Cloud Embed and Custom Domain HTTPS Orchestration

November 24, 2021

Contents

Welcome to CDHO for ACE

Custom Domain HTTPS Orchestration (CDHO) allows you to configure HTTPS delivery on your customers' vanity domains for Akamai Cloud Embed (ACE). Vanity HTTPS delivery is supported via a shared domain validated, subject alternate name (DV SAN) certificate workflow.

Akamai manages the workflow. This includes the entire certificate life cycle and Domain Name System (DNS) changes to support this functionality. We maintain a set of certificates that include subcustomer domains in the certificates' subject alternative name (SAN).

Get to know these common terms

Before you get started, take a moment and review these terms.

Term	Description
ACE	Short for "Akamai Cloud Embed" (previously referred to as "Wholesale Delivery").
CDHO	Short for "Custom Domain HTTPS Orchestration."
Subcustomer	As our cloud partner, this is *your* customer, that is using your ACE CDN. You'll add CDHO support for each subcustomer that needs to access the ACE CDN securely via HTTPS, using their custom hostnames.
Custom Domain/ Vanity Domain	This is the domain for a subcustomer's asset (website or app), that does not use your partner domain suffix. For example, assume a subcustomer's website is `www.subcustomer.com`. This is the custom, or "vanity" domain.
Delegated DNS Zone	This is the DNS zone that's delegated to Akamai. If the zone that the subcustomer CNAMEs to are records in the zone `partnerdomain.com`, you should have another zone created specifically for Akamai—such as `akamai.partnerdomain.com`—that will be delegated to Akamai. (For example, `www.customer1.com` is CNAMEd to `customer1.partnerdomain.com`, `customer1.partnerdomain.com` is CNAMEd to `customer1.akamai.partnerdomain.com`, and finally, `customer1.akamai.partnerdomain.com` is CNAMEd by CDHO to `partnerakamaiedgehostname.edgekey.net`.)
Partner Domain [1]	This is the last hostname in a CNAME chain, *before* mapping to an Edge server using the Akamai Edge Hostname that ends with `edgekey.net`, `akamaized.net`, or `edgesuite.net`. For example, if a subcustomer has the identifier "`xyz`" in the CNAME chain, and the DNS Zone that you have delegated to Akamai is `akamai.cloudpartnerdomain.com`, the customer-specific Partner Domain would be `xyz.akamai.cloudpartnerdomain.com`.
CPS	This is short for the Akamai "Certificate Provisioning System," that is used to provision certificates on our platform.
Edge DNS	This is our proprietary domain name system service. You use it to manage the DNS changes in this orchestration.
DV	This refers to a Domain Validated certificate. These are the secure certificates used in the TLS handshake for HTTPS requests.

[1] This is referred to as the "partnerDomain" in the ACE API.

Term	Description
SAN	This refers to the Subject Alternative Name—all of the custom domains added to a certificate—other than the Common Name (CN). A SAN is an extension to X.509 that allows various values to be associated with a security certificate using a `subjectAltName` field.
Let'sEncrypt	This is the Certificate Authority (CA) we use for DV SAN certificates.

Before you begin

Before you integrate the Custom Domain HTTPS Orchestration (CDHO) feature, you need to perform various tasks, inside and outside of ACE.

You need DV certificates

You need to work with your account representative to get domain validated subject alternative name (DV SAN) certificates provisioned for use with Custom Domain HTTPS Orchestration (CDHO). These certificates serve as the authority for HTTPS requests from subcustomer custom domains.

CDHO adds subcustomer custom domains as a SAN entry in a DV certificate. To do so, ACE requires dedicated access to these certificates. So, we need to generate and maintain these certificates for you. (These certificates will be accessible via the Certificate Provisioning System (CPS) in Control Center. This provides you the visibility into certificate allocations and propagations.

There are some things to consider regarding these certificates:

- **You need to contact your account representative for the following**:
  - To help you identify the total number of certificates you will need.
  - To create the certificates for you.
  - To enable the CDHO feature to use these certificates.
- **You *must not edit* these certificates**. This is because the certificates are being actively used by the CDHO feature.
- **A DV certificate can contain a maximum of 100 SANs including the Common Name (CN)**. So, we can support up to 99 custom domains in total in a certificate, because the Common Name (CN) is stored as a SAN entry. For example, if you have a total of 101 custom domains, you would need two DV certificates.

You need a base configuration

Just like a traditional ACE workflow, you need to set up a base configuration to implement support for Custom Domain HTTPS Orchestration (CDHO).

You can set up this configuration in multiple ways, but we recommend that you use Property Manager in Akamai Control Center, and that's what's discussed here. This is a multiphase process—specifically, you need to set up a secure (HTTPS) Property Hostname and apply settings in the Default Rule. The process is fully discussed in the *Cloud Embed user guide*.

Get your base configuration "property_id"

After you create your base configuration, you need to obtain its unique `property_id` value. This value is used when configuring subcustomers via the ACE API. Contact your account representative to obtain this value.

Tip: If you have access to it, and are versed in its use, you can *use the Property Manager API* to get this value.

You need Edge DNS

To configure and use this support, you need to have the Akamai Edge DNS service added to your contract.

The DNS management portion of this support requires that you delegate a domain—your partner domain—so that we can manage it on the Edge DNS platform. You may need to contact your account representative to get this service added to your contract.

Add a primary DNS configuration

Once Edge DNS has been added to your contract, use it to add a primary DNS configuration:

1. Open the application. Go to > **DNS SOLUTIONS** > **Authoritative DNS**.
2. Click **Add zone**.
3. Set the Zone type to **Primary**.
4. In the Zone names field, input the suffix of your partner domain. (For example, "akamai.partnerdomain.com.")
5. Leave all other options at their default and click **Create zone**.

All applicable custom hostnames must be CNAMEd to a record in the zone you just added.

Add records

You need to add a wildcard record to the record set in the DNS configuration:

1. Click the large **Add record sets** button that's revealed. Your new domain is listed in the Zone record sets panel.
2. Click **Add New Record Set**.
3. Generate a wildcard record by inputting an asterisk (" * ") for the Name (for example, *.<your partner domain suffix>).
4. Select **CNAME** from the **Type** drop down.
5. You can set a custom **TTL** if the default is not long enough for the "time to live" for the record.
6. In the Record data field, input the domain for the Edge Hostname used in the base configuration that all your customer domains will be CNAMEd to by default, followed by a trailing period (for example, “**partnerdomain.com.edgekey.net.**”).
7. Click **Add to change list**.
8. Make note of Akamai-managed DNS servers listed below the heading, Record data.

You need to update your DNS Server settings

As the owner of the partner domain, you need to change your DNS Server settings so that your partner domain points to the list of Akamai-managed DNS servers. These are the values you noted from the Record data column, above.

You can click the Zone name in the **Fast DNS: Zone list** UI to access and view these values.

You need to set up subcustomers

Before you can enable Custom Domain HTTPS Orchestration (CDHO) for secure transfer, a subcustomer needs to be properly registered for use, and you also need to set up a policy to configure desired settings for the subcustomer. This is all accomplished using the ACE API.

Registration and policy configuration for a subcustomer are covered in the *Cloud Embed user guide*:

- You begin by generating and registering a unique identifier ("ID") for a subcustomer. This is covered in the *Register subcustomers* topic in this document.
- You determine rules and behaviors that are to be applied for delivery of that subcustomers' content. This policy needs to be associated with the vanity domain that end users are using to request via HTTPS. This is covered in the *Create a delivery policy for each subcustomer* topic in this document.

During the CDHO enabling process, ACE verifies that the policy exists for the vanity domain. If one doesn't exist, Akamai returns a "409-Request Conflict" error code, indicating a request conflict. During CDHO disabling, Akamai doesn't check for the policy.

Subcustomer set up requires v2 of the API

All API operations used to register and configure a subcustomer require that you use version 2 API endpoints. This means that all applicable calls must include "v2" in the request path (for example, `/partner-api/`**`v2`**`/network/{network}/properties/{propertyId}/customers/{subcustomerId}`).

This is an earlier version from what is used to configure the CDHO support covered in this documentation (which uses v3).

Enable CDHO

You need to make individual PUT requests using the ACE API to enable Custom Domain HTTPS Orchestration (CDHO) for *each* of a subcustomer's custom domains.

