to the flexible nature of this capability, you must choose how the value associated with this match condition is interpreted. The following table describes the available set of options:

TYPE	DESCRIPTION
Literal	Select this option to prevent most characters from taking on special meaning by using their <a href="#">literal value</a> .
Wildcard	Select this option to take advantage of all [wildcard characters]( <a href="#">wildcard values</a> ).
Regex	Select this option to use <a href="#">regular expressions</a> . Regular expressions are useful for defining a pattern of characters.

#### WURFL capabilities

A WURFL capability refers to a category that describes mobile devices. The selected capability determines the type of mobile device description that is used to identify requests.

The following table lists WURFL capabilities and their variables for the rules engine.

#### NOTE

The following variables are supported in the **Modify Client Request Header** and **Modify Client Response Header** features.

CAPABILITY	VARIABLE	DESCRIPTION	SAMPLE VALUES
Brand Name	%{wurfl_cap_brand_name}	A string that indicates the brand name of the device.	Samsung
Device OS	%{wurfl_cap_device_os}	A string that indicates the operating system installed on the device.	iOS
Device OS Version	%{wurfl_cap_device_os_version}	A string that indicates the version number of the operating system installed on the device.	1.0.1
Dual Orientation	%{wurfl_cap_dual_orientation}	A Boolean that indicates whether the device supports dual orientation.	true
HTML Preferred DTD	%{wurfl_cap_html_preferred_dtd}	A string that indicates the mobile device's preferred document type definition (DTD) for HTML content.	none xhtml_basic html5
Image Inlining	%{wurfl_cap_image_inlining}	A Boolean that indicates whether the device supports Base64 encoded images.	false

Capability	Variable	Description	Sample Values
Is Android	<code>%{wurfl_vcap_is_android}</code>	A Boolean that indicates whether the device uses the Android OS.	true
Is iOS	<code>%{wurfl_vcap_is_ios}</code>	A Boolean that indicates whether the device uses iOS.	false
Is Smart TV	<code>%{wurfl_cap_is_smarttv}</code>	A Boolean that indicates whether the device is a smart TV.	false
Is Smartphone	<code>%{wurfl_vcap_is_smartphone}</code>	A Boolean that indicates whether the device is a smartphone.	true
Is Tablet	<code>%{wurfl_cap_is_tablet}</code>	A Boolean that indicates whether the device is a tablet. This description is OS-independent.	true
Is Wireless Device	<code>%{wurfl_cap_is_wireless_device}</code>	A Boolean that indicates whether the device is considered a wireless device.	true
Marketing Name	<code>%{wurfl_cap_marketing_name}</code>	A string that indicates the device's marketing name.	BlackBerry 8100 Pearl
Mobile Browser	<code>%{wurfl_cap_mobile_browser}</code>	A string that indicates the browser that's used to request content from the device.	Chrome
Mobile Browser Version	<code>%{wurfl_cap_mobile_browser_version}</code>	A string that indicates the version of the browser that's used to request content from the device.	31
Model Name	<code>%{wurfl_cap_model_name}</code>	A string that indicates the device's model name.	s3
Progressive Download	<code>%{wurfl_cap_progressive_download}</code>	A Boolean that indicates whether the device supports the playback of audio and video while it is still being downloaded.	true
Release Date	<code>%{wurfl_cap_release_date}</code>	A string that indicates the year and month on which the device was added to the WURFL database.  Format: <code>yyyy_mm</code>	2013_december
Resolution Height	<code>%{wurfl_cap_resolution_height}</code>	An integer that indicates the device's height in pixels.	768

CAPABILITY	VARIABLE	DESCRIPTION	SAMPLE VALUES
Resolution Width	%{wurfl_cap_resolution_width}	An integer that indicates the device's width in pixels.	1024

[Back to top](#)

## Edge Cname

Key information:

- The list of available edge CNAMEs is limited to those edge CNAMEs that have been configured on the Edge CNAMEs page for the platform on which the rules engine is being configured.
- Before you attempt to delete an edge CNAME configuration, make sure that an Edge Cname match condition does not reference it. Edge CNAME configurations that have been defined in a rule cannot be deleted from the Edge CNAMEs page.
- Don't use an AND IF statement to combine certain match conditions. For example, combining an Edge Cname match condition with a Customer Origin match condition would create a match pattern that could never be matched. For this reason, two Edge Cname match conditions cannot be combined through an AND IF statement.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

## Referring Domain

The host name associated with the referrer through which content was requested determines whether the Referring Domain condition is met.

The **Matches/Does Not Match** option determines the conditions under which the Referring Domain match condition is met:

- **Matches:** Requires the referring host name to match the specified values.
- **Does Not Match:** Requires that the referring host name does not match the specified value.

Key information:

- Specify multiple host names by delimiting each one with a single space.
- This match condition supports [wildcard values](#).
- If the specified value does not contain an asterisk, it must be an exact match for the referrer's host name. For example, specifying "mydomain.com" would not match "www.mydomain.com."
- Use the **Ignore Case** option to control whether a case-sensitive comparison is made.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill

- Default Internal Max-Age
- Force Internal Max-Age
- Ignore Origin No-Cache
- Internal Max-Stale

[Back to top](#)

---

## Request Header Literal

The **Matches/Does Not Match** option determines the conditions under which the Request Header Literal match condition is met.

- **Matches:** Requires the request to contain the specified header. Its value must match the one that's defined in this match condition.
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified header.
  - It contains the specified header, but its value does not match the one that's defined in this match condition.

Key information:

- Header name comparisons are always case-insensitive. Use the **Ignore Case** option to control the case-sensitivity of header value comparisons.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## Request Header Regex

The **Matches/Does Not Match** option determines the conditions under which the Request Header Regex match condition is met.

- **Matches:** Requires the request to contain the specified header. Its value must match the pattern that's defined in the specified [regular expression](#).
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified header.
  - It contains the specified header, but its value does not match the specified regular expression.

Key information:

- Header name:
  - Header name comparisons are case-insensitive.
  - Replace spaces in the header name with "%20."
- Header value:
  - A header value can take advantage of regular expressions.

- Use the **Ignore Case** option to control the case-sensitivity of header value comparisons.
- The match condition is met only when a header value exactly matches at least one of the specified patterns.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## Request Header Wildcard

The **Matches/Does Not Match** option determines the conditions under which the Request Header Wildcard match condition is met.

- **Matches:** Requires the request to contain the specified header. Its value must match at least one of the values that are defined in this match condition.
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified header.
  - It contains the specified header, but its value does not match any of the specified values.

Key information:

- Header name:
  - Header name comparisons are case-insensitive.
  - Spaces in the header name should be replaced with "%20." You can also use this value to specify spaces in a header value.
- Header value:
  - A header value can take advantage of [wildcard values](#).
  - Use the **Ignore Case** option to control the case-sensitivity of header value comparisons.
  - This match condition is met when a header value exactly matches to at least one of the specified patterns.
  - Specify multiple values by delimiting each one with a single space.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## Request Method

The Request Method match condition is met only when assets are requested through the selected request method. The available request methods are:

- GET
- HEAD
- POST
- OPTIONS
- PUT
- DELETE
- TRACE
- CONNECT

Key information:

- By default, only the GET request method can generate cached content on the network. All other request methods are proxied through the network.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

### Request Scheme

The Request Scheme match condition is met only when assets are requested through the selected protocol. The available protocols are:

- HTTP
- HTTPS

Key information:

- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

### URL Path Directory

Identifies a request by its relative path, which excludes the file name of the requested asset.

The **Matches/Does Not Match** option determines the conditions under which the URL Path Directory match condition is met.

- **Matches:** Requires the request to contain a relative URL path, excluding the file name, that matches the specified URL pattern.

- **Does Not Match:** Requires the request to contain a relative URL path, excluding file name, that does not match the specified URL pattern.

Key information:

- Use the **Relative to** option to specify whether the URL comparison starts before or after the content access point. The content access point is the portion of the path that appears between the Verizon CDN hostname and the relative path to the requested asset (for example, /800001/CustomerOrigin). It identifies a location by server type (for example, CDN or customer origin) and your customer account number.

The following values are available for the **Relative to** option:

- **Root:** Indicates that the URL comparison point begins directly after the CDN hostname.

For example: `http://wpc.0001.<domain>/800001/myorigin/myfolder/index.htm`

- **Origin:** Indicates that the URL comparison point begins after the content access point (for example, /000001 or /800001/myorigin). Because the \*.azureedge.net CNAME is created relative to the origin directory on the Verizon CDN hostname by default, Azure CDN users should use the **Origin** value.

For example: `https://<endpoint>.azureedge.net/myfolder/index.htm`

This URL points to the following Verizon CDN hostname: `http://wpc.0001.`

`<domain>/800001/myorigin/myfolder/index.htm`

- An edge CNAME URL is rewritten to a CDN URL prior to the URL comparison.

For example, both of the following URLs point to the same asset and therefore have the same URL path.

- CDN URL: `http://wpc.0001.<domain>/800001/CustomerOrigin/path/asset.htm`
- Edge CNAME URL: `http://<endpoint>.azureedge.net/path/asset.htm`

Additional information:

- Custom domain: `https://my.domain.com/path/asset.htm`
  - URL path (relative to root): `/800001/CustomerOrigin/path/`
  - URL path (relative to origin): `/path/`
- The portion of the URL that is used for the URL comparison ends just before the file name of the requested asset. A trailing forward slash is the last character in this type of path.
- Replace any spaces in the URL path pattern with "%20."
- Each URL path pattern can contain one or more asterisks (\*), where each asterisk matches a sequence of one or more characters.
- Specify multiple URL paths in the pattern by delimiting each one with a single space.

For example: `*/sales/ */marketing/`

- A URL path specification can take advantage of [wildcard values](#).
- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.

[Back to top](#)

## URL Path Extension

Identifies requests by the file extension of the requested asset.

The **Matches/Does Not Match** option determines the conditions under which the URL Path Extension match condition is met.

- **Matches:** Requires the URL of the request to contain a file extension that exactly matches the specified pattern.

For example, if you specify "htm", "htm" assets are matched, but not "html" assets.

- **Does Not Match:** Requires the URL request to contain a file extension that does not match the specified pattern.

Key information:

- Specify the file extensions to match in the **Value** box. Do not include a leading period; for example, use htm instead of .htm.
- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.
- Specify multiple file extensions by delimiting each extension with a single space.

For example: htm html

- For example, specifying "htm" matches "htm" assets, but not "html" assets.

#### Sample Scenario

The following sample configuration assumes that this match condition is met when a request matches one of the specified extensions.

Value specification: asp aspx php html

This match condition is met when it finds URLs that end with the following extensions:

- .asp
- .aspx
- .php
- .html

[Back to top](#)

---

#### URL Path Filename

Identifies requests by the file name of the requested asset. For the purposes of this match condition, a file name consists of the name of the requested asset, a period, and the file extension (for example, index.html).

The **Matches/Does Not Match** option determines the conditions under which the URL Path Filename match condition is met.

- **Matches:** Requires the request to contain a file name in its URL path that matches the specified pattern.
- **Does Not Match:** Requires the request to contain a file name in its URL path that does not match the specified pattern.

Key information:

- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.
  - To specify multiple file extensions, separate each extension with a single space.
- For example: index.htm index.html
- Replace spaces in a file name value with "%20."

- A file name value can take advantage of [wildcard values](#). For example, each file name pattern can consist of one or more asterisks (\*), where each asterisk matches a sequence of one or more characters.
- If wildcard characters are not specified, then only an exact match will satisfy this match condition.

For example, specifying "presentation.ppt" matches an asset named "presentation.ppt," but not one named "presentation.pptx."

[Back to top](#)

---

## URL Path Literal

Compares a request's URL path, including file name, to the specified value.

The **Matches/Does Not Match** option determines the conditions under which the URL Path Literal match condition is met.

- **Matches:** Requires the request to contain a URL path that matches the specified pattern.
- **Does Not Match:** Requires the request to contain a URL path that does not match the specified pattern.

Key information:

- Use the **Relative to** option to specify whether the URL comparison point begins before or after the content access point.

The following values are available for the **Relative to** option:

- **Root:** Indicates that the URL comparison point begins directly after the CDN hostname.

For example: **http://wpc.0001.<domain>/800001/myorigin/myfolder/index.htm**

- **Origin:** Indicates that the URL comparison point begins after the content access point (for example, /000001 or /800001/myorigin). Because the \*.azureedge.net CNAME is created relative to the origin directory on the Verizon CDN hostname by default, Azure CDN users should use the **Origin** value.

For example: **https://<endpoint>.azureedge.net/myfolder/index.htm**

This URL points to the following Verizon CDN hostname: **http://wpc.0001.**

**<domain>/800001/myorigin/myfolder/index.htm**

- An edge CNAME URL is rewritten to a CDN URL prior to a URL comparison.

For example, both of the following URLs point to the same asset and therefore have the same URL path:

- CDN URL: **http://wpc.0001.<domain>/800001/CustomerOrigin/path/asset.htm**
- Edge CNAME URL: **http://<endpoint>.azureedge.net/path/asset.htm**

Additional information:

- URL path (relative to root): /800001/CustomerOrigin/path/asset.htm
- URL path (relative to origin): /path/asset.htm
- Query strings in the URL are ignored.
- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.
- The value specified for this match condition is compared against the relative path of the exact request made by the client.

- To match all requests made to a particular directory, use the [URL Path Directory](#) or the [URL Path Wildcard](#) match condition.

[Back to top](#)

---

## URL Path Regex

Compares a request's URL path to the specified [regular expression](#).

The **Matches/Does Not Match** option determines the conditions under which the URL Path Regex match condition is met.

- **Matches:** Requires the request to contain a URL path that matches the specified regular expression.
- **Does Not Match:** Requires the request to contain a URL path that does not match the specified regular expression.

Key information:

- An edge CNAME URL is rewritten to a CDN URL prior to URL comparison.

For example, both URLs point to the same asset and therefore have the same URL path.

- CDN URL: `http://wpc.0001.<domain>/800001/CustomerOrigin/path/asset.htm`
- Edge CNAME URL: `http://my.domain.com/path/asset.htm`

Additional information:

- URL path: `/800001/CustomerOrigin/path/asset.htm`
- Query strings in the URL are ignored.
- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.
- Spaces in the URL path should be replaced with "%20."

[Back to top](#)

---

## URL Path Wildcard

Compares a request's relative URL path to the specified wildcard pattern.

The **Matches/Does Not Match** option determines the conditions under which the URL Path Wildcard match condition is met.

- **Matches:** Requires the request to contain a URL path that matches the specified wildcard pattern.
- **Does Not Match:** Requires the request to contain a URL path that does not match the specified wildcard pattern.

