Akamai Provider

The Akamai provider is used to interact with the Akamai platform for content delivery, security, and performance.

To use this provider you must create API credentials valid for each service you want to use. Learn more here (https://developer.akamai.com/introduction/Prov_Creds.html).

Use the navigation to the left to read about the available resources.

Example Usage

```
# Configure the Akamai Provider
provider "akamai" {
    property {
      host = "${var.akamai_host}"
      access_token = "${var.akamai_access_token}"
      client_token = "${var.akamai_client_token}"
      client_secret = "${var.akamai_client_secret}"
    }
}


# Create a Property
resource "akamai_property" "example_property" {
  name = "www.example.org"

  # ...
}
```

Authentication

You must specify credentials for each service used. Currently the provider supports `property` (PAPI) and `dns` services.

You may use either a the Akamai standard `.edgerc` file, or you can specify the credentials inline.

Inline Credentials

To specify credentials inline, use the `property` or `dns` block to define credentials.

Argument Reference

- `property` — (Optional) Provide credentials for the Property Manager API (papi)
  - `host` — (Required) The credential hostname
  - `access_token` — (Required) The credential access_token
  - `client_token` — (Required) The credential client_token

- `client_secret` — (Required) The credential client_secret
- `max_body` — (Optional) The credential max body to sign (in bytes, Default: `131072`)

- `dns` — (Optional) Provide credentials for the Fast DNS API (config-dns)
  - `host` — (Required) The credential hostname
  - `access_token` — (Required) The credential access_token
  - `client_token` — (Required) The credential client_token
  - `client_secret` — (Required) The credential client_secret
  - `max_body` — (Optional) The credential max body to sign (in bytes, Default: `131072`)

Using an .edgerc file

The Akamai provider uses the standard Akamai Edgegrid authentication configuration, providing a path to an `.edgerc` INI file, with one or more credential sections for each service. You can read more about the .edgerc file here (https://developer.akamai.com/introduction/Conf_Client.html#edgercformat).

To use an `.edgerc` file, you should configure the provider to specify a path. By default it will look in the current users home directory.

Usage:

```
provider "akamai" {
    edgerc =  "~/.edgerc"
}
```

You should specify separate credentials for each service. You may use the same .edgerc section for multiple services. By default the default section is used.

```
provider "akamai" {
    edgerc = "~/.edgerc"
    property_section = "papi"
    dns_section = "dns"
}
```

Argument Reference

The following arguments are supported in the `provider` block:

- `edgerc` - (Optional) The location of the `.edgerc` file containing credentials. Default: `$HOME/.edgerc`
- `property_section` — (Optional) The credential section to use for the Property Manager API (PAPI). Default: `default`.
- `dns_section` — (Optional) The credential section to use for the Config DNS API. Default: `default`.
- `cps_section` — (Optional) The credential section to use for the Config CPS. Default: `default`.

Environment Variables

You can also specify credential values using environment variables. Environment variables take precedence over the contents of the `.edgerc` file.

Create environment variables in the format:

```
AKAMAI{_SECTION_NAME}_*
```

For example, if you specify `property_section = "papi"` you would set the following ENV variables:

- `AKAMAI_PAPI_HOST`
- `AKAMAI_PAPI_ACCESS_TOKEN`
- `AKAMAI_PAPI_CLIENT_TOKEN`
- `AKAMAI_PAPI_CLIENT_SECRET`
- `AKAMAI_PAPI_MAX_BODY` (optional)

If the section name is `default` , you can omit it, instead using:

- `AKAMAI_HOST`
- `AKAMAI_ACCESS_TOKEN`
- `AKAMAI_CLIENT_TOKEN`
- `AKAMAI_CLIENT_SECRET`
- `AKAMAI_MAX_BODY` (optional)

akamai_authorities_set

Use `akamai_authorities_set` datasource to retrieve a contracts authorities set for use when creating new zones.