Note: Before starting, ensure that you've met all of the prerequisites covered in *Before you begin* on page 4.

CDHO enabling requires v3 of the API

All API operations for CDHO support require that you use version 3 API endpoints, and that you include "v3" in the request path (for example, `/partner-api/v3/network/{network}/properties/...`).

This is a *different version* than what is used to perform subcustomer *registration and policy configuration*. For those required operations, you still need to use **version 2** of the API.

Each request is stored in a batch

Our system groups multiple CDHO enable requests for the same Partner Domain into a "batch" for all certificate changes. This batch is then processed to apply all requests at once. This is because certificate requests can take from three to eight hours to complete. We want subcustomers to be able to queue multiple requests and have them all propagate during a single three to eight hour window. (Instead of having individual requests each take this length of time.)

Tip: There is a default "wait time" for a batch to trigger requests to our Certificate Provisioning System (CPS). This wait time is configurable on a per-partner basis. Contact your account representative for details.

The table below offers an approximate breakdown of the time it takes for a normal request to enable HTTPS on a custom domain.

Phase	Time required
Request Batching	~ 15 minutes
Domain Validation	1 - 4 hours (This can vary, especially if each "challenge" in the process is satisfied.)
Certificate Propagation	2 - 4 hours
DNS Change (when required)	5 - 15 minutes

Note: These times are based on averages. Any phase in the process may take more time to complete.

Make a PUT /secure-delivery request

Use the following operation to enable HTTPS for an individual custom domain.

Method	Endpoint
PUT	/partner-api/**v3**/network/production/properties/{propertyId}/sub-properties/{domainName}/secure-delivery

Important: Currently, only the production network is supported for use with this operation.

The variable URI request parameters (enclosed in "{ }") are as follows:

- {propertyId}. This is the unique identifier for the applicable ACE base configuration.
- {domainName}. This is the custom domain that the subcustomer wants to enable for CDHO.

Request header

You must also include the following header with the PUT request.

```
X-Customer-ID: {subcustomerID}
```

- {subcustomerID}: This is the unique ID that's been set up for the target subcustomer. As the cloud partner, it's up to you to determine a method to create these IDs and associate them with each subcustomer. The ID can contain up to 50 alphanumeric, dot or dash characters. (When you register each subcustomer during the "Before you begin" phase, you provide this value.)

The Request body

The PUT call also requires a request body comprised of the following parameters:

```
{
    "domainInfo":
    {
        "enableHTTPS": true,
        "certificateType": "shared",
        "partnerDomain": "demo4.akamai.partnerdomain.net",
        "domainListComplete": false
    }
}
```

Parameter	Description
enableHTTPS	*Required:* This is a boolean that you need to set to true to enable CDHO.
certificateType	*Required:* Set this to "shared" for Custom Domain HTTPS using a DV SAN certificate. Shared SAN certificates only allow a maximum of 10 custom hostnames per partner domain.
partnerDomain	*Required:* Set the name of the partner domain that is mapped to the Edge hostname of the selected certificate in Edge DNS. This is used to associate all requests within a batch.
domainListComplete	*Optional:* This is a boolean that defaults to false. Include this (and set to true) to trigger a faster submission of the batch request. (The next time the partnerDomain batch is examined, it will begin processing—It won't wait for the standard timeout.)

Parameter	Description
	A use case scenario for this involves *multiple custom domains associated with the same partner domain*, that are being enabled at the same time. You can issue multiple PUT /secure-delivery requests, with the final one including this parameter and setting it to `true`, and all preceding ones setting it to `false`. As a result, the system will try to group all of these requests together. (However, there is no guarantee that this will actually happen.)

HTTP status codes

This describes the standard HTTP status code returned after a request to the API.

Code	Explanation
202	**Success**: The request was successfully accepted and will be processed asynchronously. (The final success or failure of this request will be determined at a later time.)
4xx	**Client Error**: This is a client or validation error regarding access to the API. The request itself may be improperly formatted. Correct the request and try again.
5xx	**Server Error**: An internal error occurred with our server when the API made the request. Try the request again.

Important: These codes only apply to the actual acceptance of the request. With a successfully accepted request, the resulting JSON response also includes the `statusCode` entry that offers a code that describes the actual status of the CDHO feature enablement process. (Details are included in the description of this entry in the table below.) If this standard HTTP status code results in anything other than a "202 Success" (4xx or 5xx), the JSON response with the `statusCode` entry is *not generated*.

The response

```
{
    "status": "CHANGE_REQUEST_PENDING",
    "network": "production",
    "partnerDomain": "demo4.akamai.partnerdomain.net",
    "statusCheckId": "<unique key>",
    "domainInfo": {
        "enableHTTPS": true,
        "certificateType": "shared",
        "domainListComplete": false,
        "partnerDomain": "demo4.akamai.partnerdomain.net"
    },
    "removeVanityHostList": [],
    "batch": {
        "batchRunTimestamp": 1507553484477,
        "batchTimeLeft": 896862,
        "batchId": "<unique ID>",
        "currentTimestamp": 1507552587615,
        "domainListComplete": false,
        "lastBatchModifiedTimestamp": 1507552586943
```

```
    },
    "addVanityHostList": [
        {
            "certificateType": "shared",
            "unmappedFromAkamaiTimestamp": 1507552586943,
            "vanityHostName": "data.demo4.biz"
        }
    ],
    "certificates": {},
    "propertyId": <unique ID>,
    "statusCode": 202
}
```

Parameter	Description
`status`	The status of the request after you initiate it. Applicable statuses are discussed in *Status transitions for a request* on page 25.
`network`	The network targeted for the request—`production`, in this case because it is all that is currently supported.
`partnerDomain`	Your cloud partner domain. The custom domain that is enabling CDHO needs to be CNAMEd to this domain to have HTTPS support. If a customer has live HTTPS traffic on the custom domain before using this feature, the CNAMEing can be done after HTTPS enablement is completed on that domain.
`statusCheckId`	The unique "status check ID" for the request. This ID can be used to get the status of an enablement request. (*You can check the status of requests* that have been added to the batch.) **Important:** Record this value. It's used for other operations in the API, and it's also required by technical support for use in troubleshooting. Currently, there is no other way to obtain this value via the ACE API, so it is important that you note it here.
`domainInfo`	This is a duplicate of the information you provided for the `domainInfo` object in the PUT request. (Used for confirmation purposes.) • `enableHTTPS`: If `true`, the request was to enable CDHO. If `false` the request was to disable CDHO. • `certificateType`: The type of certificate set in the request. Only “shared” is permitted at this time.

Parameter	Description
	• `domainListComplete`: Whether or not the batch was allowed to close based on timeout or a custom hostname that declared it was the last one for this batch. • `partnerDomain`: The partner domain specified in the request.
`removeVanityHostList`	For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response. If the modification batch associated with this request contains any PUT secure-delivery requests with `"enableHTTPS": false` that will result in removal from the certificate, then each object in this list represents the corresponding vanity domain that will be removed from the certificate. Otherwise this list will be empty. • `certificateType`: The type of certificate targeted by the request. • `unmappedFromAkamaiTimestamp`: This field is used for Akamai internal purposes. If you receive this error, you can ignore it. • `vanityHostName`: A custom domain to be added to the certificate included in the batch containing this request.
`batch`	For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response. This object includes various information pertaining to the batch that includes this request. • `batchRunTimestamp`: The time the batch was initiated • `batchTimeLeft`: The time left before the batch is processed. This is set back to zero, once a batch begins processing. • `batchId`: A unique ID value for the batch • `currentTimestamp`: The current time. • `domainListComplete`: Whether or not the batch was allowed to close based on