Key information:

- **Relative to** option: This option determines whether the URL comparison point begins before or after the content access point.

This option can have the following values:

- **Root:** Indicates that the URL comparison point begins directly after the CDN hostname.

For example: `http://wpc.0001.<domain>/800001/myorigin/myfolder/index.htm`

- **Origin:** Indicates that the URL comparison point begins after the content access point (for example, /000001 or /800001/myorigin). Because the \*.azureedge.net CNAME is created relative to the origin directory on the Verizon CDN hostname by default, Azure CDN users should use the **Origin** value.

For example: https://<endpoint>.azureedge.net/**myfolder/index.htm**

This URL points to the following Verizon CDN hostname: http://wpc.0001.  
<domain>/800001/myorigin/**myfolder/index.htm**

- An edge CNAME URL is rewritten to a CDN URL prior to URL comparison.

For example, both of the following URLs point to the same asset and therefore have the same URL path:

- CDN URL: http://wpc.0001.<domain>/800001/CustomerOrigin/path/asset.htm
- Edge CNAME URL: http://<endpoint>.azureedge.net/path/asset.htm

Additional information:

- URL path (relative to root): /800001/CustomerOrigin/path/asset.htm
- URL path (relative to origin): /path/asset.htm

- Specify multiple URL paths by delimiting each one with a single space.

For example: /marketing/asset.\* /sales/\*.htm

- Query strings in the URL are ignored.
- Use the **Ignore Case** option to control whether a case-sensitive comparison is performed.
- Replace spaces in the URL path with "%20."
- The value specified for a URL path can take advantage of [wildcard values](#). Each URL path pattern can contain one or more asterisks (\*), where each asterisk can match a sequence of one or more characters.

#### Sample Scenarios

The sample configurations in the following table assume that this match condition is met when a request matches the specified URL pattern:

VALUE	RELATIVE TO	RESULT
*/test.html */test.php	Root or Origin	This pattern is matched by requests for assets named "test.html" or "test.php" in any folder.
/80ABCD/origin/text/*	Root	This pattern is matched when the requested asset meets the following criteria: - It must reside on a customer origin called "origin." - The relative path must start with a folder called "text." That is, the requested asset can either reside in the "text" folder or one of its recursive subfolders.
/css/ /js/	Root or Origin	This pattern is matched by all CDN or edge CNAME URLs that contain a css or js folder.

Value	Relative To	Result
*.jpg *.gif *.png	Root or Origin	This pattern is matched by all CDN or edge CNAME URLs ending with .jpg, .gif, or .png. An alternative way to specify this pattern is with the <a href="#">URL Path Extension match condition</a> .
/images/* /media/*	Origin	This pattern is matched by CDN or edge CNAME URLs whose relative path starts with an "images" or "media" folder. - CDN URL: http://wpc.0001.<domain>/800001/myorigin/images/sales/event1.png - Sample edge CNAME URL: http://cdn.mydomain.com/images/sales/event1.png

[Back to top](#)

## URL Query Literal

Compares a request's query string to the specified value.

The **Matches/Does Not Match** option determines the conditions under which the URL Query Literal match condition is met.

- **Matches:** Requires the request to contain a URL query string that matches the specified query string.
- **Does Not Match:** Requires the request to contain a URL query string that does not match the specified query string.

Key information:

- Only exact query string matches satisfy this match condition.
- Use the **Ignore Case** option to control the case-sensitivity of query string comparisons.
- Do not include a leading question mark (?) in the query string value text.
- Certain characters require URL encoding. Use the percentage symbol to URL encode the following characters:

Character	URL Encoding
Space	%20
&	%25

- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

## URL Query Parameter

Identifies requests that contain the specified query string parameter. This parameter is set to a value that matches a specified pattern. Query string parameters (for example, parameter=value) in the request URL determine whether this condition is met. This match condition identifies a query string parameter by its name and accepts one or more values for the parameter value.

The **Matches/Does Not Match** option determines the conditions under which the URL Query Parameter match condition is met.

- **Matches:** Requires a request to contain the specified parameter with a value that matches at least one of the values that are defined in this match condition.
- **Does Not Match:** Requires that the request meets either of the following criteria:
  - It does not contain the specified parameter.
  - It contains the specified parameter, but its value does not match any of the values that are defined in this match condition.

This match condition provides an easy way to specify parameter name/value combinations. For more flexibility if you are matching a query string parameter, consider using the [URL Query Wildcard](#) match condition.

Key information:

- Only a single URL query parameter name can be specified per instance of this match condition.
- Because wildcard values are not supported when a parameter name is specified, only exact parameter name matches are eligible for comparison.
- Parameter value(s) can include [wildcard values](#).
  - Each parameter value pattern can consist of one or more asterisks (\*), where each asterisk can match a sequence of one or more characters.
  - Certain characters require URL encoding. Use the percentage symbol to URL encode the following characters:

CHARACTER	URL ENCODING
Space	%20
&	%25

- Specify multiple query string parameter values by delimiting each one with a single space. This match condition is met when a request contains one of the specified name/value combinations.
  - Example 1:
    - Configuration:

```
ValueA ValueB
```
    - This configuration matches the following query string parameters:

```
Parameter1=ValueA
Parameter1=ValueB
```

- Example 2:
  - Configuration:
 

```
Value%20A Value%20B
```
  - This configuration matches the following query string parameters:
 

```
Parameter1=Value%20A
```

```
Parameter1=Value%20B
```
- This match condition is met only when there is an exact match to at least one of the specified query string name/value combinations.

For example, if you use the configuration in the previous example, the parameter name/value combination "Parameter1=ValueAdd" would not be considered a match. However, if you specify either of the following values, it will match that name/value combination:

- ValueA ValueB ValueAdd
- ValueA\* ValueB
- Use the **Ignore Case** option to control the case-sensitivity of query string comparisons.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

#### Sample scenarios

The following example demonstrates how this option works in specific situations:

NAME	VALUE	RESULT
User	Joe	This pattern is matched when the query string for a requested URL is "?user=joe."
User	*	This pattern is matched when the query string for a requested URL contains a User parameter.
Email	Joe*	This pattern is matched when the query string for a requested URL contains an Email parameter that starts with "Joe."

[Back to top](#)

---

#### URL Query Regex

Identifies requests that contain the specified query string parameter. This parameter is set to a value that matches a specified [regular expression](#).

The **Matches/Does Not Match** option determines the conditions under which the URL Query Regex match

condition is met.

- **Matches:** Requires the request to contain a URL query string that matches the specified regular expression.
- **Does Not Match:** Requires the request to contain a URL query string that does not match the specified regular expression.

Key information:

- Only exact matches to the specified regular expression satisfy this match condition.
- Use the **Ignore Case** option to control the case-sensitivity of query string comparisons.
- For the purposes of this option, a query string starts with the first character after the question mark (?) delimiter for the query string.
- Certain characters require URL encoding. Use the percentage symbol to URL encode the following characters:

CHARACTER	URL ENCODING	VALUE
Space	%20	%20
&	%25	%25

Note that percentage symbols must be escaped.

- Double-escape special regular expression characters (for example, ^\$.+) to include a backslash in the regular expression.

For example:

VALUE	INTERPRETED AS
\+	+
\\\+	\+

- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

[Back to top](#)

---

## URL Query Wildcard

Compares the specified value(s) against the request's query string.

The **Matches/Does Not Match** option determines the conditions under which the URL Query Wildcard match condition is met.

- **Matches:** Requires the request to contain a URL query string that matches the specified wildcard value.

- **Does Not Match:** Requires the request to contain a URL query string that does not match the specified wildcard value.

Key information:

- For the purposes of this option, a query string starts with the first character after the question mark (?) delimiter for the query string.
- Parameter values can include [wildcard values](#):
  - Each parameter value pattern can consist of one or more asterisks (\*), where each asterisk can match a sequence of one or more characters.
  - Certain characters require URL encoding. Use the percentage symbol to URL encode the following characters:

CHARACTER	URL ENCODING
Space	%20
&	%25

- Specify multiple values by delimiting each one with a single space.

For example: *Parameter1=ValueA ValueB Parameter1=ValueC&Parameter2=ValueD*

- Only exact matches to at least one of the specified query string patterns satisfy this match condition.
- Use the **Ignore Case** option to control the case-sensitivity of query string comparisons.
- Due to the manner in which cache settings are tracked, this match condition is incompatible with the following features:
  - Complete Cache Fill
  - Default Internal Max-Age
  - Force Internal Max-Age
  - Ignore Origin No-Cache
  - Internal Max-Stale

#### Sample scenarios

The following example demonstrates how this option works in specific situations:

NAME	DESCRIPTION
user=joe	This pattern is matched when the query string for a requested URL is "?user=joe."
*user=* *optout=*	This pattern is matched when the CDN URL query contains either the user or optout parameter.

[Back to top](#)

## Next steps

- [Azure Content Delivery Network overview](#)
- [Rules engine reference](#)

- Rules engine conditional expressions
- Rules engine features
- Overriding default HTTP behavior using the rules engine

# Azure CDN from Verizon Premium rules engine features

7/5/2019 • 45 minutes to read • [Edit Online](#)

This article lists detailed descriptions of the available features for Azure Content Delivery Network (CDN) [Rules Engine](#).

The third part of a rule is the feature. A feature defines the type of action that is applied to the request type that is identified by a set of match conditions.

## Access features

These features are designed to control access to content.

NAME	PURPOSE
<a href="#">Deny Access (403)</a>	Determines whether all requests are rejected with a 403 Forbidden response.
<a href="#">Token Auth</a>	Determines whether Token-Based Authentication is applied to a request.
<a href="#">Token Auth Denial Code</a>	Determines the type of response that is returned to a user when a request is denied due to token-based authentication.
<a href="#">Token Auth Ignore URL Case</a>	Determines whether URL comparisons made by Token-Based Authentication are case-sensitive.
<a href="#">Token Auth Parameter</a>	Determines whether the Token-Based Authentication query string parameter should be renamed.

## Caching features

These features are designed to customize when and how content is cached.

NAME	PURPOSE
<a href="#">Bandwidth Parameters</a>	Determines whether bandwidth throttling parameters (for example, <code>ec_rate</code> and <code>ec_pbuf</code> ) are active.
<a href="#">Bandwidth Throttling</a>	Throttles the bandwidth for the response provided by the point-of-presence (POP).
<a href="#">Bypass Cache</a>	Determines whether the request should bypass caching.
<a href="#">Cache-Control Header Treatment</a>	Controls the generation of <code>Cache-Control</code> headers by the POP when External Max-Age feature is active.
<a href="#">Cache-Key Query String</a>	Determines whether the cache-key includes or excludes query string parameters associated with a request.

NAME	PURPOSE
Cache-Key Rewrite	Rewrites the cache-key associated with a request.
Complete Cache Fill	Determines what happens when a request results in a partial cache miss on a POP.
Compress File Types	Defines the file formats for the files that are compressed on the server.
Default Internal Max-Age	Determines the default max-age interval for POP to origin server cache revalidation.
Expires Header Treatment	Controls the generation of <code>Expires</code> headers by a POP when the External Max-Age feature is active.
External Max-Age	Determines the max-age interval for browser to POP cache revalidation.
Force Internal Max-Age	Determines the max-age interval for POP to origin server cache revalidation.
H.264 Support (HTTP Progressive Download)	Determines the types of H.264 file formats that may be used to stream content.
Honor No-Cache Request	Determines whether an HTTP client's no-cache requests are forwarded to the origin server.
Ignore Origin No-Cache	Determines whether the CDN ignores certain directives served from an origin server.
Ignore Unsatisfiable Ranges	Determines the response that is returned to clients when a request generates a 416 Requested Range Not Satisfiable status code.
Internal Max-Stale	Controls how long past the normal expiration time a cached asset may be served from a POP when the POP is unable to revalidate the cached asset with the origin server.
Partial Cache Sharing	Determines whether a request can generate partially cached content.
Prevalidate Cached Content	Determines whether cached content is eligible for early revalidation before its TTL expires.
Refresh Zero-Byte Cache Files	Determines how an HTTP client's request for a 0-byte cache asset is handled by the POPs.
Set Cacheable Status Codes	Defines the set of status codes that can result in cached content.
Stale Content Delivery on Error	Determines whether expired cached content is delivered when an error occurs during cache revalidation or when retrieving the requested content from the customer origin server.

NAME	PURPOSE
Stale While Revalidate	Improves performance by allowing the POPs to serve stale client to the requester while revalidation takes place.

## Comment feature

This feature is designed to provide additional information within a rule.

NAME	PURPOSE
Comment	Allows a note to be added within a rule.

## Header features

These features are designed to add, modify, or delete headers from the request or response.

NAME	PURPOSE
Age Response Header	Determines whether an Age response header is included in the response sent to the requester.
Debug Cache Response Headers	Determines whether a response may include the X-EC-Debug response header, which provides information on the cache policy for the requested asset.
Modify Client Request Header	Overwrites, appends, or deletes a header from a request.
Modify Client Response Header	Overwrites, appends, or deletes a header from a response.
Set Client IP Custom Header	Allows the IP address of the requesting client to be added to the request as a custom request header.

## Logging features

These features are designed to customize the data stored in raw log files.

NAME	PURPOSE
Custom Log Field 1	Determines the format and the content that is assigned to the custom log field in a raw log file.
Log Query String	Determines whether a query string is stored along with the URL in access logs.

## Origin features

These features are designed to control how the CDN communicates with an origin server.

NAME	PURPOSE
Maximum Keep-Alive Requests	Defines the maximum number of requests for a Keep-Alive connection before it is closed.

NAME	PURPOSE
Proxy Special Headers	Defines the set of CDN-specific request headers that are forwarded from a POP to an origin server.

## Specialty features

These features provide advanced functionality for advanced users.

NAME	PURPOSE
Cacheable HTTP Methods	Determines the set of additional HTTP methods that can be cached on the network.
Cacheable Request Body Size	Defines the threshold for determining whether a POST response can be cached.
User Variable	For internal use only.

## URL features

These features allow a request to be redirected or rewritten to a different URL.

NAME	PURPOSE
Follow Redirects	Determines whether requests can be redirected to the hostname defined in the Location header returned by a customer origin server.
URL Redirect	Redirects requests via the Location header.
URL Rewrite	Rewrites the request URL.

## Azure CDN from Verizon Premium rules engine features reference

### Age Response Header

**Purpose:** Determines whether an Age response header is included in the response sent to the requester.

VALUE	RESULT
Enabled	The Age response header is included in the response sent to the requester.
Disabled	The Age response header is excluded from the response sent to the requester.