Example Usage

Basic usage:

```
data "akamai_authorities_set" "example" {
     contract = "ctr_#####"
}
```

Argument Reference

The following arguments are supported:

- `contract` — (Required) The contract ID.

Attributes Reference

The following are the return attributes:

- `authorities` — A list of authorities

akamai_contract

Use `akamai_contract` data source to retrieve a group id.

Example Usage

Basic usage:

```
data "akamai_contract" "example" {
     group = "group name"
}

resource "akamai_property" "example" {
    contract = "${data.akamai_contract.example.id}"
    ...
}
```

Argument Reference

The following arguments are supported:

- `group` — (Optional) The group within which the contract can be found.

Attributes Reference

The following are the return attributes:

- `id` — The contract ID.

akamai_cp_code

Use `akamai_cp_code` data source to retrieve a group id.

Example Usage

Basic usage:

```
data "akamai_cp_code" "example" {
     name = "cpcode name"
     group = "grp_#####"
     contract = "ctr_#####"
}

resource "akamai_property" "example" {
    contract = "${data.akamai_cpcode.example.id}"
    ...
}
```

Argument Reference

The following arguments are supported:

- `name` — (Required) The CP code name.
- `group` — (Required) The group ID
- `contract` — (Required) The contract ID

Attributes Reference

The following are the return attributes:

- `id` — The CP code ID.

akamai_group

Use `akamai_group` data source to retrieve a group id.

Example Usage

Basic usage:

```
data "akamai_group" "example" {
    name = "group name"
}


resource "akamai_property" "example" {
    group    = "${data.akamai_group.example.id}"
    ...
}
```

Argument Reference

The following arguments are supported:

- `name` — (Required) The group name.
- `contract` — (Optional) The contract ID

Attributes Reference

The following are the return attributes:

- `id` — The group ID.

akamai_property_rules

The `akamai_property_rules` data source allows you to configure a nested block of property rules, criteria, and behaviors. A property's main functionality is encapsulated in its set of rules and rules are composed of the matches and the behavior that applies under those matches.

Example Usage

Basic usage:

```
data "akamai_property_rules" "example" {
  rules { # Default rule

    behavior { # Downstream Cache behavior
      name = "downstreamCache"
      option { # behavior option
        key = "behavior"
        value = "TUNNEL_ORIGIN"
      }
    }

    rule { # "Performance" child rule
      name = "Performance"

      rule { # "JPEG Images" child rule
        name = "JPEG Images"

        behavior { # Adaptive Image Compression behavior
          name = "adaptiveImageCompression"

          # Options
          option {
            key = "tier1MobileCompressionMethod"
            value = "COMPRESS"
          }
          option {
            key = "tier1MobileCompressionValue"
            value = "80"
          }
          option {
            key = "tier2MobileCompressionMethod"
            value = "COMPRESS"
          }
        }
      }
    }
  }
}

resource "akamai_property" "example" {
  rules = "${data.akamai_property_rules.example.json}"

  // ...
}
```

Argument Reference

The following arguments are supported:

The `rule` block supports:

- `is_secure` — (Optional) Whether the property is a secure (Enhanced TLS) property or not (top-level only).
- `criteria` — (Optional) One or more criteria to match requests on.

- `behavior` — (Optional) One or more behaviors to apply to requests that match.
- `rule` — (Optional) Child rules (may be nested five levels deep).

The `criteria` block supports:

- `name` — (Required) The name of the criteria.
- `option` — (Optional) One or more options for the criteria.

The `behavior` block supports:

- `name` — (Required) The name of the behavior.
- `option` — (Optional) One or more options for the behavior.

The `option` block supports:

- `key` — (Required) The option name.
- `value` — (Optional) A single value for the option.
- `values` — (Optional) An array of values for the option.

One of `value` or `values` is required.