Parameter	Description
	timeout or a custom domain that declared it was the last one for this batch. • `lastBatchModifiedTimestamp`: When the batch was last modified. (For example, the last time a request was added to the batch.)
`addVanityHostList`	For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response. If the modification batch associated with this request contains any PUT secure-delivery requests with `"enableHTTPS": true` that will result in addition to the certificate, then each object in this list represents the corresponding vanity domain that will be added to the certificate. Otherwise this list will be empty. • `certificateType`: The type of certificate targeted by the request. • `unmappedFromAkamaiTimestamp`: This field is used for Akamai internal purposes. If you receive this error, you can ignore it. • `vanityHostName`: A custom domain to be added to the certificate included in the batch containing this request.
`certificates`	This includes details pertaining to the certificates being used. At this phase of the process, this will be empty.
`propertyId`	subcustomers are associated with a base configuration that sets the baseline for match criteria and rules to be applied for an incoming request. A single base configuration can have several subcustomers associated with it. This is the unique ID of the base configuration to which you've registered the subcustomer that owns the custom domain.
`statusCode`	A code displayed here represents the status of the request in regards to the CDHO feature and how it is processing the request. Values shown here represent the following: • **202**: Success. This indicates that the request has been accepted by ACE, and it has been added to the property domain

Parameter	Description
	name batch. *You can get the status of the request* to view details on its progress. • **4xx**: Client or validation errors have occurred with the CDHO feature. Check the formatting of the request, the PUT request body and the X-Customer-ID header, and retry. If the error persists, contact technical support. Specific 4xx codes are as follows: – **401**: Authentication failed – **403**: Authorization failed – **404**: Not found – **409**: Request conflict error • **5xx**: A server error has occurred while CDHO enablement was being processed. Try the request again. ⓘ **Note:** This code is different than what is supplied as the standard HTTP status code after an API request is issued. This only applies to the current status of enablement for CDHO.

Disable CDHO

If necessary, you can disable CDHO for a specific custom domain.

This is as simple as using the PUT /secure-delivery request, and setting `"enableHTTPS": false` in the request body:

```
{
    "domainInfo": {
        "enableHTTPS": false,
        "certificateType": "shared",
        "partnerDomain": "demo4.akamai.partnerdomain.net",
        "domainListComplete": false
    }
}
```

All other parameters and requirements are the same as covered in *Enable CDHO* on page 7.

Get the status of your CDHO requests

We offer a specific API request that you can use to get the status of your HTTPS enablement request for Custom Domain HTTPS Orchestration (CDHO).

CDHO enabling requires v3 of the API

All API operations for CDHO support require that you use version 3 API endpoints, and that you include "v3" in the request path (for example, `/partner-api/v3/network/{network}/properties/...`).

This is a *different version* than what is used to perform subcustomer *registration and policy configuration*. For those required operations, you still need to use **version 2** of the API.

Make a GET the batch status request

Method	URI
`GET`	`/partner-api/v3/network/production/properties/{property-id}/check/{statusCheckId}`

Important: Currently, only the production network is supported for use with this operation.

The variable URI request parameters (enclosed in "{ }") are as follows:

- `{propertyId}`. This is the unique identifier for the applicable ACE base configuration.
- `{statusCheckId}`: This is the unique "status check ID" for the target enablement (or disablement) request. This ID is returned in the response output for an enablement request.

HTTP status codes

This describes the standard HTTP status code returned after a request to the API.

Code	Explanation
200	**Success**
4xx	**Client Error**: This is a client or validation error regarding access to the API. The request itself may be improperly formatted. Correct the request and try again. Specific 4xx codes are as follows: • **401**: Authentication failed • **403**: Authorization failed • **404**: Not found • **409**: Request conflict error
5xx	**Server Error**: An internal error occurred with our server when the API made the request. Try the request again.

Important: These codes only apply to the actual acceptance of the request. With a succesfully accepted request, the resulting JSON response also includes the `statusCode` entry that offers a code that describes the actual status of the CDHO feature enablement process. (Details are included in the description of this entry in the table below.) If this standard HTTP status code results in anything other than a "202 Success" (4xx or 5xx), the JSON response with the `statusCode` entry is *not generated*.

Example response

```
{
    "status": "PENDING_DOMAIN_VALIDATION",
    "network": "production",
    "partnerDomain": "demo4.akamai.partnerdomain.net",
    "statusCheckId": "<statusCheckId specified in the request>",
    "removeVanityHostList": [
        {
            "certificateType": "shared",
            "unmappedFromAkamaiTimestamp": 1507552586834,
            "vanityHostName": "data.demo3.biz"
        }
    ],
    "batch": {
        "batchRunTimestamp": 1507553484477,
        "batchTimeLeft": 896862,
        "batchId": "<unique ID>",
        "currentTimestamp": 1507552587615,
        "domainListComplete": false,
        "lastBatchModifiedTimestamp": 1507552586943
    },
    "addVanityHostList": [
        {
            "certificateType": "shared",
            "unmappedFromAkamaiTimestamp": 1507552586943,
            "vanityHostName": "data.demo4.biz"
        }
    ],
    "certificates": {
        "current": {
            "edgeHostname": "demo4.akamai.partnerdomain.net.edgekey.net",
            "type": "shared",
            "challenges": {
                "data.demo4.biz": {
                    "status": "AWAITING_USER",
                    "domain": "<custom domain>",
                    "requestTimestamp": "2018-09-26T19:11:42Z",
                    "validatedTimestamp": null,
                    "redirectFullPath": "<URL path>",
                    "responseBody": "<key>",
                    "token": "<token>",
                    "error": null,
                    "expires": "2018-09-24T22:06:06Z",
                    "fullPath": "<URL path>",
                    "partnerDomain": "demo4.akamai.partnerdomain.net"
                }
            }
        }
```

```
    },
    "propertyId": <unique ID>,
    "statusCode": 202
}
```

Parameter	Description
`status`	The current status of the requests in the partner domain branch. Applicable statuses are discussed in *Status transitions for a request* on page 25.
`network`	The network targeted for the request. With this request, is always `production` because it is currently all that is supported.
`partnerDomain`	Your cloud partner domain. The custom domain that is enabling CDHO needs to be CNAMEd to this domain.
`statusCheckId`	The unique "status check ID" for the request.
`domainInfo`	Various information regarding domains included in the enablement request. • `enableHTTPS`: Whether or not the request was sent to enable CDHO (`true` = yes). • `endpointDomain`: The endpoint domain included in the request, if applicable. • `certificateType`: The type of certificate set in the request. • `domainListComplete`: Whether or not all domains have been added to the certificate. • `partnerDomain`: The partner domain specified in the request.
`removeVanityHostList`	For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response. If the modification batch associated with {statusCheckId} contains any PUT secure-delivery requests with `"enableHTTPS": false` that will result in removal from the certificate, then each object in this list represents the corresponding vanity domain that is being removed from the certificate. Otherwise this list will be empty.
`batch`	Various information pertaining to the batch that includes this request. For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response

Parameter	Description
	• `batchRunTimestamp`: The time the batch was initiated • `batchTimeLeft`: The time left before the batch is processed. • `batchId`: A unique ID value for the batch • `currentTimestamp`: The current time. • `domainListComplete`: Whether or not all of the domain requests in the batch have been added to the applicable circuits. • `lastBatchModifiedTimestamp`: When the batch was last modified. (For example, the last time a request was added to the batch.)
`addVanityHostList`	For diagnostic purposes only; details could change in the future. Please ensure that your integration doesn't break if this changes or is absent from the response. If the modification batch associated with `{statusCheckId}` contains any PUT secure-delivery requests with `"enableHTTPS": true` that will result in addition to the certificate, then each object in this list represents the corresponding vanity domain that is being added to the certificate. Otherwise this list will be empty. • `certificateType`: The type of certificate targeted by the request. • `unmappedFromAkamaiTimestamp`: This field is used for Akamai internal purposes. If you receive this error, you can ignore it. • `vanityHostName`: A custom domain to be added to the certificate included in the batch containing this request.
`certificates`	This includes details pertaining to the certificates being used during `status=PENDING_DOMAIN_VALIDATION`; otherwise this object will be empty. • `edgeHostName`: The Edge hostname pointing to the certificate.