**Default Behavior:** Disabled.

[Back to top](#)

---

### Bandwidth Parameters

**Purpose:** Determines whether bandwidth throttling parameters (for example, ec\_rate and ec\_pbuf) are active.

Bandwidth throttling parameters determine whether the data transfer rate for a client's request are limited to a custom rate.

VALUE	RESULT
Enabled	Allows the POPs to honor bandwidth throttling requests.
Disabled	Causes the POPs to ignore bandwidth throttling parameters. The requested content is served normally (that is, without bandwidth throttling).

**Default Behavior:** Enabled.

[Back to top](#)

---

## Bandwidth Throttling

**Purpose:** Throttles the bandwidth for the response provided by the POPs.

Both of the following options must be defined to properly set up bandwidth throttling.

OPTION	DESCRIPTION
Kbytes per second	Set this option to the maximum bandwidth (Kb per second) that may be used to deliver the response.
Prebuf seconds	Set this option to the number of seconds for the POPs to wait until bandwidth is throttled. The purpose of this time period of unrestricted bandwidth is to prevent a media player from experiencing stuttering or buffering issues due to bandwidth throttling.

**Default Behavior:** Disabled.

[Back to top](#)

---

## Bypass Cache

**Purpose:** Determines whether the request should bypass caching.

VALUE	RESULT
Enabled	Causes all requests to fall through to the origin server, even if the content was previously cached on POPs.
Disabled	Causes POPs to cache assets according to the cache policy defined in its response headers.

**Default Behavior:**

- **HTTP Large:** Disabled

[Back to top](#)

---

## Cacheable HTTP Methods

**Purpose:** Determines the set of additional HTTP methods that can be cached on the network.

Key information:

- This feature assumes that GET responses should always be cached. As a result, the GET HTTP method should not be included when setting this feature.
- This feature supports only the POST HTTP method. Enable POST response caching by setting this feature to `POST`.
- By default, only requests whose body is smaller than 14 Kb are cached. Use the Cacheable Request Body Size Feature to set the maximum request body size.

**Default Behavior:** Only GET responses are cached.

[Back to top](#)

---

## Cacheable Request Body Size

**Purpose:** Defines the threshold for determining whether a POST response can be cached.

This threshold is determined by specifying a maximum request body size. Requests that contain a larger request body are not cached.

Key information:

- This Feature is only applicable when POST responses are eligible for caching. Use the Cacheable HTTP Methods Feature to enable POST request caching.
- The request body is taken into consideration for:
  - `x-www-form-urlencoded` values
  - Ensuring a unique cache-key
- Defining a large maximum request body size may impact data delivery performance.
  - **Recommended Value:** 14 Kb
  - **Minimum Value:** 1 Kb

**Default Behavior:** 14 Kb

[Back to top](#)

---

## Cache-Control Header Treatment

**Purpose:** Controls the generation of `Cache-Control` headers by the POP when the External Max-Age Feature is active.

The easiest way to achieve this type of configuration is to place the External Max-Age and the Cache-Control Header Treatment features in the same statement.

VALUE	RESULT

VALUE	RESULT
Overwrite	Ensures that the following actions occur: - Overwrites the <code>Cache-Control</code> header generated by the origin server. - Adds the <code>Cache-Control</code> header produced by the External Max-Age feature to the response.
Pass Through	Ensures that the <code>Cache-Control</code> header produced by the External Max-Age feature is never added to the response. If the origin server produces a <code>Cache-Control</code> header, it passes through to the end user. If the origin server does not produce a <code>Cache-Control</code> header, then this option may cause the response header to not contain a <code>Cache-Control</code> header.
Add if Missing	If a <code>Cache-Control</code> header was not received from the origin server, then this option adds the <code>Cache-Control</code> header produced by the External Max-Age feature. This option is useful for ensuring that all assets are assigned a <code>Cache-Control</code> header.
Remove	This option ensures that a <code>Cache-Control</code> header is not included with the header response. If a <code>Cache-Control</code> header has already been assigned, then it is removed from the header response.

**Default Behavior:** Overwrite.

[Back to top](#)

## Cache-Key Query String

**Purpose:** Determines whether the cache-key includes or excludes query string parameters associated with a request.

Key information:

- Specify one or more query string parameter names and separate each parameter name with a single space.
  - This feature determines whether query string parameters are included or excluded from the cache-key.
- Additional information is provided for each option in the following table.

TYPE	DESCRIPTION
Include	Indicates that each specified parameter should be included in the cache-key. A unique cache-key is generated for each request that contains a unique value for a query string parameter defined in this feature.

TYPE	DESCRIPTION
Include All	Indicates that a unique cache-key is created for each request to an asset that includes a unique query string. This type of configuration is not typically recommended because it can lead to a small percentage of cache hits. A low number of cache hits increases the load on the origin server, because it must serve more requests. This configuration duplicates the caching behavior known as "unique-cache" on the Query-String Caching page.
Exclude	Indicates that only the specified parameter(s) is excluded from the cache-key. All other query string parameters are included in the cache-key.
Exclude All	Indicates that all query string parameters are excluded from the cache-key. This configuration duplicates the "standard-cache" default caching behavior on the Query-String Caching page.

The rules engine allows you to customize the manner in which query string caching is implemented. For example, you can specify that query string caching is performed only on certain locations or file types.

To duplicate the "no-cache" query string caching behavior on the Query-String Caching page, create a rule that contains a URL Query Wildcard match condition and a Bypass Cache feature. Set the URL Query Wildcard match condition to an asterisk (\*).

#### IMPORTANT

If token authorization is enabled for any path on this account, standard-cache mode is the only mode that can be used for query string caching. For more information, see [Control Azure CDN caching behavior with query strings](#).

#### Sample Scenarios

The following sample usage for this feature provides a sample request and the default cache-key:

- **Sample request:** `http://wpc.0001.<Domain>/800001/Origin/folder/asset.htm?sessionid=1234&language=EN&userid=01`
- **Default cache-key:** `/800001/Origin/folder/asset.htm`

#### Include

Sample configuration:

- **Type:** Include
- **Parameter(s):** language

This type of configuration would generate the following query string parameter cache-key:

```
/800001/Origin/folder/asset.htm?language=EN
```

#### Include All

Sample configuration:

- **Type:** Include All

This type of configuration would generate the following query string parameter cache-key:

```
/800001/Origin/folder/asset.htm?sessionid=1234&language=EN&userid=01
```

#### Exclude

Sample configuration:

- **Type:** Exclude
- **Parameter(s):** sessioned userid

This type of configuration would generate the following query string parameter cache-key:

```
/800001/Origin/folder/asset.htm?language=EN
```

#### Exclude All

Sample configuration:

- **Type:** Exclude All

This type of configuration would generate the following query string parameter cache-key:

```
/800001/Origin/folder/asset.htm
```

[Back to top](#)

---

## Cache-Key Rewrite

**Purpose:** Rewrites the cache-key associated with a request.

A cache-key is the relative path that identifies an asset for the purposes of caching. In other words, the servers check for a cached version of an asset according to its path as defined by its cache-key.

Configure this feature by defining both of the following options:

OPTION	DESCRIPTION
Original Path	Define the relative path to the types of requests whose cache-key is rewritten. A relative path can be defined by selecting a base origin path and then defining a regular expression pattern.
New Path	Define the relative path for the new cache-key. A relative path can be defined by selecting a base origin path and then defining a regular expression pattern. This relative path can be dynamically constructed through the use of <a href="#">HTTP variables</a> .

**Default Behavior:** A request's cache-key is determined by the request URI.

[Back to top](#)

---

## Comment

**Purpose:** Allows a note to be added within a rule.

One use for this feature is to provide additional information on the general purpose of a rule or why a particular match condition or feature was added to the rule.

Key information:

- A maximum of 150 characters may be specified.
- Use only alphanumeric characters.
- This feature does not affect the behavior of the rule. It is merely meant to provide an area where you can provide information for future reference or that may help when troubleshooting the rule.

[Back to top](#)

---

## Complete Cache Fill

**Purpose:** Determines what happens when a request results in a partial cache miss on a POP.

A partial cache miss describes the cache status for an asset that was not completely downloaded to a POP. If an asset is only partially cached on a POP, then the next request for that asset will be forwarded again to the origin server.

A partial cache miss typically occurs after a user aborts a download or for assets that are solely requested using HTTP range requests. This feature is most useful for large assets that are not typically downloaded from start to finish (for example, videos). As a result, this feature is enabled by default on the HTTP Large platform. It is disabled on all other platforms.

Keep the default configuration for the HTTP Large platform, because it reduces the load on your customer origin server and increases the speed at which your customers download your content.

VALUE	RESULT
Enabled	Restores the default behavior. The default behavior is to force the POP to initiate a background fetch of the asset from the origin server. After which, the asset will be in the POP's local cache.
Disabled	Prevents a POP from performing a background fetch for the asset. The result is that the next request for that asset from that region causes a POP to request it from the customer origin server.

**Default Behavior:** Enabled.

### Compatibility

Due to the manner in which cache settings are tracked, this feature cannot be associated with the following match conditions:

- AS Number
- Client IP Address
- Cookie Parameter
- Cookie Parameter Regex
- Country
- Device
- Microsoft Edge Cname
- Referring Domain
- Request Header Literal
- Request Header Regex
- Request Header Wildcard

- Request Method
- Request Scheme
- URL Query Literal
- URL Query Regex
- URL Query Wildcard
- URL Query Parameter

[Back to top](#)

---

## Compress File Types

**Purpose:** Defines the file formats for the files that are compressed on the server.

A file format can be specified using its Internet media type (for example, Content-Type). Internet media type is platform-independent metadata that allows the servers to identify the file format of a particular asset. A list of common Internet media types is provided below.

INTERNET MEDIA TYPE	DESCRIPTION
text/plain	Plain text files
text/html	HTML files
text/css	Cascading Style Sheets (CSS)
application/x-javascript	Javascript
application/javascript	Javascript

Key information:

- Specify multiple Internet media types by delimiting each one with a single space.
- This feature only compresses assets whose size is less than 1 MB. Larger assets are not compressed by the servers.
- Certain types of content, such as images, video, and audio media assets (for example, JPG, MP3, MP4, etc.), are already compressed. Because additional compression on these types of assets does not significantly diminish file size, it is recommended that you do not enable compression on them.
- Wildcard characters, such as asterisks, are not supported.
- Before you add this feature to a rule, ensure that you set the Compression Disabled option on the Compression page for the platform to which this rule is applied.

[Back to top](#)

---

## Custom Log Field 1

**Purpose:** Determines the format and the content that will be assigned to the custom log field in a raw log file.

This custom field allows you to determine which request and response header values are stored in your log files.

By default, the custom log field is called "x-ec\_custom-1." The name of this field can be customized from the Raw Log Settings page.

The format for specifying request and response headers is defined as follows:

HEADER TYPE	FORMAT	EXAMPLES
Request Header	<code>%{[RequestHeader]()}[i]()</code>	<code>%{Accept-Encoding}i</code> <code>{Referrer}i</code> <code>%{Authorization}i</code>
Response Header	<code>%{[ResponseHeader]()}[o]()</code>	<code>%{Age}o</code> <code>%{Content-Type}o</code> <code>%{Cookie}o</code>

Key information:

- A custom log field can contain any combination of header fields and plain text.
- Valid characters for this field are as follows: alphanumeric (0-9, a-z, and A-Z), dashes, colons, semi-colons, apostrophes, commas, periods, underscores, equal signs, parentheses, brackets, and spaces. The percentage symbol and curly braces are only allowed when used to specify a header field.
- The spelling for each specified header field must match the desired request/response header name.
- If you want to specify multiple headers, use a separator to indicate each header. For example, you could use an abbreviation for each header:
  - AE: `%{Accept-Encoding}i` A: `%{Authorization}i` CT: `%{Content-Type}o`

**Default Value:** -

[Back to top](#)

## Debug Cache Response Headers

**Purpose:** Determines whether a response can include [X-EC-Debug response headers](#), which provides information on the cache policy for the requested asset.

Debug cache response headers will be included in the response when both of the following are true:

- The Debug Cache Response Headers feature has been enabled on the specified request.
- The specified request defines the set of debug cache response headers that will be included in the response.

Debug cache response headers may be requested by including the following header and the specified directives in the request:

```
X-EC-Debug: <Directive1>;<Directive2>;...<DirectiveN>;
```

### Example:

`X-EC-Debug: x-ec-cache,x-ec-check-cacheable,x-ec-cache-key,x-ec-cache-state`

VALUE	RESULT
Enabled	Requests for debug cache response headers will return a response that includes the X-EC-Debug header.
Disabled	The X-EC-Debug response header will be excluded from the response.

**Default Behavior:** Disabled.

[Back to top](#)

---

## Default Internal Max-Age

**Purpose:** Determines the default max-age interval for POP to origin server cache revalidation. In other words, the amount of time that will pass before a POP will check whether a cached asset matches the asset stored on the origin server.

Key information:

- This action will only take place for responses from an origin server that did not assign a max-age indication in the `Cache-Control` or `Expires` header.
- This action will not take place for assets that are not deemed cacheable.
- This action does not affect browser to POP cache revalidations. These types of revalidations are determined by the `Cache-Control` or `Expires` headers sent to the browser, which can be customized with the External Max-Age feature.
- The results of this action do not have an observable effect on the response headers and the content returned from POPs for your content, but it may have an effect on the amount of revalidation traffic sent from POPs to your origin server.
- Configure this feature by:
  - Selecting the status code for which a default internal max-age can be applied.
  - Specifying an integer value and then selecting the desired time unit (for example, seconds, minutes, hours, etc.). This value defines the default internal max-age interval.
- Setting the time unit to "Off" will assign a default internal max-age interval of 7 days for requests that have not been assigned a max-age indication in their `Cache-Control` or `Expires` header.

**Default Value:** 7 days

### Compatibility

Due to the manner in which cache settings are tracked, this feature cannot be associated with the following match conditions:

- AS Number
- Client IP Address
- Cookie Parameter
- Cookie Parameter Regex
- Country
- Device
- Edge Cname
- Referring Domain
- Request Header Literal
- Request Header Regex
- Request Header Wildcard
- Request Method
- Request Scheme
- URL Query Literal
- URL Query Regex
- URL Query Wildcard
- URL Query Parameter

[Back to top](#)

## Deny Access (403)

**Purpose:** Determines whether all requests are rejected with a 403 Forbidden response.

VALUE	RESULT
Enabled	Causes all requests that satisfy the matching criteria to be rejected with a 403 Forbidden response.
Disabled	Restores the default behavior. The default behavior is to allow the origin server to determine the type of response that will be returned.

**Default Behavior:** Disabled

### TIP

One possible use for this feature is to associate it with a Request Header match condition to block access to HTTP referrers that are using inline links to your content.