Parameter	Description
	• `type`: The type of certificate in use ("shared"). • `challenges`: If applicable, a list of pending domain challenges per domain. This will only be populated if the request has reached this stage of the process, and the values displayed here may vary, based on the stage of the process. For more details, see *Domain challenge example* on page 19. • `activeVanityHostNameList`: The list of hostnames that have been added to the certificate.
`errors`	If any errors have occurred during the process, they are displayed here. For diagnostic purposes only; details could change in the future. • `certificateErrors`: This will contain a list of any certificate-related errors. • `fastDNSErrors`: This will contain a list of any FastDNS-related errors. • `partnerDomainErrors`: This will contain a list of any other errors.
`propertyId`	The unique ID for the ACE base configuration you've associated with the subcustomer who owns the custom domain.
`statusCode`	A code displayed here represents the status of the request in regards to the CDHO feature and how it is processing the request. Values shown here represent the following: • **200**: All requests have completed. • **202**: Success. This indicates that the request has been accepted and is still processing. (There are still sub-tasks that haven't completed.) • **4xx**: This indicates that at least one of the CDHO enablement sub-tasks encountered an error, and this may be the result of the formatting of the request. Check the formatting of the request and retry. If the error persists (and you are sure the request is formatted correctly), contact

Parameter	Description
	technical support. Specific 4xx codes are as follows: – **401**: Authentication failed – **403**: Authorization failed – **404**: Not found – **409**: Request conflict error • **5xx**: A server error has occurred while CDHO enablement was being processed. Try the request again. **Note:** This code is different than what is supplied as the standard HTTP status code after an API request is issued. This only applies to the current status of enablement for CDHO.

Domain challenge example

As the Certificate Authority (CA), LetsEncrypt needs to validate that the certificate requester has control to publish the challenge response to a custom Custom Domain HTTPS Orchestration (CDHO) domain.

Assume the following values apply in the example discussed here.

- **Custom Domain**: `demo2001.wsdsandbox.com`
- **Example Challenge URL**: `http://demo2001.wsdsandbox.com/.well-known/acme-challenge/BDE0iL5c6-x-t8smOqtoJ-qLJf1-95tK1flbtoSDc-w`
- **Example Challenge Response**: `BDE0iL5c6-x-t8smOqtoJ-qLJf1-95tK1flbtoSDc-w.i0hz6i52A_J3Ym5XYwGxI52x3uEhpZ7-A83gvUqhWaU`

Example domain challenge response

Below is an example of a domain challenge response from a *GET the batch status* request. (This would only be revealed for this operation, if the CDHO process was at this phase of the process.)

```
{
    "status": "valid",
    "domain": "vanity2018-01-31-001.dschultz.com",
    "requestTimestamp": "2018-02-07T19:17:38Z",
    "validatedTimestamp": null,
    "expires": "2018-04-01T14:58:47Z",
    "responseBody":
"bQCUDxdFKpIowOq2Ng-8UXnpgFHm9EK_KsIGWgzJEO4.lge5uUGo9XxzpizRfRZuSxUW_ymesk8lW
9NYvr7AO5E",
    "token": "bQCUDxdFKpIowOq2Ng-8UXnpgFHm9EK_KsIGWgzJEO4",
    "error": null,
    "fullPath": "http://vanity2018-01-31-001.dschultz.com/.well-known/acme-
```

```
challenge/bQCUDxdFKpIowOq2Ng-8UXnpgFHm9EK_KsIGWgzJEO4",
    "redirectFullPath": "http://dcv.akamai.com/.well-known/acme-challenge/
bQCUDxdFKpIowOq2Ng-8UXnpgFHm9EK_KsIGWgzJEO4"
}
```

Note: You only see pending domain challenges in the GET the batch status API response. Domain challenges that have already been satisfied are not displayed in the response.

How are domain challenges verified?

There are multiple ways a domain challenge can be verified for a certificate:

1. **The Hostname is CNAMEd to Akamai**: We automatically satisfy the challenge by making the object available at the required path.
2. **The Hostname is not CNAMEd to Akamai**: In this case, there are two options:
    a. **Redirect the request to dcv.akamai.com**. In this example, redirect the requests to the path, `.well-known/acme-challenge/BDE0iL5c6-x-t8smOqtoJ-qLJf1-95tK1flbtoSDc-w" to dcv.akamai.com` (from the Example Challenge URL, above). As a result, Akamai will also satisfy the challenge.
    b. **Publish the challenge object yourself**. Here, you should publish the Example Challenge Response (above), `BDE0iL5c6-x-t8smOqtoJ-qLJf1-95tK1flbtoSDc-w.i0hz6i52A_J3Ym5XYwGxI52x3uEhpZ7-A83gvUqhWaU` to the path `.well-known/acme-challenge/BDE0iL5c6-x-t8smOqtoJ-qLJf1-95tK1flbtoSDc-w` (from the Example Challenge URL, above).

Note: Publishing the challenge object yourself only works during the first challenge when CDHO is initially being enabled for a custom domain. It doesn't work during subsequent domain re-validations. It won't work when this custom domain validation ages out of Let'sEncrypt's validation cache. (This is because Let'sEncrypt communicates *new* values to store only to our Certificate Provisioning Service, not to the custom domain.) Because of this, CNAME'ing to Akamai, or redirecting the request to dcv.akamai.com should be more robust.

View a specific custom domain

You can view general information on a specific Custom Domain HTTPS Orchestration (CDHO) domain, and where it is in the request process.

This request returns basic information on a specific custom domain, and its current processing status. If you want more detail, you can use the *GET the batch status request*.

CDHO enabling requires v3 of the API

All API operations for CDHO support require that you use version 3 API endpoints, and that you include "v3" in the request path (for example, `/partner-api/`**`v3`**`/network/{network}/properties/...`).

This is a *different version* than what is used to perform subcustomer *registration and policy configuration*. For those required operations, you still need to use **version 2** of the API.

Make a GET a custom domain request

Method	URI
`GET`	`/partner-api/v3/network/production/properties/{property-id}/sub-properties/{domainName}/secure-delivery`

Important: Currently, only the production network is supported for use with this operation.

The variable URI request parameters (enclosed in "{ }") are as follows:

- `{propertyId}`. This is the unique identifier for the applicable ACE base configuration.
- `{domainName}`: This is the subcustomer custom domain you want to view. Any custom domain that has been included in an enablement request can be targeted.

HTTP status codes

This describes the standard HTTP status code returned after a request to the API.

Code	Explanation
200	**Success**
4xx	**Client Error**: This is a client or validation error regarding access to the API. The request itself may be improperly formatted. Correct the request and try again. Specific 4xx codes are as follows: • **401**: Authentication failed • **403**: Authorization failed • **404**: Not found • **409**: Request conflict error
5xx	**Server Error**: An internal error occurred with our server when the API made the request. Try the request again.

Important: These codes only apply to the actual acceptance of the request. With a succesfully accepted request, the resulting JSON response also includes the `statusCode` entry that offers a code that describes the actual status of the CDHO feature enablement process. (Details are included in the description of this entry in the table below.) If this standard HTTP status code results in anything other than a "202 Success" (4xx or 5xx), the JSON response with the `statusCode` entry is *not generated*.

Response examples

Here's an example of a response for a successful request:

```
{
    "status": "CHANGE_REQUEST_PENDING",
    "endpointDomain": "demo4.partnerdomain.net",
```

```
    "httpsActive": true,
    "lastUpdateTimestamp": 1508801008440,
    "partnerDomain": "demo4.akamai.partnerdomain.net",
    "vanityHostName": "data.demo4.biz",
    "propertyId": <unique ID>
}
```

Here's an example of a response containing a certificate error that has occurred with the specified custom domain:

```
{
    "status": "HTTP_ONLY",
    "endpointDomain": "demo4.partnerdomain.net",
    "httpsActive": false,
    "lastUpdateTimestamp": 1508801008440,
    "partnerDomain": "demo4.akamai.partnerdomain.net",
    "vanityHostName": "data.demo4.biz",
    "propertyId": <unique ID>
    "previousErrors": {
        "certificateErrors": [
            {
                "status": "Unable to modify
demo4.akamai.partnerdomain.net.edgekey.net, retrying later",
                "statusCheckId": "<unique key>",
                "statusCode": 409
            },
            {
                "status": Critical error modifying
demo4.akamai.partnerdomain.net.edgekey.net",
                "statusCode": 400
            }
        ]
    }
}
```

Here's an example of a response containing a Fast DNS error that occurred with the custom domain

```
{
    "status": "HTTP_ONLY",
    "httpsActive": false,
    "lastUpdateTimestamp": 1525312097955,
    "partnerDomain": "demo4.akamai.partnerdomain.net",
    "vanityHostName": "data.demo4.biz",
    "propertyId": <unique ID>
    "previousErrors": {
        "fastDNSErrors": [
            {
                "status": "Maximum of 60000 milliseconds for retries has been
reached.",
                "statusCheckId": "<unique key>",
                "statusCode": 400
            }
        ]
    }
}
```

Parameter	Description
`status`	The current status of the request for the specified custom domain. Applicable statuses are discussed in *Status transitions for a request* on page 25.
`httpsActive`	Whether or not CDHO has been enabled for the custom domain (`true` = yes).
`lastUpdateTimestamp`	For diagnostic purposes only, may change or be removed without notice. The UTC timestamp, in milliseconds indicating the last time this custom domain was updated.
`partnerDomain`	Your cloud partner domain. The custom domain that is enabling CDHO needs to be CNAMEd to this domain.
`vanityHostName`	The custom domain you targeted with this request.
`propertyId`	subcustomers are associated with a base configuration that sets the baseline for match criteria and rules to be applied for an incoming request. A single base configuration can have several subcustomers associated with it. This is the unique ID of the base configuration to which you've registered the subcustomer that owns the custom domain.
`previousErrors`	If any errors have occurred during the last request that was processed, this object is included to display them. • `certificateErrors`: This will contain a list of any certificate-related errors. • `fastDNSErrors`: This will contain a list of any Edge DNS-related errors. **Note:** For "fastDNS" objects, this refers to responses related to the "Edge DNS" product, that's required for use with CDHO. (Edge DNS was formerly known as "Fast DNS."

Assign a vanity domain to a different partner domain

You can perform specific steps to assign a vanity domain you've already configured, to a different partner domain.

Step 1: Disable HTTPS for the vanity domain

This ensures that the vanity domain is successfully unlinked from the current or old partner domain. *Disable* and include a PUT request body like this:

```
{
    "domainInfo":
    {
        "enableHTTPS": false,
        "certificateType": "shared",
        "partnerDomain": "old.akamai.partnerdomain.net",
        "domainListComplete": true
    }
}
```

The `old.akamai.partnerdomain.net` value is variable. The vanity domain is being *disassociated* from this old property domain.

Step 2: Enable HTTPS for the vanity domain with the new partner domain

Once HTTPS is successfully disabled, use *PUT /secure-delivery* to enable HTTPS on the *same* vanity domain with the *new* partner domain.

```
{
    "domainInfo": {
        "enableHTTPS": true,
        "certificateType": "shared",
        "partnerDomain": "new.akamai.partnerdomain.net",
        "domainListComplete": true
    }
}
```

The `new.akamai.partnerdomain.net` value is variable. The vanity domain is being associated with this new partner domain.

Error and failure scenarios

A request is rejected with "`HTTP status 409 - Request Conflict`" in the following scenarios:

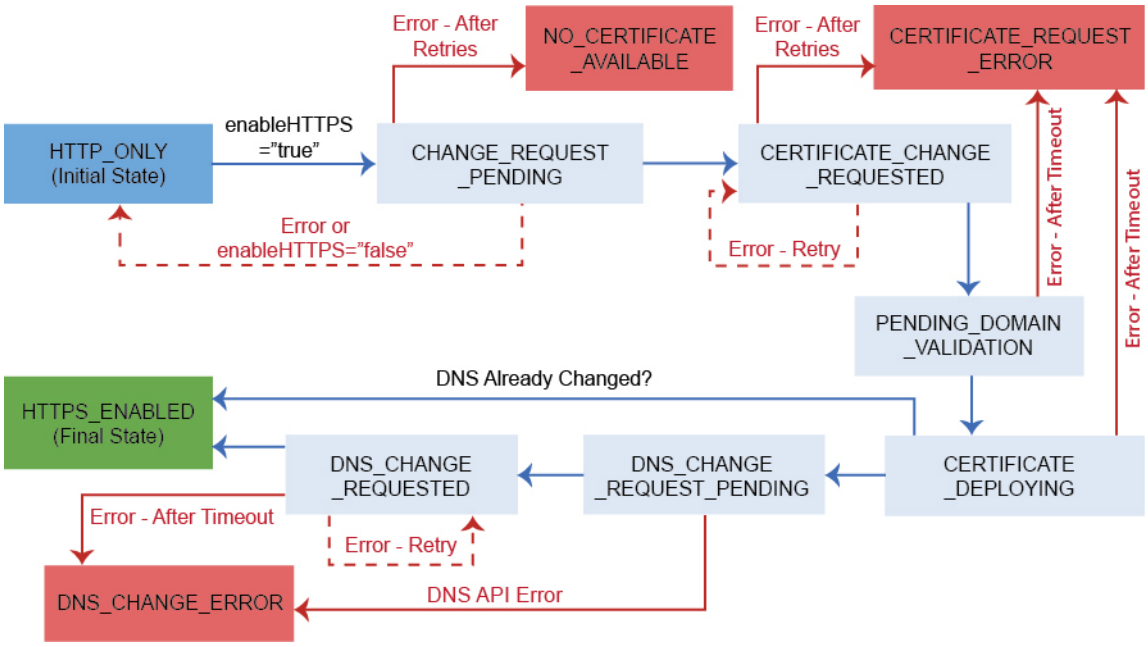
- **If the vanity domain is HTTPS enabled with the old partner domain**. You can't skip "Step 1," above.
- **If the vanity domain has pending changes against the old partner domain**. You'll see this if you attempt this and active changes are being propagated to the old partner domain. (That is, it's *request status* is anything other than "`HTTPS_ENABLED`.")

Status transitions for a request

These values are returned as the "status" parameter in the response for a request.

Enablement Request Workflow

Below is an example of the *enable CDHO* workflow, complete with applicable request statuses at each phase.



Enable Request Status[2]	Description
`HTTP_ONLY`	This status will be revealed in one of two circumstances: • The request has been sent to enable CDHO for a custom domain. (`HTTP_ONLY` is the initial status for a custom domain when the process is started.) • An error occurred, and CDHO was *not enabled*. (More detail can be found in the `errors` object in the response.)
`CHANGE_REQUEST_PENDING`	The enablement request for CDHO has been received and accepted by ACE.
`NO_CERTIFICATE_AVAILABLE`	No certificates are available. We will keep retrying until another certificate is available, or the request times out. (Two days is the default.) However, you

[2] **If any request stays in any of these statuses for more than 12 hours, please contact technical support.**

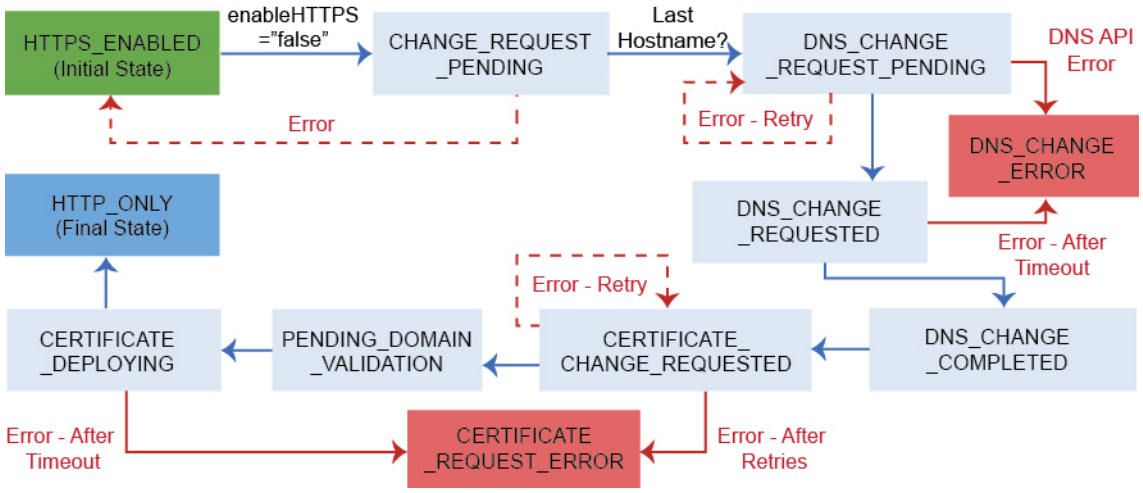
Enable Request Status[2]	Description
	should contact technical support for assistance if this persists for more than 12 hours. **Note:** This can occur if you run out of Certificate capacity. If this the case, work with technical support to get new DV certificates provisioned.
CERTIFICATE_CHANGE_REQUESTED	A partner domain batch has kicked off, and a request for updates to a certificate has been submitted to the Certificate Provisioning System (CPS). (This applies to both enable and disable requests for CDHO.)
CERTIFICATE_REQUEST_ERROR	This occurs if either of the following occur: • A fatal error has been issued from the Certificate Provisioning System • The batch timed out while still waiting for the request to finish The system attempts one retry. Contact technical support for assistance.
PENDING_DOMAIN_VALIDATION	The Certificate Provisioning System (CPS) has issued a domain challenge (via LetsEncrypt) to ensure validity of the custom domain. **Note:** A request can remain in this state for up to two days if the partner domain is not CNAMEd to Akamai, and the custom domain doesn't satisfy the domain validation challenge.
CERTIFICATE_DEPLOYING	The certificate involved in a request has been updated and is deploying for use.
DNS_CHANGE_REQUEST_PENDING	A change to the DNS is pending for the enablement request.
DNS_CHANGE_ERROR	This occurs if either of the following occur: • A fatal error has been issued from Edge DNS • The batch timed out while still waiting for the request to finish