[Back to top](#)

## Expires Header Treatment

**Purpose:** Controls the generation of `Expires` headers by a POP when the External Max-Age feature is active.

The easiest way to achieve this type of configuration is to place the External Max-Age and the Expires Header Treatment features in the same statement.

VALUE	RESULT
Overwrite	Ensures that the following actions will take place: <ul style="list-style-type: none"><li>- Overwrites the <code>Expires</code> header generated by the origin server.</li><li>- Adds the <code>Expires</code> header produced by the External Max-Age feature to the response.</li></ul>
Pass Through	Ensures that the <code>Expires</code> header produced by the External Max-Age feature is never added to the response. If the origin server produces an <code>Expires</code> header, it will pass through to the end user. If the origin server does not produce an <code>Expires</code> header, then this option may cause the response header to not contain an <code>Expires</code> header.
Add if Missing	If an <code>Expires</code> header was not received from the origin server, then this option adds the <code>Expires</code> header produced by the External Max-Age feature. This option is useful for ensuring that all assets will be assigned an <code>Expires</code> header.
Remove	Ensures that an <code>Expires</code> header is not included with the header response. If an <code>Expires</code> header has already been assigned, then it is removed from the header response.

**Default Behavior:** Overwrite

[Back to top](#)

---

### External Max-Age

**Purpose:** Determines the max-age interval for browser to POP cache revalidation. In other words, the amount of time that will pass before a browser can check for a new version of an asset from a POP.

Enabling this feature will generate `Cache-Control: max-age` and `Expires` headers from the POPs and send them to the HTTP client. By default, these headers will overwrite those headers created by the origin server. However, the Cache-Control Header Treatment and the Expires Header Treatment features may be used to alter this behavior.

Key information:

- This action does not affect POP to origin server cache revalidations. These types of revalidations are determined by the `Cache-Control` and `Expires` headers received from the origin server, and can be customized with the Default Internal Max-Age and the Force Internal Max-Age features.
- Configure this feature by specifying an integer value and selecting the desired time unit (for example, seconds, minutes, hours, etc.).
- Setting this feature to a negative value causes the POPs to send a `Cache-Control: no-cache` and an `Expires` time that is set in the past with each response to the browser. Although an HTTP client will not cache the response, this setting will not affect the POPs' ability to cache the response from the origin server.
- Setting the time unit to "Off" will disable this feature. The `Cache-Control` and `Expires` headers cached with the response of the origin server will pass through to the browser.

**Default Behavior:** Off

[Back to top](#)

---

### Follow Redirects

**Purpose:** Determines whether requests can be redirected to the hostname defined in the Location header returned by a customer origin server.

Key information:

- Requests can only be redirected to edge CNAMEs that correspond to the same platform.

VALUE	RESULT
Enabled	Requests can be redirected.
Disabled	Requests will not be redirected.

**Default Behavior:** Disabled.

[Back to top](#)

---

### Force Internal Max-Age

**Purpose:** Determines the max-age interval for POP to origin server cache revalidation. In other words, the

amount of time that will pass before a POP can check whether a cached asset matches the asset stored on the origin server.

Key information:

- This feature will override the max-age interval defined in `Cache-Control` or `Expires` headers generated from an origin server.
- This feature does not affect browser to POP cache revalidations. These types of revalidations are determined by the `Cache-Control` or `Expires` headers sent to the browser.
- This feature does not have an observable effect on the response delivered by a POP to the requester. However, it may have an effect on the amount of revalidation traffic sent from the POPs to the origin server.
- Configure this feature by:
  - Selecting the status code for which an internal max-age will be applied.
  - Specifying an integer value and selecting the desired time unit (for example, seconds, minutes, hours, etc.). This value defines the request's max-age interval.
- Setting the time unit to "Off" disables this feature. An internal max-age interval will not be assigned to requested assets. If the original header does not contain caching instructions, then the asset will be cached according to the active setting in the Default Internal Max-Age feature.

**Default Behavior:** Off

#### Compatibility

Due to the manner in which cache settings are tracked, this feature cannot be associated with the following match conditions:

- AS Number
- Client IP Address
- Cookie Parameter
- Cookie Parameter Regex
- Country
- Device
- Edge Cname
- Referring Domain
- Request Header Literal
- Request Header Regex
- Request Header Wildcard
- Request Method
- Request Scheme
- URL Query Literal
- URL Query Regex
- URL Query Wildcard
- URL Query Parameter

[Back to top](#)

---

#### H.264 Support (HTTP Progressive Download)

**Purpose:** Determines the types of H.264 file formats that may be used to stream content.

## Key information:

- Define a space-delimited set of allowed H.264 filename extensions in the File Extensions option. The File Extensions option will override the default behavior. Maintain MP4 and F4V support by including those filename extensions when setting this option.
- Include a period when you specify each filename extension (for example, `.mp4`, `.f4v`).

**Default Behavior:** HTTP Progressive Download supports MP4 and F4V media by default.

[Back to top](#)

---

## Honor No-Cache Request

**Purpose:** Determines whether an HTTP client's no-cache requests will be forwarded to the origin server.

A no-cache request occurs when the HTTP client sends a `Cache-Control: no-cache` and/or `Pragma: no-cache` header in the HTTP request.

VALUE	RESULT
Enabled	Allows an HTTP client's no-cache requests to be forwarded to the origin server, and the origin server will return the response headers and the body through the POP back to the HTTP client.
Disabled	Restores the default behavior. The default behavior is to prevent no-cache requests from being forwarded to the origin server.

For all production traffic, it is highly recommended to leave this feature in its default disabled state. Otherwise, origin servers will not be shielded from end users who may inadvertently trigger many no-cache requests when refreshing web pages, or from the many popular media players that are coded to send a no-cache header with every video request. Nevertheless, this feature can be useful to apply to certain non-production staging or testing directories, in order to allow fresh content to be pulled on-demand from the origin server.

The cache status that is reported for a request that can be forwarded to an origin server due to this feature is `TCP_Client_Refesh_Miss`. The Cache Statuses report, which is available in the Core reporting module, provides statistical information by cache status. This report allows you to track the number and percentage of requests that are being forwarded to an origin server due to this feature.

**Default Behavior:** Disabled.

[Back to top](#)

---

## Ignore Origin No-Cache

**Purpose:** Determines whether the CDN will ignore the following directives served from an origin server:

- `Cache-Control: private`
- `Cache-Control: no-store`
- `Cache-Control: no-cache`
- `Pragma: no-cache`

Key information:

- Configure this feature by defining a space-delimited list of status codes for which the above directives will be ignored.
- The set of valid status codes for this feature are: 200, 203, 300, 301, 302, 305, 307, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 500, 501, 502, 503, 504, and 505.
- Disable this feature by setting it to a blank value.

**Default Behavior:** The default behavior is to honor the above directives.

#### Compatibility

Due to the manner in which cache settings are tracked, this feature cannot be associated with the following match conditions:

- AS Number
- Client IP Address
- Cookie Parameter
- Cookie Parameter Regex
- Country
- Device
- Edge Cname
- Referring Domain
- Request Header Literal
- Request Header Regex
- Request Header Wildcard
- Request Method
- Request Scheme
- URL Query Literal
- URL Query Regex
- URL Query Wildcard
- URL Query Parameter

[Back to top](#)

## Ignore Unsatisfiable Ranges

**Purpose:** Determines the response that will be returned to clients when a request generates a 416 Requested Range Not Satisfiable status code.

By default, this status code is returned when the specified byte-range request cannot be satisfied by a POP and an If-Range request header field was not specified.

VALUE	RESULT
Enabled	Prevents the POPs from responding to an invalid byte-range request with a 416 Requested Range Not Satisfiable status code. Instead the servers will deliver the requested asset and return a 200 OK to the client.
Disabled	Restores the default behavior. The default behavior is to honor the 416 Requested Range Not Satisfiable status code.

**Default Behavior:** Disabled.

[Back to top](#)

---

## Internal Max-Stale

**Purpose:** Controls how long past the normal expiration time a cached asset may be served from a POP when the POP is unable to revalidate the cached asset with the origin server.

Normally, when an asset's max-age time expires, the POP will send a revalidation request to the origin server. The origin server will then respond with either a 304 Not Modified to give the POP a fresh lease on the cached asset, or else with 200 OK to provide the POP with an updated version of the cached asset.

If the POP is unable to establish a connection with the origin server while attempting such a revalidation, then this Internal Max-Stale feature controls whether, and for how long, the POP may continue to serve the now-stale asset.

Note that this time interval starts when the asset's max-age expires, not when the failed revalidation occurs.

Therefore, the maximum period during which an asset can be served without successful revalidation is the amount of time specified by the combination of max-age plus max-stale. For example, if an asset was cached at 9:00 with a max-age of 30 minutes and a max-stale of 15 minutes, then a failed revalidation attempt at 9:44 would result in an end user receiving the stale cached asset, while a failed revalidation attempt at 9:46 would result in the end user receiving a 504 Gateway Timeout.

Any value configured for this feature is superseded by `Cache-Control: must-revalidate` or `Cache-Control: proxy-revalidate` headers received from the origin server. If either of those headers is received from the origin server when an asset is initially cached, then the POP will not serve a stale cached asset. In such a case, if the POP is unable to revalidate with the origin when the asset's max-age interval has expired, the POP returns a 504 Gateway Timeout error.

Key information:

- Configure this feature by:
  - Selecting the status code for which a max-stale will be applied.
  - Specifying an integer value and then selecting the desired time unit (for example, seconds, minutes, hours, etc.). This value defines the internal max-stale that will be applied.
- Setting the time unit to "Off" will disable this feature. A cached asset will not be served beyond its normal expiration time.

**Default Behavior:** Two minutes

### Compatibility

Due to the manner in which cache settings are tracked, this feature cannot be associated with the following match conditions:

- AS Number
- Client IP Address
- Cookie Parameter
- Cookie Parameter Regex
- Country
- Device
- Edge Cname
- Referring Domain
- Request Header Literal
- Request Header Regex
- Request Header Wildcard
- Request Method

- Request Scheme
- URL Query Literal
- URL Query Regex
- URL Query Wildcard
- URL Query Parameter

[Back to top](#)

---

## Log Query String

**Purpose:** Determines whether a query string will be stored along with the URL in access logs.

VALUE	RESULT
Enabled	Allows the storage of query strings when recording URLs in an access log. If a URL does not contain a query string, then this option will not have an effect.
Disabled	Restores the default behavior. The default behavior is to ignore query strings when recording URLs in an access log.

**Default Behavior:** Disabled.

[Back to top](#)

---

## Maximum Keep-Alive Requests

**Purpose:** Defines the maximum number of requests for a Keep-Alive connection before it is closed.

Setting the maximum number of requests to a low value is discouraged and may result in performance degradation.

Key information:

- Specify this value as a whole integer.
- Do not include commas or periods in the specified value.

**Default Value:** 10,000 requests

[Back to top](#)

---

## Modify Client Request Header

**Purpose:** Each request contains a set of request headers that describe it. This feature can either:

- Append or overwrite the value assigned to a request header. If the specified request header does not exist, then this feature will add it to the request.
- Delete a request header from the request.

Requests that are forwarded to an origin server will reflect the changes made by this feature.

One of the following actions can be performed on a request header:

OPTION	DESCRIPTION	EXAMPLE
Append	The specified value will be added to the end of the existing request header value.	<b>Request header value (client):</b> Value1 <b>Request header value (rules engine):</b> Value2 <b>New request header value:</b> Value1Value2
Overwrite	The request header value will be set to the specified value.	<b>Request header value (client):</b> Value1 <b>Request header value (rules engine):</b> Value2 <b>New request header value:</b> Value2
Delete	Deletes the specified request header.	<b>Request header value (client):</b> Value1 <b>Modify client request header configuration:</b> Delete the request header in question. <b>Result:</b> The specified request header will not be forwarded to the origin server.

Key information:

- Ensure that the value specified in the Name option is an exact match for the desired request header.
- Case is not taken into account for the purpose of identifying a header. For example, any of the following variations of the `Cache-Control` header name can be used to identify it:
  - cache-control
  - CACHE-CONTROL
  - cachE-Control
- When specifying a header name, use only alphanumeric characters, dashes, or underscores.
- Deleting a header will prevent it from being forwarded to an origin server by the POPs.
- The following headers are reserved and cannot be modified by this feature:
  - forwarded
  - host
  - via
  - warning
  - x-forwarded-for
  - All header names that start with "x-ec" are reserved.

[Back to top](#)

## Modify Client Response Header

Each response contains a set of response headers that describe it. This feature can either:

- Append or overwrite the value assigned to a response header. If the specified response header does not exist, then this feature will add it to the response.
- Delete a response header from the response.

By default, response header values are defined by an origin server and by the POPs.

One of the following actions can be performed on a response header:

OPTION	DESCRIPTION	EXAMPLE
Append	The specified value will be added to the end of the existing response header value.	<b>Response header value (client):</b> Value1 <b>Response header value (rules engine):</b> Value2 <b>New response header value:</b> Value1Value2
Overwrite	The response header value will be set to the specified value.	<b>Response header value (client):</b> Value1 <b>Response header value (rules engine):</b> Value2 <b>New response header value:</b> Value2
Delete	Deletes the specified response header.	<b>Response header value (client):</b> Value1 <b>Modify client response header configuration:</b> Delete the response header in question. <b>Result:</b> The specified response header will not be forwarded to the requester.

Key information:

- Ensure that the value specified in the Name option is an exact match for the desired response header.
- Case is not taken into account for the purpose of identifying a header. For example, any of the following variations of the `Cache-Control` header name can be used to identify it:
  - cache-control
  - CACHE-CONTROL
  - cachE-Control
- Deleting a header prevents it from being forwarded to the requester.
- The following headers are reserved and cannot be modified by this feature:
  - accept-encoding
  - age
  - connection
  - content-encoding
  - content-length
  - content-range
  - date
  - server
  - trailer
  - transfer-encoding
  - upgrade
  - vary
  - via
  - warning
  - All header names that start with "x-ec" are reserved.

[Back to top](#)

---

### Partial Cache Sharing

**Purpose:** Determines whether a request can generate partially cached content.

This partial cache may then be used to fulfill new requests for that content until the requested content is fully cached.

VALUE	RESULT
Enabled	Requests can generate partially cached content.
Disabled	Requests can only generate a fully cached version of the requested content.

**Default Behavior:** Disabled.

[Back to top](#)

---

### Prevalidate Cached Content

**Purpose:** Determines whether cached content will be eligible for early revalidation before its TTL expires.

Define the amount of time prior to the expiration of the requested content's TTL during which it will be eligible for early revalidation.