[2] **If any request stays in any of these statuses for more than 12 hours, please contact technical support.**

Enable Request Status[2]	Description
	The system attempts one retry. Contact technical support for assistance.
`DNS_CHANGE_REQUESTED`	A change to the DNS has been requested in support of the enablement request.
`HTTPS_ENABLED`	This can be revealed in one of two circumstances: • The request has successfully completed for HTTPS enablement. • A request has been sent to disable CDHO for a custom domain. (`HTTPS_ENABLED` is the initial status for a custom domain when the process is started.)

Disable Request Workflow

Below is an example of the *disable CDHO* workflow, complete with applicable request statuses at each phase.



Disable Request Status [1]	Description
`HTTPS_ENABLED`	The request has been sent to disable CDHO for a custom domain. (`HTTPS_ENABLED` is the initial status for a custom domain when the process is started.)
`CHANGE_REQUEST_PENDING`	The disable request for CDHO has been received and accepted by ACE.

[2] **If any request stays in any of these statuses for more than 12 hours, please contact technical support.**

Disable Request Status [1]	Description
DNS_CHANGE_REQUEST_PENDING	A change to the DNS is pending for the disable request.
DNS_CHANGE_ERROR	This occurs if either of the following occur: • A fatal error has been issued from Edge DNS • The batch timed out while still waiting for the request to finish The system attempts one retry. Contact technical support for assistance.
DNS_CHANGE_REQUESTED	A change to the DNS has been requested to disable HTTPS.
DNS_CHANGE_COMPLETED	The change to the DNS has successfully completed.
CERTIFICATE_CHANGE_REQUESTED	A partner domain batch has kicked off, and a request for updates to a certificate has been submitted to the Certificate Provisioning System (CPS). (This applies to both enable and disable requests for CDHO.)
CERTIFICATE_REQUEST_ERROR	This occurs if either of the following occur: • A fatal error has been issued from the Certificate Provisioning System • The batch timed out while still waiting for the request to finish The system attempts one retry. Contact technical support for assistance.
PENDING_DOMAIN_VALIDATION	The Certificate Provisioning System (CPS) has issued a domain challenge (via LetsEncrypt) to ensure validity of the custom domain. **Note:** A request can remain in this state for up to two days if the partner domain is not CNAMEd to Akamai, and the custom domain doesn't satisfy the domain validation challenge.
CERTIFICATE_DEPLOYING	The certificate involved in a request has been updated and is deploying for use.
HTTP_ONLY	HTTPS has been successfully *disabled* for a specific custom domain.

Other statuses

These are additional statuses that you may see in a request, that are outside the Request Workflows, discussed above.

Enable Request Status	Description
`REQUEST_COMPLETED` [3]	The partner domain batch in which this request exists has completed.
`REQUEST_CANCELED` [2]	The partner domain batch in which this request exists has been canceled. **Note:** In addition to manually canceling a request, one can be canceled if a specific phase of the process times out.

[3] This status only applies to the "GET the batch status request" operations.

Add multi-CDN support

If you're using multiple content delivery networks ("multi-CDN") and require HTTPS, you can configure your Akamai Cloud Embed (ACE) environment to support them, along with Akamai's network.

The problem

Custom Domain HTTPS Orchestration (CDHO) shares domain validation subject alternative name certificates (DV SAN certs) for validation when enabling HTTPS. With a standard, Akamai-only CDN environment, a subcustomer vanity domain ("custom domain") is CNAMEd to Akamai and the **Auto Domain Validation** behavior is included in your base configuration. The behavior automatically completes vanity domain validations when Akamai receives checks against the DV SAN cert from the certificate authority. If you're using multi-CDN, Akamai probably won't get all of the traffic for these domain validation checks, so this automatic validation won't happen.

A seemingly simple solution could be to identify domains that aren't specifically CNAMEd to Akamai and remove them from the DV SAN certs. However, this can result in disablement of service (DoS) to some customer traffic.

Yet another solution could be to create custom DV SAN certs for each vanity hostname, but this isn't scalable for a large number of customers, and it's not flexible—if you initially set up ACE in the normal way and then decided to add a traffic manager app or multi-CDN configuration, you'd need to perform a lot of added work.

Possible use cases

If any of the following apply to you, multi-CDN support may be required in your environment:

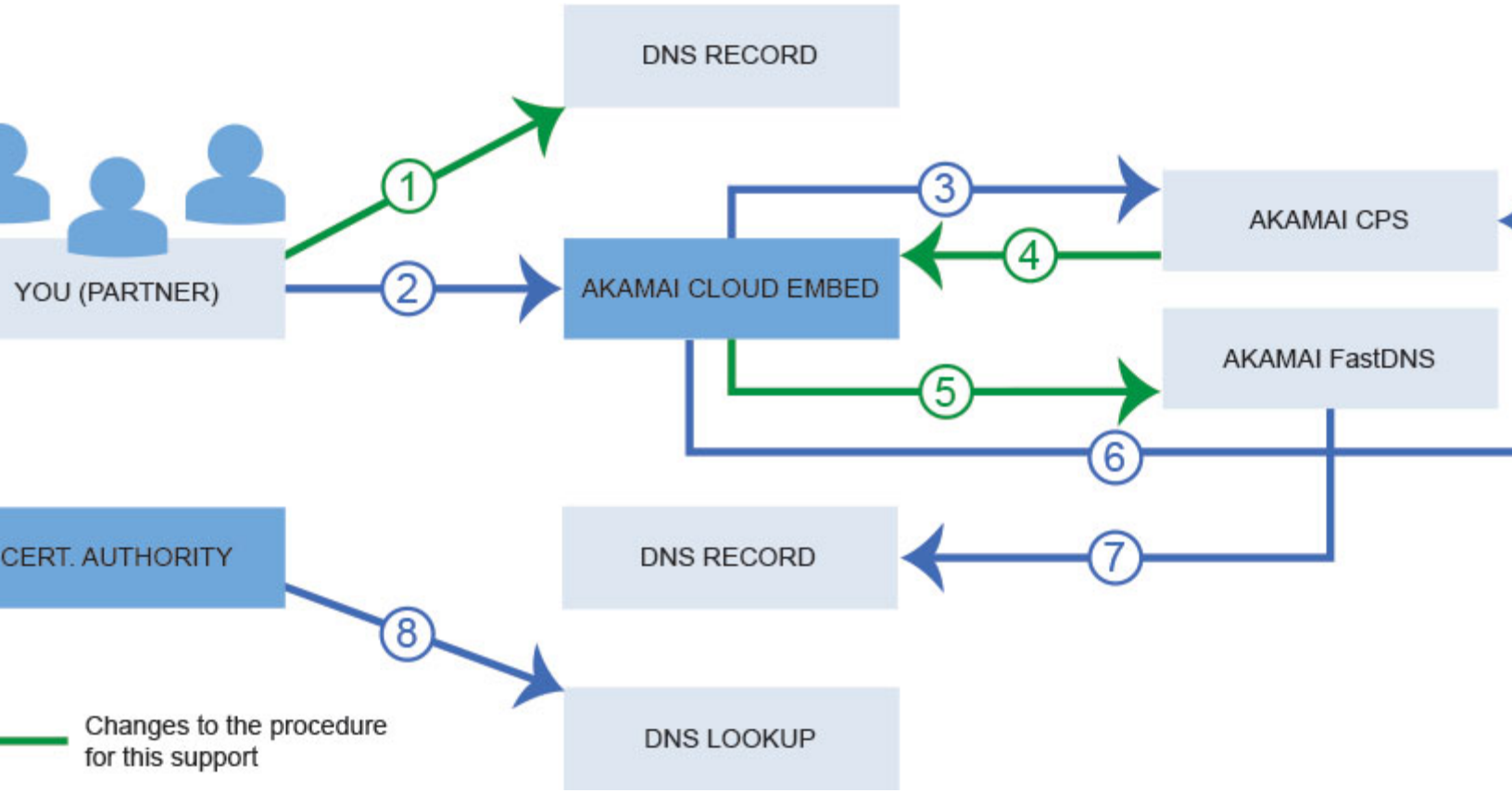
- **Traffic splitting or multi-CDN**. Do you temporarily have traffic directed away from Akamai because you're using a Traffic Manager or CDN splitter? If so, a vanity domain can fail an Akamai-specific CNAME check, but that vanity domain needs to remain in DV SAN cert.
- **Using Zone Apex or Root Domain with Akamai**. As per *DNS RFC*, you can't have a CNAME record *at the root domain*. Some DNS providers provide alternate options such as *ALIAS records* to resolve the root domain to a CDN such as Akamai. Root domains mapped to Akamai also need to remain in the DV SAN cert.
- **Mapping out of Akamai temporarily, with an intent to come back**. This is similar to the traffic splitting or multi-CDN use case. Here, you have HTTPS enabled on a custom domain, but you've temporarily mapped out of Akamai. Your delivery policy is still available, and you eventually want to CNAME that domain back to Akamai to serve HTTPS content.