Key information:

- Selecting "Off" as the time unit requires revalidation to take place after the cached content's TTL has expired. Time should not be specified and is ignored.

**Default Behavior:** Off. Revalidation may only take place after the cached content's TTL has expired.

[Back to top](#)

---

### Proxy Special Headers

**Purpose:** Defines the set of [Verizon-specific HTTP request headers](#) that will be forwarded from a POP to an origin server.

Key information:

- Each CDN-specific request header defined in this feature is forwarded to an origin server. Excluded headers are not forwarded.
- To prevent a CDN-specific request header from being forwarded, remove it from space-separated list in the header list field.

The following HTTP headers are included in the default list:

- Via
- X-Forwarded-For
- X-Forwarded-Proto
- X-Host

- X-Midgress
- X-Gateway-List
- X-EC-Name
- Host

**Default Behavior:** All CDN-specific request headers will be forwarded to the origin server.

[Back to top](#)

---

### Refresh Zero-Byte Cache Files

**Purpose:** Determines how an HTTP client's request for a 0-byte cache asset is handled by the POPs.

Valid values are:

VALUE	RESULT
Enabled	Causes the POP to refetch the asset from the origin server.
Disabled	Restores the default behavior. The default behavior is to serve up valid cache assets upon request.

This feature is not required for correct caching and content delivery, but may be useful as a workaround. For example, dynamic content generators on origin servers can inadvertently result in 0-byte responses being sent to the POPs. These types of responses are typically cached by the POPs. If you know that a 0-byte response is never a valid response for such content, this feature can prevent these types of assets from being served to your clients.

**Default Behavior:** Disabled.

[Back to top](#)

---

### Set Cacheable Status Codes

**Purpose:** Defines the set of status codes that can result in cached content.

By default, caching is only enabled for 200 OK responses.

Define a space-delimited set of the desired status codes.

Key information:

- Enable the Ignore Origin No-Cache feature. If this feature is not enabled, then non-200 OK responses may not be cached.
- The set of valid status codes for this feature are: 203, 300, 301, 302, 305, 307, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 500, 501, 502, 503, 504, and 505.
- This feature cannot be used to disable caching for responses that generate a 200 OK status code.

**Default Behavior:** Caching is enabled only for responses that generate a 200 OK status code.

[Back to top](#)

---

### Set Client IP Custom Header

**Purpose:** Adds a custom header that identifies the requesting client by IP address to the request.

The Header name option defines the name of the custom request header where the client's IP address is stored.

This feature allows a customer origin server to find out client IP addresses through a custom request header. If the request is served from cache, then the origin server will not be informed of the client's IP address. Therefore, it is recommended that this feature be used with assets that aren't cached.

Ensure that the specified header name does not match any of the following names:

- Standard request header names. A list of standard header names can be found in [RFC 2616](#).
- Reserved header names:
  - forwarded-for
  - host
  - vary
  - via
  - warning
  - x-forwarded-for
  - All header names that start with "x-ec" are reserved.

[Back to top](#)

---

### Stale Content Delivery on Error

**Purpose:** Determines whether expired cached content will be delivered when an error occurs during cache revalidation or when retrieving the requested content from the customer origin server.

VALUE	RESULT
Enabled	Stale content is served to the requester when an error occurs during a connection to an origin server.
Disabled	The origin server's error is forwarded to the requester.

**Default Behavior:** Disabled

[Back to top](#)

---

### Stale While Revalidate

**Purpose:** Improves performance by allowing the POPs to serve stale content to the requester while revalidation takes place.

Key information:

- The behavior of this feature varies according to the selected time unit.
  - **Time Unit:** Specify a length of time and select a time unit (for example, Seconds, Minutes, Hours, etc.) to allow stale content delivery. This type of setup allows the CDN to extend the length of time that it may deliver content before requiring validation according to the following formula: **TTL + Stale While Revalidate Time**
  - **Off:** Select "Off" to require revalidation before a request for stale content may be served.
    - Do not specify a length of time since it is inapplicable and will be ignored.

**Default Behavior:** Off. Revalidation must take place before the requested content can be served.

## Token Auth

**Purpose:** Determines whether Token-Based Authentication will be applied to a request.

If Token-Based Authentication is enabled, then only requests that provide an encrypted token and comply to the requirements specified by that token will be honored.

The encryption key that is used to encrypt and decrypt token values is determined by the Primary Key and the Backup Key options on the Token Auth page. Keep in mind that encryption keys are platform-specific.

**Default Behavior:** Disabled.

This feature takes precedence over most features with the exception of the URL Rewrite feature.

VALUE	RESULT
Enabled	Protects the requested content with Token-Based Authentication. Only requests from clients that provide a valid token and meet its requirements will be honored. FTP transactions are excluded from Token-Based Authentication.
Disabled	Restores the default behavior. The default behavior is to allow your Token-Based Authentication configuration to determine whether a request will be secured.

### Compatibility

Do not use Token Auth with an Always match condition.

## Token Auth Denial Code

**Purpose:** Determines the type of response that will be returned to a user when a request is denied due to token-based authentication.

The available response codes are listed in the following table.

RESPONSE CODE	RESPONSE NAME	DESCRIPTION
301	Moved Permanently	This status code redirects unauthorized users to the URL specified in the Location header.
302	Found	This status code redirects unauthorized users to the URL specified in the Location header. This status code is the industry standard method of performing a redirect.
307	Temporary Redirect	This status code redirects unauthorized users to the URL specified in the Location header.

Response Code	Response Name	Description
401	Unauthorized	Combining this status code with the WWW-Authenticate response header allows you to prompt a user for authentication.
403	Forbidden	This message is the standard 403 Forbidden status message that an unauthorized user will see when trying to access protected content.
404	File Not Found	This status code indicates that the HTTP client was able to communicate with the server, but the requested content was not found.

#### Compatibility

Do not use Token Auth Denial Code with an Always match condition. Instead, use the **Custom Denial Handling** section in the **Token Auth** page of the **Manage** portal. For more information, see [Securing Azure CDN assets with token authentication](#).

#### URL Redirection

This feature supports URL redirection to a user-defined URL when it is configured to return a 3xx status code. This user-defined URL can be specified by performing the following steps:

1. Select a 3xx response code for the Token Auth Denial Code feature.
2. Select "Location" from the Optional Header Name option.
3. Set the Optional Header Value option to the desired URL.

If a URL is not defined for a 3xx status code, then the standard response page for a 3xx status code will be returned to the user.

URL redirection is only applicable for 3xx response codes.

The Optional Header Value option supports alphanumeric characters, quotation marks, and spaces.

#### Authentication

This feature supports the capability to include the WWW-Authenticate header when responding to an unauthorized request for content protected by Token-Based Authentication. If the WWW-Authenticate header has been set to "basic" in your configuration, then the unauthorized user will be prompted for account credentials.

The above configuration can be achieved by performing the following steps:

1. Select "401" as the response code for the Token Auth Denial Code feature.
2. Select "WWW-Authenticate" from the Optional Header Name option.
3. Set the Optional Header Value option to "basic."

The WWW-Authenticate header is only applicable for 401 response codes.

[Back to top](#)

---

#### Token Auth Ignore URL Case

**Purpose:** Determines whether URL comparisons made by Token-Based Authentication are case-sensitive.

The parameters affected by this feature are:

- ec\_url\_allow
- ec\_ref\_allow
- ec\_ref\_deny

Valid values are:

VALUE	RESULT
Enabled	Causes the POP to ignore case when comparing URLs for Token-Based Authentication parameters.
Disabled	Restores the default behavior. The default behavior is for URL comparisons for Token Authentication to be case-sensitive.

**Default Behavior:** Disabled.

[Back to top](#)

## Token Auth Parameter

**Purpose:** Determines whether the Token-Based Authentication query string parameter should be renamed.

Key information:

- The Value option defines the query string parameter name through which a token may be specified.
- The Value option cannot be set to "ec\_token."
- Ensure that the name defined in the Value option contains only valid URL characters.

VALUE	RESULT
Enabled	The Value option defines the query string parameter name through which tokens should be defined.
Disabled	A token may be specified as an undefined query string parameter in the request URL.

**Default Behavior:** Disabled. A token may be specified as an undefined query string parameter in the request URL.

[Back to top](#)

## URL Redirect

**Purpose:** Redirects requests via the Location header.

The configuration of this feature requires setting the following options:

OPTION	DESCRIPTION
Code	Select the response code that will be returned to the requester.

OPTION	DESCRIPTION
Source & Pattern	<p>These settings define a request URI pattern that identifies the type of requests that may be redirected. Only requests whose URL satisfies both of the following criteria will be redirected:</p> <p><b>Source (or content access point):</b> Select a relative path that identifies an origin server. This path is the /XXXX/ section and your endpoint name.</p> <p><b>Source (pattern):</b> A pattern that identifies requests by relative path must be defined. This regular expression pattern must define a path that starts directly after the previously selected content access point (see above).</p> <ul style="list-style-type: none"> <li>- Ensure that the request URI criteria (that is, Source &amp; Pattern) previously defined doesn't conflict with any match conditions defined for this feature.</li> <li>- Specify a pattern; if you use a blank value as the pattern, all strings are matched.</li> </ul>
Destination	<p>Define the URL to which the above requests will be redirected.</p> <p>Dynamically construct this URL using:</p> <ul style="list-style-type: none"> <li>- A regular expression pattern</li> <li>- <a href="#">HTTP variables</a></li> </ul> <p>Substitute the values captured in the source pattern into the destination pattern using \$n where n identifies a value by the order in which it was captured. For example, \$1 represents the first value captured in the source pattern, while \$2 represents the second value.</p>

It is highly recommended to use an absolute URL. The use of a relative URL may redirect CDN URLs to an invalid path.

### Sample Scenario

This example, demonstrates how to redirect an edge CNAME URL that resolves to this base CDN URL:  
<http://marketing.azureedge.net/brochures>

Qualifying requests will be redirected to this base edge CNAME URL: <http://cdn.mydomain.com/resources>

This URL redirection may be achieved through the following configuration:



### Key points:

- The URL Redirect feature defines the request URLs that will be redirected. As a result, additional match conditions are not required. Although the match condition was defined as "Always," only requests that point to the "brochures" folder on the "marketing" customer origin will be redirected.
- All matching requests will be redirected to the edge CNAME URL defined in the Destination option.
  - Sample scenario #1:
    - Sample request (CDN URL): <http://marketing.azureedge.net/brochures/widgets.pdf>
    - Request URL (after redirect): <http://cdn.mydomain.com/resources/widgets.pdf>
  - Sample scenario #2:
    - Sample request (Edge CNAME URL): <http://marketing.mydomain.com/brochures/widgets.pdf>

- Request URL (after redirect): <http://cdn.mydomain.com/resources/widgets.pdf> Sample scenario
- Sample scenario #3:
  - Sample request (Edge CNAME URL):
   
<http://brochures.mydomain.com/campaignA/final/productC.ppt>
  - Request URL (after redirect):
   
<http://cdn.mydomain.com/resources/campaignA/final/productC.ppt>
- The Request Scheme (%{scheme}) variable is leveraged in the Destination option, which ensures that the request's scheme remains unchanged after redirection.
- The URL segments that were captured from the request are appended to the new URL via "\$1."

[Back to top](#)

## URL Rewrite

**Purpose:** Rewrites the request URL.

Key information:

- The configuration of this feature requires setting the following options:

OPTION	DESCRIPTION
Source & Pattern	<p>These settings define a request URI pattern that identifies the type of requests that may be rewritten. Only requests whose URL satisfies both of the following criteria will be rewritten:</p> <ul style="list-style-type: none"> <li>- <b>Source (or content access point):</b> Select a relative path that identifies an origin server. This path is the /XXXX/ section and your endpoint name.</li> <li>- <b>Source (pattern):</b> A pattern that identifies requests by relative path must be defined. This regular expression pattern must define a path that starts directly after the previously selected content access point (see above). Verify that the request URI criteria (that is, Source &amp; Pattern) previously defined doesn't conflict with any of the match conditions defined for this feature. Specify a pattern; if you use a blank value as the pattern, all strings are matched.</li> </ul>
Destination	<p>Define the relative URL to which the above requests will be rewritten by:</p> <ol style="list-style-type: none"> <li>1. Selecting a content access point that identifies an origin server.</li> <li>2. Defining a relative path using:           <ul style="list-style-type: none"> <li>- A regular expression pattern</li> <li>- <a href="#">HTTP variables</a></li> </ul> </li> </ol> <p>Substitute the values captured in the source pattern into the destination pattern using \$n where n identifies a value by the order in which it was captured. For example, \$1 represents the first value captured in the source pattern, while \$2 represents the second value.</p>

This feature allows the POPs to rewrite the URL without performing a traditional redirect. That is, the requester receives the same response code as if the rewritten URL had been requested.

### Sample Scenario 1

This example demonstrates how to redirect an edge CNAME URL that resolves to this base CDN URL:  
<http://marketing.azureedge.net/brochures/>

Qualifying requests will be redirected to this base edge CNAME URL: <http://MyOrigin.azureedge.net/resources/>

This URL redirection may be achieved through the following configuration:



## Sample Scenario 2

This example demonstrates how to redirect an edge CNAME URL from UPPERCASE to lowercase using regular expressions.

This URL redirection may be achieved through the following configuration:



### Key points:

- The URL Rewrite feature defines the request URLs that will be rewritten. As a result, additional match conditions are not required. Although the match condition was defined as "Always," only requests that point to the "brochures" folder on the "marketing" customer origin will be rewritten.
- The URL segments that were captured from the request are appended to the new URL via "\$1."

### Compatibility

This feature includes matching criteria that must be met before it can be applied to a request. In order to prevent setting up conflicting match criteria, this feature is incompatible with the following match conditions:

- AS Number
- CDN Origin
- Client IP Address
- Customer Origin
- Request Scheme
- URL Path Directory
- URL Path Extension
- URL Path Filename
- URL Path Literal
- URL Path Regex
- URL Path Wildcard
- URL Query Literal
- URL Query Parameter
- URL Query Regex
- URL Query Wildcard

[Back to top](#)

---

### User Variable

**Purpose:** For internal use only.

[Back to top](#)

## Next steps

- [Rules engine reference](#)
- [Rules engine conditional expressions](#)
- [Rules engine match conditions](#)
- [Override HTTP behavior using the rules engine](#)
- [Azure CDN overview](#)

# HTTP variables for Azure CDN rules engine

7/5/2019 • 10 minutes to read • [Edit Online](#)

HTTP variables provide the means through which you can retrieve HTTP request and response metadata. This metadata can then be used to dynamically alter a request or a response. The use of HTTP variables is restricted to the following rules engine features:

- [Cache-Key Rewrite](#)
- [Modify Client Request Header](#)
- [Modify Client Response Header](#)
- [URL Redirect](#)
- [URL Rewrite](#)

## Definitions

The following table describes the supported HTTP variables. A blank value is returned when GEO metadata (for example, postal code) is unavailable for a particular request.