Akamai's DNS Domain Validation Orchestration

The core issue in all of the use cases is that Akamai can't satisfy the HTTP challenge response for domain validation, *if the request is not served by Akamai*. So, we offer a DNS-based solution to address these challenges: DNS Domain Validation Orchestration.

The certificate authority (Let's Encrypt)—and by proxy, Akamai's Certificate Provisioning System (CPS)—provides the DNS token-based domain validation. This works well when on-boarding a new customer domain but, it doesn't work well for ongoing renewals. This is because the certificate authority issues a new DNS token for every validation or revalidation of a domain. So, we've built an orchestration layer around DNS challenges.

The workflow looks something like this:



1. A new DNS record is created: "`acme-challenge.{custom domain}. 60 IN CNAME {custom domain}.ak-acme-challenge.{your partner-dv-dns-zone}`."
2. HTTPS is enabled for `{custom domain}`, via CDHO.
3. The `{custom domain}` is added to an Akamai CPS certificate via ACE.
4. CPS fetches the DNS domain validation token from ACE.
5. ACE adds the domain validation token to your partner-specific "`ak-acme-challenge.{your partner-dv-dns-zone}`" in Edge DNS.
6. ACE triggers a domain validation token check in CPS.
7. Edge DNS records the data validation token in the DNS record.
8. The certificate authority checks for the token and issues the certificate. A DNS lookup would consist of an "`;; ANSWER SECTION`," that looks something like the following:

```
DNS Lookup

dig TXT _acme-challenge.{custom domain}
;; ANSWER SECTION:
```

```
_acme-challenge.www.example.org. 600 IN CNAME www.example.org.ak-acme-
challenge.azureedge.net.

{custom domain}.ak-acme-challenge.{your partner-dv-dns-zone}. 60 IN TXT
"{token}"
```

The multi-CDN workflow

To implement multi-CDN support you—as the Akamai partner—need to apply a specific workflow. Some steps are required by the partner, and others are performed by the subcustomer.

Partner requirements for multi-CDN

As an Akamai partner using ACE to distribute our CDN to your customers ("subcustomers"), you need to perform the following steps *a single time* to support multi-CDN.

1. Follow the same procedure to set up a primary DNS configuration using Akamai *Edge DNS* for your partner domain. Use the Zone name, "`ak-acme-challenge.{partner-dv-dns-zone}`." For example, this might look like this, for Azure:

   ```
   ak-acme-challenge.azureedge.net
   ```

2. You need to set up a portal to enable CDHO for ACE. Set it up using the process discussed in *Enable CDHO* on page 7.

   > **Note:** You can only use a single DNS DV Zone with multiple certificates that belong to the same Akamai Contract ID.

3. Update your portal to instruct subcustomers to add a DNS record, similar to this: "`_acme-challenge.{custom domain}. 60 IN CNAME {custom domain}.ak-acme-challenge.{partner-dv-dns-zone}`." Using the Azure example above, this would look as follows if the subcustomer domain was "`www.moviestreams2020.com`":

   ```
   _acme-challenge.moviestreams2020. 60 IN CNAME moviestreams2020.com.ak-
   acme-challenge.azureedge.net
   ```

The subcustomer workflow

Your customers need to perform the following to use multi-CDN support for CDHO.

1. The subcustomer uses your portal to add the DNS record, using their vanity domain. For example:

   ```
   _acme-challenge.moviestreams2020. 60 IN CNAME moviestreams2020.com.ak-
   acme-challenge.azureedge.net
   ```

2. The subcustomer uses the portal you created to enable CDHO for ACE.

3. The subcustomer waits for enablement to complete. You can *view the status* of enablement requests.

What happens next?

Once the service has been properly enabled, the remainder of the workflow is automated.

1. ACE uses the Akamai Certificate Provisioning System (CPS) APIs to fetch DNS challenge responses.
2. ACE uses Akamai Edge DNS APIs to add TXT responses to the zone you created—`multi-CDN--ak-acme-challenge.{partner-dv-dns-zone}`—for the appropriate domains.
3. ACE uses the CPS API again to let CPS know that the DNS challenge response is available.

Known issues with multi-CDN support

There are some known issues that apply to the use of multi-CDN with ACE. Review them before trying to configure this support.

Issue	Description
Using more than one CA is not fully supported	This can be an issue if you need to perform DNS DV verification with other certificate authorities (CA). However, you can use other means of domain validations with other CAs/CDNs, or even put the appropriate responses for `_acme-challenge.www.mydomain.com` for other CDNs/CAs when required and retain the CNAME to *ak-acme-challenge* on a long-term basis.
MDC doesn't function properly	This is because the MDC hostname must be present in the CNAME chain, and multi-CDN requires a specific, fixed CNAME chain.

How to implement automated slot matching

Slot matching looks to speed-up the process required to associate multiple vanity hostnames with the secure, Akamai edge hostname used to access your ACE property.

Traditionally, you need to use Property Manager to configure each individual subcustomer hostname that you want to securely access your ACE property—you manually include them in individual, secure "property hostname to edge hostname associations." You then CNAME your hostnames to the edge hostname set in the association. A client request to one of your hostnames is resolved to the Akamai edge hostname, a certificate is verified, and your ACE property is read.

What is slot matching?

Slot matching lets you skip the manual hostname association process. Instead, an edge hostname in your base configuration is mapped to one or more "slots." A slot is a numeric value that Akamai assigns to a secure, standard TLS certificate that was created using the Akamai Certificate Provisioning System (CPS). You still need to CNAME subcustomer hostnames to this edge hostname in your DNS. When a request comes from one of these subcustomer hostnames, it's resolved to the edge hostname and its associated slots are used to access the applicable certificate for secure access.

How is slot matching automated?

In the past, slot matching for your ACE environment required a lengthy, manual process that required permissions, provisioning, and data file deployment—all of which were performed behind the scenes by Akamai personnel and could take a week or longer. While the process is still performed internally by Akamai, it is much quicker to implement, taking approximately an hour. In addition, the new automated process lets you see your slot matching configuration in your ACE base configuration.