NAME	VARIABLE	DESCRIPTION	SAMPLE VALUE
ASN (Requester)	{geo_asnum}	Indicates the requester's AS number.  <b>Deprecated:</b> {virt_dst_asnum}. This variable has been deprecated in favor of {geo_asnum}. Although a rule that uses this deprecated variable will continue to work, you should update it to use the new variable.	AS15133
City (Requester)	{geo_city}	Indicates the requester's city.	Los Angeles

Name	Variable	Description	Sample Value
Continent (Requester)	{geo_continent}	<p>Indicates the requester's continent through its abbreviation.</p> <p>Valid values are:</p> <ul style="list-style-type: none"> <li>AF: Africa</li> <li>AS: Asia</li> <li>EU: Europe</li> <li>NA: North America</li> <li>OC: Oceania</li> <li>SA: South America</li> </ul> <p><b>Deprecated:</b> {virt_dst_continent}.</p> <p>This variable has been deprecated in favor of {geo_continent}. Although a rule that uses this deprecated variable will continue to work, you should update it to use the new variable.</p>	N/A
Cookie Value	{cookie_Cookie}	Returns the value corresponding to the cookie key identified by the Cookie term.	<p>Sample Usage: {cookie_utma}</p> <p>Sample Value: 111662281.2.10.12221001 23</p>
Country (Requester)	{geo_country}	<p>Indicates the requester's country of origin through its country code.</p> <p><b>Deprecated:</b> {virt_dst_country}.</p> <p>This variable has been deprecated in favor of {geo_country}. Although a rule that uses this deprecated variable will continue to work, you should update it to use the new variable.</p>	US
Designated Market Area (Requester)	{geo_dma_code}	<p>Indicates the requester's media market by its region code.</p> <p>This field is only applicable to requests that originate from the United States.</p>	745
HTTP Request Method	{request_method}	Indicates the HTTP request method.	GET
HTTP Status Code	{status}	Indicates the HTTP status code for the response.	200

Name	Variable	Description	Sample Value
IP Address (Requester)	%{virt_dst_addr}	Indicates the requester's IP address.	192.168.1.1
Latitude (Requester)	%{geo_latitude}	Indicates the requester's latitude.	34.0995
Longitude (Requester)	%{geo_longitude}	Indicates the requester's longitude.	-118.4143
Metropolitan Statistical Area (Requester)	%{geo_metro_code}	Indicates the requester's metropolitan area.  This field is only applicable to requests that originate from the United States.	745
Port (Requester)	%{virt_dst_port}	Indicates the requester's ephemeral port.	55885
Postal Code (Requester)	%{geo_postal_code}	Indicates the requester's postal code.	90210
Query String Found	%{is_args}	The value for this variable varies according to whether the request contains a query string.  - Query String Found: ? - No Query String: NULL	?
Query String Parameter Found	%{is_amp}	The value for this variable varies according to whether the request contains at least one query string parameter.  - Parameter Found: & - No Parameters: NULL	&
Query String Parameter Value	%{arg_<parameter>}	Returns the value corresponding to the query string parameter identified by the <parameter> term.	Sample Usage: %{arg_language}  Sample Query String Parameter: ?language=en  Sample Value: en
Query String Value	%{query_string}	Indicates the entire query string value defined in the request URL.	key1=val1&key2=val2&key3=val3
Referrer Domain	%{referring_domain}	Indicates the domain defined in the Referrer request header.	<www.google.com>

NAME	VARIABLE	DESCRIPTION	SAMPLE VALUE
Region (Requester)	%{geo_region}	Indicates the requester's region (for example, state or province) through its alphanumeric abbreviation.	CA
Request Header Value	%{http_RequestHeader}	Returns the value corresponding to the request header identified by the RequestHeader term.  If the name of the request header contains a dash (for example, User-Agent), replace it with an underscore (for example, User_Agent).	Sample Usage: %{http_Connection}  Sample Value: Keep-Alive
Request Host	%{host}	Indicates the host defined in the request URL.	<www.mydomain.com>
Request Protocol	%{request_protocol}	Indicates the request protocol.	HTTP/1.1
Request Scheme	%{scheme}	Indicates the request scheme.	http
Request URI (Relative)	%{request_uri}	Indicates the relative path, including the query string, defined in the request URI.	/marketing/foo.js? loggedin=true
Request URI (Relative without query string)	%{uri}	Indicates the relative path to the requested content.  Key information: - This relative path excludes the query string. - This relative path reflects URL rewrites. A URL will be rewritten under the following conditions: - URL Rewrite Feature: This feature rewrites the relative path defined in the request URI. - Edge CNAME URL: This type of request is rewritten to the corresponding CDN URL.	/800001/corigin/rewrittendir /foo.js
Request URI	%{request}	Describes the request. Syntax: <HTTP method> <relative path> <HTTP protocol>	GET /marketing/foo.js? loggedin=true HTTP/1.1

NAME	VARIABLE	DESCRIPTION	SAMPLE VALUE
Response Header Value	<code>%{resp_&lt;ResponseHeader&gt;}</code>	<p>Returns the value corresponding to the response header identified by the <code>&lt;ResponseHeader&gt;</code> term.</p> <p>If the name of the response header contains a dash (for example, <code>User-Agent</code>), replace it with an underscore (for example, <code>User_Agent</code>).</p>	<p>Sample Usage: <code>%{resp_Content_Length}</code></p> <p>Sample Value: 100</p>

## Usage

The following table describes the proper syntax for specifying an HTTP variable.

SYNTAX	EXAMPLE	DESCRIPTION
<code>%{&lt;HTTPVariable&gt;}</code>	<code>%{host}</code>	Use this syntax to get the entire value corresponding to the specified <code>&lt;HTTPVariable&gt;</code> .
<code>%{&lt;HTTPVariableDelimiter&gt;}</code>	<code>%{host,}</code>	Use this syntax to set the case for the entire value corresponding to the specified <code>&lt;HTTPVariableDelimiter&gt;</code> .
<code>%{&lt;HTTPVariableDelimiterExpression&gt;}</code>	<code>%{host/=^www.([^.]+).([^.]+)/cdn.\$2.\$3:80}</code>	Use a regular expression for <code>&lt;HTTPVariableDelimiterExpression&gt;</code> to replace, delete, or manipulate an HTTP variable's value.

HTTP variable names only support alphabetic characters and underscores. Convert unsupported characters to underscores.

## Delimiter reference

A delimiter can be specified after an HTTP variable to achieve any of the following effects:

- Transform the value associated with the variable.

Example: Convert the entire value to lowercase.

- Delete the value associated with the variable.

- Manipulate the value associated with the variable.

Example: Use regular expressions to change the value associated with the HTTP variable.

The delimiters are described in the following table.

DELIMITER	DESCRIPTION
<code>:=</code>	Indicates that a default value will be assigned to the variable when it is either: <ul style="list-style-type: none"> <li>- Missing</li> <li>- Set to NULL.</li> </ul>

DELIMITER	DESCRIPTION
:+	Indicates that a default value will be assigned to the variable when a value has been assigned to it.
:	Indicates that a substring of the value assigned to the variable will be expanded.
#	Indicates that the pattern specified after this delimiter should be deleted when it is found at the beginning of the value associated with the variable.
%	Indicates that the pattern specified after this delimiter should be deleted when it is found at the end of the value associated with the variable. This definition is only applicable when the % symbol is used as a delimiter.
/	Delimits an HTTP variable or a pattern.
//	Find and replace all instances of the specified pattern.
/=	Find, copy, and rewrite all occurrences of the specified pattern.
,	Convert the value associated with the HTTP variable to lowercase.
^	Convert the value associated with the HTTP variable to uppercase.
"	Convert all instances of the specified character in the value associated with the HTTP variable to lowercase.
^^	Convert all instances of the specified character in the value associated with the HTTP variable to uppercase.

## Exceptions

The following table describes circumstances under which the specified text isn't treated as an HTTP variable.

CONDITION	DESCRIPTION	EXAMPLE
Escaping % symbol	The percentage symbol can be escaped through the use of a backslash. The sample value to the right will be treated as a literal value and not as an HTTP variable.	%{host}
Unknown variables	An empty string is always returned for unknown variables.	%{unknown_variable}

CONDITION	DESCRIPTION	EXAMPLE
Invalid characters or syntax	<p>Variables that contain invalid characters or syntax are treated as literal values.</p> <p>Example #1: The specified value contains an invalid character (for example, -).</p> <p>Example #2: The specified value contains a double set of curly braces.</p> <p>Example #3: The specified value is missing a closing curly brace.</p>	<p>Example #1: %{resp_user-agent}</p> <p>Example #2: %{host}}</p> <p>Example #3: %{host}</p>
Missing variable name	A NULL value is always returned when a variable is not specified.	%{}
Trailing characters	<p>Characters that trail a variable are treated as literal values.</p> <p>The sample value to the right contains a trailing curly brace that will be treated as a literal value.</p>	%{host}}

## Setting default header values

A default value can be assigned to a header when it meets any of the following conditions:

- Missing/unset
- Set to NULL.

The following table describes how to define a default value.

CONDITION	SYNTAX	EXAMPLE	DESCRIPTION
Set a header to a default value when it meets any of the following conditions: - Missing Header - Header value is set to NULL.	%{Variable:=Value}	%{http_referrer:=unspecified}	The Referrer header will only be set to <i>unspecified</i> when it is either missing or set to NULL. No action will take place if it has been set.
Set a header to a default value when it is missing.	%{Variable=Value}	%{http_referrer=unspecified}	The Referrer header will only be set to <i>unspecified</i> when it is missing. No action will take place if it has been set.
Set the header to a default value when it does not meet any of the following conditions: - Missing - Set to NULL.	%{Variable:+Value}	%{http_referrer:+unspecified}	The Referrer header will only be set to <i>unspecified</i> when a value has been assigned to it. No action will take place if it is either missing or set to NULL.

# Manipulating variables

Variables can be manipulated in the following ways:

- Expanding substrings
- Removing patterns

## Substring expansion

By default, a variable will expand to its full value. Use the following syntax to only expand a substring of the variable's value.

```
%<Variable>:<Offset>:<Length>}
```

Key information:

- The value assigned to the *Offset* term determines the starting character of the substring:
  - Positive: The starting character of the substring is calculated from the first character in the string.
  - Zero: The starting character of the substring is the first character in the string.
  - Negative: The starting character of the substring is calculated from the last character in the string.
- The length of the substring is determined by the *Length* term:
  - Omitted: Omitting the *Length* term allows the substring to include all characters between the starting character and the end of the string.
  - Positive: Determines the length of the substring from the starting character to the right.
  - Negative: Determines the length of the substring from the starting character to the left.

### Example:

The following example relies on the following sample request URL:

<https://cdn.mydomain.com/folder/marketing/myconsultant/proposal.html>

The following string demonstrates various methods for manipulating variables:

[https://www%{http\\_host:3}/mobile/%{request\\_uri:7:10}/%{request\\_uri:-5:-8}.htm](https://www%{http_host:3}/mobile/%{request_uri:7:10}/%{request_uri:-5:-8}.htm)

Based on the sample request URL, the above variable manipulation will produce the following value:

<https://www.mydomain.com/mobile/marketing/proposal.htm>

## Pattern removal

Text that matches a specific pattern can be removed from either the beginning or the end of a variable's value.

SYNTAX	ACTION
<code>%{Variable#Pattern}</code>	Remove text when the specified pattern is found at the beginning of a variable's value.
<code>%{Variable%Pattern}</code>	Remove text when the specified pattern is found at the end of a variable's value.

### Example:

In this sample scenario, the *request\_uri* variable is set to:

```
/800001/myorigin/marketing/product.html?language=en-US
```

The following table demonstrates how this syntax works.

SAMPLE SYNTAX	RESULTS	
%{request_uri#/800001}/customerorigin	/customerorigin/myorigin/marketing/product.html?language=en-US	Because the variable starts with the pattern, it was replaced.
%{request_uri%html}htm	/800001/myorigin/marketing/product.html?language=en-US	Because the variable doesn't end with the pattern, there was no change.

## Find and replace

The find and replace syntax is described in the following table.

SYNTAX	ACTION
%{Variable/Find/Replace}	Find and replace first occurrence of the specified pattern.
%{Variable//Find/Replace}	Find and replace all occurrences of the specified pattern.
%{Variable^}	Convert the entire value to uppercase.
%{Variable^Find}	Convert the first occurrence of the specified pattern to uppercase.
%{Variable,}	Convert the entire value to lowercase.
%{Variable,Find}	Convert the first occurrence of the specified pattern to lowercase.

## Find and rewrite

For a variation on find and replace, use the text that matches the specified pattern when rewriting it. The find and rewrite syntax is described in the following table.

SYNTAX	ACTION
%{Variable/=Find/Rewrite}	Find, copy, and rewrite all occurrences of the specified pattern.
%{Variable/^ Find/Rewrite}	Find, copy, and rewrite the specified pattern when it occurs at the start of the variable.
%{Variable/\$Find/Rewrite}	Find, copy, and rewrite the specified pattern when it occurs at the end of the variable.
%{Variable/Find}	Find and delete all occurrences of the specified pattern.

Key information:

- Expand text that matches the specified pattern by specifying a dollar sign followed by a whole integer (for example, \$1).
- Multiple patterns can be specified. The order in which the pattern is specified determines the integer that will be assigned to it. In the following example, the first pattern matches "www," the second pattern matches the second-level domain, and the third pattern matches the top-level domain:

```
%{host/=^www\.(^\.)+\.\.([^\.:]+)/cdn.$2.$3:80}
```

- The rewritten value can consist of any combination of text and these placeholders.

In the previous example, the hostname is rewritten to `cdn.$2.$3:80` (for example, cdn.mydomain.com:80).

- The case of a pattern placeholder (for example, \$1) can be modified through the following flags:

- U: Uppercase the expanded value.

Sample syntax:

```
%{host/=^www\.( [^.]+)\.( [^.]+)/cdn.$U2.$3:80}
```

- L: Lowercase the expanded value.

Sample syntax:

```
%{host/=^www\.( [^.]+)\.( [^.]+)/cdn.$L2.$3:80}
```

- An operator must be specified before the pattern. The specified operator determines the pattern-capturing behavior:
  - `=`: Indicates that all occurrences of the specified pattern must be captured and rewritten.
  - `^`: Indicates that only text that starts with the specified pattern will be captured.
  - `$`: Indicates that only text that ends with the specified pattern will be capture.
- If you omit the `/Rewrite` value, the text that matches the pattern is deleted.

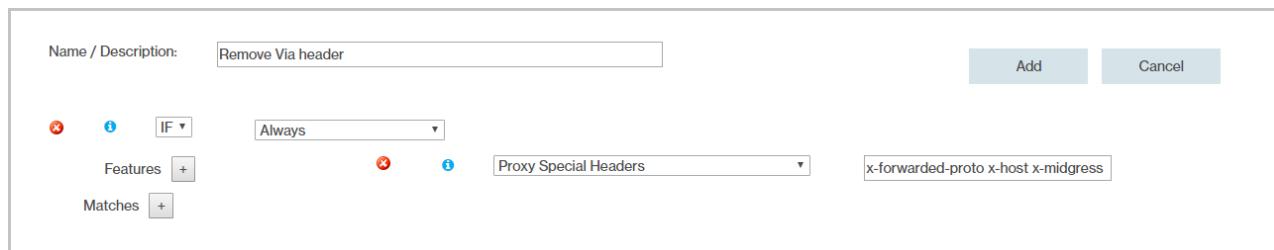
# Verizon-specific HTTP headers for Azure CDN rules engine

7/5/2019 • 2 minutes to read • [Edit Online](#)

For **Azure CDN Premium from Verizon** products, when an HTTP request is sent to the origin server, the point-of-presence (POP) server can add one or more reserved headers (or proxy special headers) in the client request to the POP. These headers are in addition to the standard forwarding headers received. For information about standard request headers, see [Request fields](#).

If you want to prevent one of these reserved headers from being added in the Azure CDN (Content Delivery Network) POP request to the origin server, you must create a rule with the [Proxy Special Headers feature](#) in the rules engine. In this rule, exclude the header you want to remove from the default list of headers in the headers field. If you've enabled the [Debug Cache Response Headers feature](#), be sure to add the necessary `X-EC-Debug` headers.

For example, to remove the `Via` header, the headers field of the rule should include the following list of headers: `X-Forwarded-For, X-Forwarded-Proto, X-Host, X-Midgress, X-Gateway-List, X-EC-Name, Host`.



The following table describes the headers that can be added by the Verizon CDN POP in the request:

REQUEST HEADER	DESCRIPTION	EXAMPLE
Via	Identifies the POP server that proxied the request to an origin server.	HTTP/1.1 ECS (dca/1A2B)
X-Forwarded-For	Indicates the requester's IP address.	10.10.10.10
X-Forwarded-Proto	Indicates the request's protocol.	http
X-Host	Indicates the request's hostname.	cdn.mydomain.com
X-Midgress	Indicates whether the request was proxied through an additional CDN server. For example, a POP server-to-origin shield server or a POP server-to-ADN gateway server. This header is added to the request only when midgress traffic takes place. In this case, the header is set to 1 to indicate that the request was proxied through an additional CDN server.	1
Host	Identifies the host and the port where the requested content may be found.	marketing.mydomain.com:80

REQUEST HEADER	DESCRIPTION	EXAMPLE
X-Gateway-List	ADN: Identifies the failover list of ADN Gateway servers assigned to a customer origin. Origin shield: Indicates the set of origin shield servers assigned to a customer origin.	icn1,hhp1,hnd1
X-EC-<name>	Request headers that begin with X-EC (for example, X-EC-Tag, <a href="#">X-EC-Debug</a> ) are reserved for use by the CDN.	waf-production

## Via request header

The format through which the `Via` request header identifies a POP server is specified by the following syntax:

```
Via: Protocol from Platform (POP/ID)
```

The terms used in the syntax are defined as follows:

- Protocol: Indicates the version of the protocol (for example, HTTP/1.1) used to proxy the request.
- Platform: Indicates the platform on which the content was requested. The following codes are valid for this field:

CODE	PLATFORM
ECAcc	HTTP Large
ECS	HTTP Small
ECD	Application delivery network (ADN)

- POP: Indicates the POP that handled the request.
- ID: For internal use only.

### Example Via request header

```
Via: HTTP/1.1 ECD (dca/1A2B)
```

## Host request header

The POP servers will overwrite the `Host` header when both of the following conditions are true:

- The source for the requested content is a customer origin server.
- The corresponding customer origin's HTTP Host Header option is not blank.

The `Host` request header will be overwritten to reflect the value defined in the HTTP Host Header option. If the customer origin's HTTP Host Header option is set to blank, then the `Host` request header that is submitted by the requester will be forwarded to the customer's origin server.

## X-Gateway-List request header

A POP server will add/overwrite the 'X-Gateway-List request header when either of the following conditions are met:

- The request points to the ADN platform.
- The request is forwarded to a customer origin server that is protected by the Origin Shield feature.

# X-EC-Debug HTTP headers for Azure CDN rules engine

7/5/2019 • 5 minutes to read • [Edit Online](#)

The debug cache request header, `X-EC-Debug`, provides additional information about the cache policy that is applied to the requested asset. These headers are specific to **Azure CDN Premium from Verizon** products.

## Usage

The response sent from the POP servers to a user includes the `X-EC-Debug` header only when the following conditions are met:

- The [Debug Cache Response Headers feature](#) has been enabled on the rules engine for the specified request.
- The specified request defines the set of debug cache response headers that will be included in the response.

## Requesting debug cache information

Use the following directives in the specified request to define the debug cache information that will be included in the response:

REQUEST HEADER	DESCRIPTION
X-EC-Debug: x-ec-cache	<a href="#">Cache status code</a>
X-EC-Debug: x-ec-cache-remote	<a href="#">Cache status code</a>
X-EC-Debug: x-ec-check-cacheable	<a href="#">Cacheable</a>
X-EC-Debug: x-ec-cache-key	<a href="#">Cache-key</a>
X-EC-Debug: x-ec-cache-state	<a href="#">Cache state</a>

## Syntax

Debug cache response headers may be requested by including the following header and the specified directives in the request:

`X-EC-Debug: Directive1,Directive2,DirectiveN`

### Sample X-EC-Debug header

`X-EC-Debug: x-ec-cache,x-ec-check-cacheable,x-ec-cache-key,x-ec-cache-state`

## Cache status code information

The X-EC-Debug response header can identify a server and how it handled the response through the following directives:

HEADER	DESCRIPTION
--------	-------------

HEADER	DESCRIPTION
X-EC-Debug: x-ec-cache	This header is reported whenever content is routed through the CDN. It identifies the POP server that fulfilled the request.
X-EC-Debug: x-ec-cache-remote	This header is reported only when the requested content was cached on an origin shield server or an ADN gateway server.

### Response header format

The X-EC-Debug header reports cache status code information in the following format:

- `X-EC-Debug: x-ec-cache: <StatusCode from Platform (POP/ID)>`
- `X-EC-Debug: x-ec-cache-remote: <StatusCode from Platform (POP/ID)>`

The terms used in the above response header syntax are defined as follows:

- StatusCode: Indicates how the requested content was handled by the CDN, which is represented through a cache status code.

The TCP\_DENIED status code may be reported instead of NONE when an unauthorized request is denied due to Token-Based Authentication. However, the NONE status code will continue to be used when viewing Cache Status reports or raw log data.

- Platform: Indicates the platform on which the content was requested. The following codes are valid for this field:

CODE	PLATFORM
ECAcc	HTTP Large
ECS	HTTP Small
ECD	Application Delivery Network (ADN)

- POP: Indicates the [POP](#) that handled the request.

### Sample response headers

The following sample headers provide cache status code information for a request:

- `X-EC-Debug: x-ec-cache: TCP_HIT from ECD (lga/0FE8)`
- `X-EC-Debug: x-ec-cache-remote: TCP_HIT from ECD (dca/EF00)`

## Cacheable response header

The `X-EC-Debug: x-ec-check-cacheable` response header indicates whether the requested content could have been cached.

This response header does not indicate whether caching took place. Rather, it indicates whether the request was eligible for caching.

### Response header format

The `x-EC-Debug` response header reporting whether a request could have been cached is in the following format:

```
X-EC-Debug: x-ec-check-cacheable: <cacheable status>
```

The term used in the above response header syntax is defined as follows:

VALUE	DESCRIPTION
YES	Indicates that the requested content was eligible for caching.
NO	Indicates that the requested content was ineligible for caching. This status may be due to one of the following reasons: <ul style="list-style-type: none"><li>- Customer-Specific Configuration: A configuration specific to your account can prevent the pop servers from caching an asset. For example, Rules Engine can prevent an asset from being cached by enabling the Bypass Cache feature for qualifying requests.</li><li>- Cache Response Headers: The requested asset's Cache-Control and Expires headers can prevent the POP servers from caching it.</li></ul>
UNKNOWN	Indicates that the servers were unable to assess whether the requested asset was cacheable. This status typically occurs when the request is denied due to token-based authentication.

### Sample response header

The following sample response header indicates whether the requested content could have been cached:

```
X-EC-Debug: x-ec-check-cacheable: YES
```

## Cache-Key response header

The `x-EC-Debug: x-ec-cache-key` response header indicates the physical cache-key associated with the requested content. A physical cache-key consists of a path that identifies an asset for the purposes of caching. In other words, the servers will check for a cached version of an asset according to its path as defined by its cache-key.

This physical cache-key starts with a double forward slash (//) followed by the protocol used to request the content (HTTP or HTTPS). This protocol is followed by the relative path to the requested asset, which starts with the content access point (for example, /000001/).

By default, HTTP platforms are configured to use *standard-cache*, which means that query strings are ignored by the caching mechanism. This type of configuration prevents the cache-key from including query string data.

If a query string is recorded in the cache-key, it's converted to its hash equivalent and then inserted between the name of the requested asset and its file extension (for example, asset<hash value>.html).

### Response header format

The `x-EC-Debug` response header reports physical cache-key information in the following format:

```
X-EC-Debug: x-ec-cache-key: CacheKey
```

### Sample response header

The following sample response header indicates the physical cache-key for the requested content:

```
X-EC-Debug: x-ec-cache-key: //http/800001/origin/images/foo.jpg
```

## Cache state response header

The `x-EC-Debug: x-ec-cache-state` response header indicates the cache state of the requested content at the time it was requested.

## Response header format

The `x-EC-Debug` response header reports cache state information in the following format:

```
X-EC-Debug: x-ec-cache-state: max-age=MASeconds (MATimePeriod); cache-ts=UnixTime (ddd, dd MMM yyyy HH:mm:ss GMT); cache-age=CASSeconds (CATimePeriod); remaining-ttl=RTSeconds (RTTimePeriod); expires-delta=ExpiresSeconds
```

The terms used in the above response header syntax are defined as follows:

- **MASeconds:** Indicates the max-age (in seconds) as defined by the requested content's Cache-Control headers.
- **MATimePeriod:** Converts the max-age value (that is, MASeconds) to the approximate equivalent of a larger unit (for example, days).
- **UnixTime:** Indicates the cache timestamp of the requested content in Unix time (also known as POSIX time or Unix epoch). The cache timestamp indicates the starting date/time from which an asset's TTL will be calculated.

If the origin server does not utilize a third-party HTTP caching server or if that server does not return the Age response header, then the cache timestamp will always be the date/time when the asset was retrieved or revalidated. Otherwise, the POP servers will use the Age field to calculate the asset's TTL as follows: Retrieval/RevalidateDateTime - Age.

- `ddd, dd MMM yyyy HH:mm:ss GMT`: Indicates the cache timestamp of the requested content. For more information, please see the UnixTime term above.
- **CASSeconds:** Indicates the number of seconds that have elapsed since the cache timestamp.
- **RTSeconds:** Indicates the number of seconds remaining for which the cached content will be considered fresh. This value is calculated as follows: RTSeconds = max-age - cache age.
- **RTTimePeriod:** Converts the remaining TTL value (that is, RTSeconds) to the approximate equivalent of a larger unit (for example, days).
- **ExpiresSeconds:** Indicates the number of seconds remaining before the date/time specified in the `Expires` response header. If the `Expires` response header was not included in the response, then the value of this term is *none*.

## Sample response header

The following sample response header indicates the cache state of the requested content at the time that it was requested:

```
X-EC-Debug: x-ec-cache-state: max-age=604800 (7d); cache-ts=1341802519 (Mon, 09 Jul 2012 02:55:19 GMT); cache-age=0 (0s); remaining-ttl=604800 (7d); expires-delta=none
```

# Debug HTTP header for Azure CDN from Microsoft

8/5/2019 • 2 minutes to read • [Edit Online](#)

The debug response header, `X-Cache`, provides details as to what layer of the CDN stack the content was served from. This header is specific to Azure CDN from Microsoft.

## Response header format

HEADER	DESCRIPTION
X-Cache: TCP_HIT	This header is returned when the content is served from the CDN edge cache.
X-Cache: TCP_REMOTE_HIT	This header is returned when the content is served from the CDN regional cache (Origin shield layer)
X-Cache: TCP_MISS	This header is returned when there is a cache miss, and the content is served from the Origin.

# Azure CDN Coverage by Metro

2/20/2020 • 2 minutes to read • [Edit Online](#)

This article lists current Metros containing point-of-presence (POP) locations, sorted by region, for Azure Content Delivery Network (CDN) products. Each Metro may contain more than one POP. For example, Azure CDN from Microsoft has 130 POPs across 80 Metros.

## IMPORTANT

POP city locations for **Azure CDN from Akamai** are not individually disclosed.

Because each Azure CDN product has a distinct way of building its CDN infrastructures, Microsoft recommends against using POP locations to decide which Azure CDN product to use. Instead, consider its features and end-user performance. Test the performance with each Azure CDN product to choose the right product for your users.