Before you begin

Before you can use slot matching, you need to meet some prerequisites:

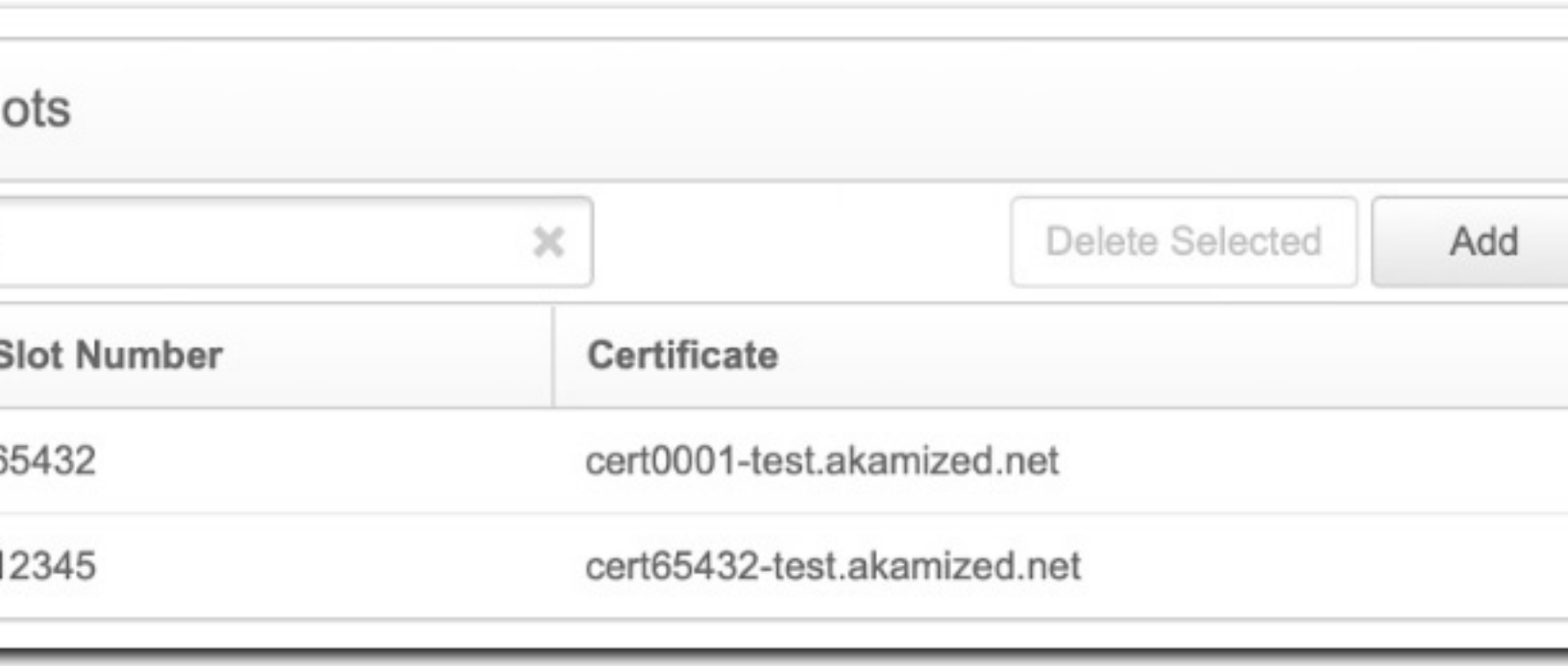
- **Get slot matching added to your contract**. Work with your account representative to get the `WholesaleDelivery::SlotMatching_Trial` service added to your contract.
- **This applies to existing ACE customers, only**. Currently, you can only add slot matching if you already have ACE configured and in use. You need to have a base configuration that already has CDHO set up for it (as described in this guide.)

The automated slot matching workflow

Review what's here for guidelines on managing slot matching in your ACE base configuration.

How to activate slot matching

You need to work with your account representative ("rep") to add slot matching to your ACE base configuration. Once added, you can review its settings in the Slots pane in your property.



The Slots field shows each "slot" that's been configured, listed by its unique numeric identifier. The Certificate column lists the common name (CN) from the secure certificate that's been assigned to that slot. Requests that include a CN will use that slot. All existing slots that have been enabled with slot matching for the property are listed.

Note: Property Manager requires that every property have a standard Property Hostname set up. You can't remove all of your Property Hostnames and exclusively use slot matching.

How to deactivate or modify slot matching

You'll need to work with your account rep to remove slot matching, or perform any changes. The benefit to the "automated" method is that these changes will take considerably less time compared to the old, manual method.

What is dangling slot matching?

If a slot that's been set up for slot matching is deleted, its matching certificate (and CNs) are still valid. This is referred to as "dangling slot matching." We have an internal monitoring tool in place that recognizes this scenario and alerts your account rep, so the proper removal (or fix) can be performed.

Slots are the same in both Staging and Production

With slot matching added to your base configuration property, once you activate that property—on either the staging or production networks—that property is locked and can't be edited further. You'll need to create a new version of the property to edit *any* of its settings.

Known issues with slot matching

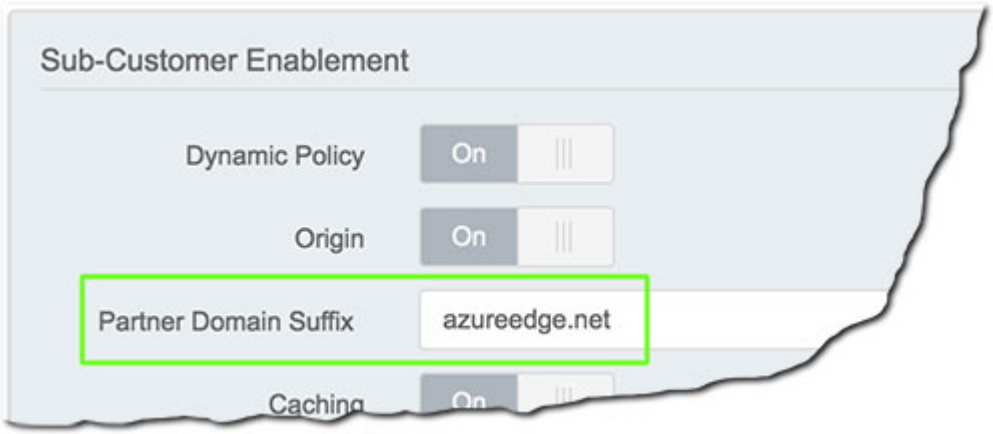
There are some known issues that apply to slot matching with ACE. Review them before adding this functionality to your property.

The Purge by URL functionality doesn't work with slot matching

When Purge by URL receives a request, it looks for cache key information for that URL. With slot matching configurations, the data required to compute the cache key isn't available because the system can't determine the specific "slot" that was used.

There is a workaround for this. You can use a wildcard hostname match to find the right configuration during the purge request. You need to add a suffix to the domain during Purge by URL requests.

1. Set the applicable hostname suffix in the ACE base configuration as the **Partner Domain Suffix**. Work with your account representative to ensure that you include the applicable hostname suffix.



2. When using Purge by URL, include the hostname suffix in the hostname of the URL to be purged. For example, if your hostname suffix is "`azureedge.net`," and you need to purge the URL `www.xyz.com/foo/bar.png`, the Purge by URL request should be set to `www.xyz.com.purge.azureedge.net/foo/bar.png`.

You can't rollback to an older property version once a slot is removed

Take caution when you have a slot removed. Once it's fully removed, *it's gone*. You won't be able to rollback to a previous version of the property that included this slot.

Notice

Akamai secures and delivers digital experiences for the world's largest companies. Akamai's Intelligent Edge Platform surrounds everything, from the enterprise to the cloud, so customers and their businesses can be fast, smart, and secure. Top brands globally rely on Akamai to help them realize competitive advantage through agile solutions that extend the power of their multi-cloud architectures. Akamai keeps decisions, apps, and experiences closer to users than anyone — and attacks and threats far away. Akamai's portfolio of edge security, web and mobile performance, enterprise access, and video delivery solutions is supported by unmatched customer service, analytics, and 24/7/365 monitoring. To learn why the world's top brands trust Akamai, visit *www.akamai.com*, *blogs.akamai.com*, or *@Akamai* on Twitter. You can find our global contact information at *www.akamai.com/locations*.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers, and contact information for all locations are listed on *www.akamai.com/locations*.

Published 11/2021