REGION	MICROSOFT	VERIZON	AKAMAI
North America	Toronto, Canada Vancouver, Canada Montreal, Canada Querétaro, Mexico San Juan, Puerto Rico Ashburn, VA, USA Atlanta, GA, USA Boston, MA, USA Cheyenne, WY, USA Chicago, IL, USA Dallas, TX, USA Denver, CO, USA Honolulu, HI, USA Houston, TX, USA Las Vegas, NV, USA Los Angeles, CA, USA Miami, FL, USA New York, NY, USA Newark, NJ, USA Phoenix, AZ, USA Portland, OR, USA San Antonio, TX, USA San Jose, CA, USA Seattle, WA, USA	Guadalajara, Mexico Mexico City, Mexico Puebla, Mexico Querétaro, Mexico Atlanta, GA, USA Boston, MA, USA Chicago, IL, USA Dallas, TX, USA Denver, CO, USA Detroit, MI, USA Los Angeles, CA, USA Miami, FL, USA New York, NY, USA Philadelphia, PA, USA San Jose, CA, USA Seattle, WA, USA Washington, DC, USA	Canada Mexico USA
South America	Campinas, Brazil Rio de Janeiro, Brazil Sao Paulo, Brazil Santiago, Chile	Buenos Aires, Argentina Rio de Janeiro, Brazil São Paulo, Brazil Valparaíso, Chile Barranquilla, Colombia Medellín, Colombia Quito, Ecuador Lima, Peru	Argentina Brazil Chile Columbia Ecuador Peru Uruguay

Region	Microsoft	Verizon	Akamai
Europe	Vienna, Austria Brussels, Belgium Sofia, Bulgaria Zagreb, Croatia Prague, Czech Republic Copenhagen, Denmark Helsinki, Finland Oslo, Norway Marseille, France Paris, France Berlin, Germany Frankfurt, Germany Athens, Greece Budapest, Hungary Dublin, Ireland Milan, Italy Amsterdam, Netherlands Warsaw, Poland Lisbon, Portugal Bucharest, Romania Barcelona, Spain Madrid, Spain Stockholm, Sweden Zurich, Switzerland London, UK Manchester, UK	Vienna, Austria Copenhagen, Denmark Helsinki, Finland Marseille, France Paris, France Frankfurt, Germany Milan, Italy Riga, Latvia Amsterdam, Netherlands Warsaw, Poland Madrid, Spain Stockholm, Sweden London, UK	Austria Bulgaria Denmark Finland France Germany Greece Ireland Italy Netherlands Poland Russia Spain Sweden Switzerland United Kingdom
Africa	Cape Town, South Africa Johannesburg, South Africa Lagos, Nigeria Nairobi, Kenya	Johannesburg, South Africa	South Africa
Middle East	Dubai, United Arab Emirates	Muscat, Oman Fujirah, United Arab Emirates	Qatar United Arab Emirates
India	Chennai, India Hyderabad, India Mumbai, India New Delhi, India	Bengaluru (Bangalore), India Chennai, India Mumbai, India New Delhi, India	India
Asia	Hong Kong Osaka, Japan Tokyo, Japan Kuala Lumpur, Malaysia Jakarta, Indonesia Manila, Philippines Singapore Busan, South Korea Seoul, South Korea Taipei, Taiwan Bangkok, Thailand Saigon, Vietnam	Hong Kong Batam, Indonesia Jakarta, Indonesia Osaka, Japan Tokyo, Japan Singapore Seoul, South Korea Kaohsiung, Taiwan Taipei, Taiwan	Hong Kong Indonesia Israel Japan Macau Malaysia Philippines Singapore South Korea Taiwan Thailand Turkey Vietnam

REGION	MICROSOFT	VERIZON	AKAMAI
Australia and New Zealand	Brisbane, Australia Melbourne, Australia Perth, Australia Sydney, Australia Auckland, New Zealand	Melbourne, Australia Sydney, Australia Auckland, New Zealand	Australia New Zealand

## Next steps

- To get the latest IP addresses for whitelisting, see the [Azure CDN Edge Nodes API](#).

# Azure CDN POP locations by abbreviation

7/5/2019 • 4 minutes to read • [Edit Online](#)

This article lists POP locations, sorted by POP abbreviation, for **Azure CDN from Verizon**.

ABBREVIATION	LOCATION	REGION
AGA	Atlanta, Georgia, USA	North America: East Coast U.S.
AGB	Atlanta, Georgia, USA	North America: East Coast U.S.
AKL	Auckland, New Zealand	Asia
AMA	Amsterdam, Netherlands	Europe
AMB	Amsterdam, Netherlands	Europe
AMS	Amsterdam, Netherlands	Europe
ARN	Stockholm, Sweden	Europe
ATL	Atlanta, Georgia, USA	North America: East Coast U.S.
BAQ	Barranquilla, Colombia	Latin America
BLR	Bengaluru (Bangalore), India	Asia
BNJ	Newark, New Jersey, USA	North America: East Coast U.S.
BOS	Boston, Massachusetts, USA	North America: East Coast U.S.
BTH	Batam, Indonesia	Asia
BUE	Buenos Aires, Argentina	Latin America
BUR	Los Angeles, California, USA	North America: West Coast U.S.
CDG	Paris, France	Europe
CGH	São Paulo, Brazil	Latin America
CGK	Jakarta, Indonesia	Asia
CHA	Chicago, Illinois, USA	North America: East Coast U.S.
CHB	Chicago, Illinois, USA	North America: East Coast U.S.
CNJ	Newark, New Jersey, USA	North America: East Coast U.S.

ABBREVIATION	LOCATION	REGION
CPH	Copenhagen, Denmark	Europe
CPM	Los Angeles, California, USA	North America: West Coast U.S.
CVA	Ashburn, Virginia, USA	North America: East Coast U.S.
DAA	Dallas, Texas, USA	North America: Central Coast U.S.
DAB	Dallas, Texas, USA	North America: Central Coast U.S.
DCA	Ashburn, Virginia, USA	North America: East Coast U.S.
DCB	Ashburn, Virginia, USA	North America: East Coast U.S.
DCC	Ashburn, Virginia, USA	North America: East Coast U.S.
DEL	Noida, India	Asia
DEN	Denver, Colorado, USA	North America: Central U.S.
DFW	Dallas, Texas, USA	North America: Central U.S.
EWR	New York, New York USA	North America: East Coast U.S.
EZE	Buenos Aires, Argentina	Latin America
FCN	Frankfurt, Germany	Europe
FLL	Miami, Florida, USA	North America: East Coast U.S.
FRA	Frankfurt, Germany	Europe
FRB	Frankfurt, Germany	Europe
FRC	Frankfurt, Germany	Europe
FRF	Frankfurt, Germany	Europe
FTW	Dallas, Texas, USA	North America: Central U.S.
FTY	Atlanta, Georgia, USA	North America: East Coast U.S.
GIG	Rio de Janeiro, Brazil	Latin America
GRU	São Paulo, Brazil	Latin America
HEL	Helsinki, Finland	Europe
HHP	Hong Kong SAR	Asia

ABBREVIATION	LOCATION	REGION
HKC	Hong Kong SAR	Asia
HKG	Hong Kong SAR	Asia
HND	Tokyo, Japan	Asia
IAD	Ashburn, Virginia, USA	North America: East Coast U.S.
ICN	Seoul, Korea	Asia
ITM	Osaka, Japan	Asia
JFK	New York, New York, USA	North America: East Coast U.S.
JNB	Johannesburg, South Africa	Europe
KHH	Kaohsiung, Taiwan	Asia
KIX	Osaka, Japan	Asia
LAA	Los Angeles, California, USA	North America: West Coast U.S.
LAB	Los Angeles, California, USA	North America: West Coast U.S.
LAM	Los Angeles, California, USA	North America: West Coast U.S.
LAN	Los Angeles, California, USA	North America: West Coast U.S.
LAX	Los Angeles, California, USA	North America: West Coast U.S.
LCY	London, UK	Europe
LGA	New York, New York, USA	North America: East Coast U.S.
LHA	London, UK	Europe
LHB	London, UK	Europe
LHM	London, UK	Europe
LHN	London, UK	Europe
LHR	London, UK	Europe
LIM	Lima, Peru	Latin America
MAA	Chennai, India	Asia
MAD	Madrid, Spain	Europe

ABBREVIATION	LOCATION	REGION
MDE	Medellin, Colombia	Latin America
MDW	Chicago, Illinois, USA	North America: East Coast U.S.
MEB	Melbourne, Australia	Asia
MEL	Melbourne, Australia	Asia
MEX	Mexico City, Mexico	North America
MIA	Miami, Florida, USA	North America: East Coast U.S.
MIB	Miami, Florida, USA	North America: East Coast U.S.
MIC	Miami, Florida, USA	North America: East Coast U.S.
MRS	Marseille, France	Europe
MXP	Milan, Italy	Europe
NAG	Mumbai, India	Asia
NDL	Delhi, India	Asia
NRT	Tokyo, Japan	Asia
NYA	New York, New York, USA	North America: East Coast U.S.
NYB	New York, New York, USA	North America: East Coast U.S.
OMM	Muscat, Oman	Middle East
ORD	Chicago, Illinois, USA	North America: Central U.S.
ORY	Paris, France	Europe
OXR	Los Angeles, California, USA	North America: West Coast U.S.
PAB	Paris, France	Europe
PAE	Seattle, Washington, USA	North America: West Coast U.S.
PBC	Puebla, Mexico	North America
PHL	Philadelphia, Pennsylvania, USA	North America: East Coast U.S.
PNQ	Mumbai, India	Asia
POX	Paris, France	Europe

ABBREVIATION	LOCATION	REGION
QRO	Santiago de Querétaro, Mexico	North America
RHV	San Jose, California, USA	North America: West Coast U.S.
RIB	Rio de Janeiro, Brazil	Latin America
RIX	Riga, Latvia	Europe
RTM	Amsterdam, Netherlands	Europe
SAA	San Jose, California, USA	North America: West Coast U.S.
SAB	San Jose, California, USA	North America: West Coast U.S.
SCL	Valparaíso, Chile	Latin America
SEA	Seattle, Washington, USA	North America: West Coast U.S.
SEB	Seattle, Washington, USA	North America: West Coast U.S.
SEC	Seattle, Washington, USA	North America: West Coast U.S.
SGB	Singapore	Asia
SIN	Singapore	Asia
SJC	San Jose, California, USA	North America: West Coast U.S.
SJO	San Jose, California, USA	North America: West Coast U.S.
STO	Stockholm, Sweden	Europe
SYD	Sydney, Australia	Asia
TIR	Chennai, India	Asia
TKA	Tokyo, Japan	Asia
TKB	Tokyo, Japan	Asia
TOJ	Madrid, Spain	Europe
UAE	Fujairah, United Arab Emirates	Europe
UIO	Quito, Ecuador	Latin America
VIE	Vienna, Austria	Europe
VNY	Los Angeles, California, USA	North America: West Coast U.S.

ABBREVIATION	LOCATION	REGION
WAW	Warsaw, Poland	Europe

# Understanding Azure CDN billing

11/4/2019 • 4 minutes to read • [Edit Online](#)

This FAQ describes the billing structure for content hosted by Azure Content Delivery Network (CDN).

## What is a billing region?

A billing region is a geographic area used to determine what rate is charged for delivery of objects from Azure CDN. The current billing zones and their regions are as follows:

- Zone 1: North America, Europe, Middle East, and Africa
- Zone 2: Asia Pacific (including Japan)
- Zone 3: South America
- Zone 4: Australia and New Zealand
- Zone 5: India

For information about point-of-presence (POP) regions, see [Azure CDN POP locations by region](#). For example, a POP located in Mexico is in the North America region and is therefore included in zone 1.

For information about Azure CDN pricing, see [Content Delivery Network pricing](#).

## How are delivery charges calculated by region?

The Azure CDN billing region is based on the location of the source server delivering the content to the end user. The destination (physical location) of the client is not considered the billing region.

For example, if a user located in Mexico issues a request and this request is serviced by a server located in a United States POP due to peering or traffic conditions, the billing region will be the United States.

## What is a billable Azure CDN transaction?

Any HTTP(S) request that terminates at the CDN is a billable event, which includes all response types: success, failure, or other. However, different responses may generate different traffic amounts. For example, *304 Not Modified* and other header-only responses generate little traffic because they are a small header response; similarly, error responses (for example, *404 Not Found*) are billable but incur a small cost because of the tiny response payload.

## What other Azure costs are associated with Azure CDN use?

Using Azure CDN also incurs some usage charges on the services used as the origin for your objects. These costs are typically a small fraction of the overall CDN usage cost.

If you are using Azure Blob storage as the origin for your content, you also incur the following storage charges for cache fills:

- Actual GB used: The actual storage of your source objects.
- Transactions: As needed to fill the cache.
- Transfers in GB: The amount of data transferred to fill the CDN caches.

#### **NOTE**

Starting October 2019, If you are using Azure CDN from Microsoft, the cost of data transfer from Origins hosted in Azure to CDN PoPs is free of charge. Azure CDN from Verizon and Azure CDN from Akamai are subject to the rates described below.

For more information about Azure Storage billing, see [Understanding Azure Storage Billing – Bandwidth, Transactions, and Capacity](#).

If you are using *hosted service delivery*, you will incur charges as follows:

- Azure compute time: The compute instances that act as the origin.
- Azure compute transfer: The data transfers from the compute instances to fill the Azure CDN caches.

If your client uses byte-range requests (regardless of origin service), the following considerations apply:

- A *byte-range request* is a billable transaction at the CDN. When a client issues a byte-range request, this request is for a subset (range) of the object. The CDN responds with only a partial portion of the content that is requested. This partial response is a billable transaction and the transfer amount is limited to the size of the range response (plus headers).
- When a request arrives for only part of an object (by specifying a byte-range header), the CDN may fetch the entire object into its cache. As a result, even though the billable transaction from the CDN is for a partial response, the billable transaction from the origin may involve the full size of the object.

## How much transfer activity occurs to support the cache?

Each time a CDN POP needs to fill its cache, it makes a request to the origin for the object being cached. As a result, the origin incurs a billable transaction on every cache miss. The number of cache misses depends on a number of factors:

- How cacheable the content is: If the content has high TTL (time-to-live)/expiration values and is accessed frequently so it stays popular in cache, then the vast majority of the load is handled by the CDN. A typical good cache-hit ratio is well over 90%, meaning that less than 10% of client requests have to return to origin, either for a cache miss or object refresh.
- How many nodes need to load the object: Each time a node loads an object from the origin, it incurs a billable transaction. As a result, more global content (accessed from more nodes) results in more billable transactions.
- TTL influence: A higher TTL for an object means it needs to be fetched from the origin less frequently. It also means clients, such as browsers, can cache the object longer, which can reduce the transactions to the CDN.

## Which origin services are eligible for free data transfer with Azure CDN from Microsoft?

If you use one of the following Azure services as your CDN origin, you will not be charged for Data transfer from the Origin to the CDN PoPs.

- Azure Storage
- Azure Media Services
- Azure Virtual Machines
- Virtual Network
- Load Balancer
- Application Gateway

- Azure DNS
- ExpressRoute
- VPN Gateway
- Traffic Manager
- Network Watcher
- Azure Firewall
- Azure Front Door Service
- Azure Bastion
- Azure App service
- Azure Functions
- Azure Data Factory
- Azure API Management
- Azure Batch
- Azure Data Explorer
- HDInsight
- Azure Cosmos DB
- Azure Data Lake Store
- Azure Machine Learning
- Azure SQL database
- Azure Cache for Redis

## How do I manage my costs most effectively?

Set the longest TTL possible on your content.