Attributes Reference

The following are the return attributes:

- `json` — The resulting JSON rule tree

Appendix

Domain Suffixes for Different Edge Hostname Types

Each type of edge hostname has its own domain suffix. Knowing which one to use is important when setting the cnameTovalue:

Edge Hostname Type	Domain Suffix
Enhanced TLS	`edgekey.net`
Standard TLS	`edgesuite.net`
Shared Cert	`akamaized.net`
Non-TLS	`edgesuite.net`

Secure Hostnames

For secure hostnames you must include the certificate enrollment ID in your `akamai_edge_hostname` resource (/docs/providers/akamai/r/edge_hostname.html).

1. Retrieve the enrollment-id from the CPS CLI (https://github.com/akamai/cli-cps)
2. Enter the ID as the certificate attribute.

Common Product IDs

Leveraging Product IDs in your setup requires you to retrieve the ID for the specific Akamai product you are using. The following is a list of commonly used product IDs for different products:

Product	Code
Ion/Ion Premier	`prd_SPM`
Ion Standard	`prd_Fresca`
Dynamic Site Accelerator	`prd_Site_Accel`
Rich Media Accelerator	`prd_Rich_Media_Accel`
Ion Standard	`prd_Fresca`

Frequently Asked Questions

Migrating a property to Terraform

If you have an existing property you would like to migrate to Terraform we recommend the following process:

1. Export your rules.json from your existing property (using the API, CLI, or Control Center)
2. Create a terraform configuration that pulls in the rules.json
3. Assign a temporary hostname for testing (hint: you can use the edge hostname as the public hostname to allow testing without changing any DNS)
4. Activate the property and test thoroughly
5. Once testing has concluded successfully, update the configuration to assign the production public hostnames
6. Activate again

Once this second activation completes Akamai will automatically route all traffic to the new property and will deactivate the original property entirely if all hostnames are no longer pointed at it.

Since Terraform assumes it is the de-facto state for any resource it leverages, we strongly recommend creating a new property based off an existing rules.json tree when starting with the provider to mitigate any risks to existing setups.

Dynamic Rule Trees Using Templates

If you wish to inject terraform data into your rules.json, for example an origin address, you can use Terraform templates to do so like so:

First decide where your origin value will come from, this could be another Terraform resource such as your origin cloud provider, or it could be a terraform input variable like this:

```
variable "origin" { }
```

Because we have not specified a default, a value is required when executing the config. We can then reference this variable using `${vars.origin}` in our template data source:

```
data "template_file" "init" {
  template = "${file("rules.json")}"
  vars = {
    origin = "${vars.origin}"
  }
}
```

Then in our `rules.json` we would have:

```
{
  "name": "origin",
  "options": {
    "hostname": "**${origin}**",
    ...
  }
},
```

You can also inject entire JSON blocks using the same mechanism:

```
{
    "rules": {
        "behaviors": [
            ${origin}

        ]
    }
}
```

Upgrading the Akamai Provider

To upgrade the provider, simply run `terraform init` again, and all providers will be updated to their latest version within specified version constraints.

Get Started with Property Management

The Akamai Provider for Terraform provides you the ability to automate the creation, deployment, and management of property configuration and activation, edge hostnames, and CP Codes.

Configure the Terraform Provider

Set up your credential files as described in Get Started with Akamai APIs (https://developer.akamai.com/api/getting-started), and include authorization for the Property Manager API

Next, we need to configure the provider with our credentials. This is done using a provider configuration block.

1. Create a new folder called `terraform`
2. Inside the new folder, create a new file called `akamai.tf` .
3. Add the provider configuration to your `akamai.tf` file:

```
provider "akamai" {
    property {
      host = "..."
        access_token = "..."
        client_token = "..."
        client_secret = "..."
    }

    dns {
        host = "..."
        access_token = "..."
        client_token = "..."
        client_secret = "..."
    }
}
```

> **Note:** You only need to specify one of the property or dns blocks. The values correspond to the Akamai EdgeGrid API credential parts of the same names, and can be set using any mechanism through which you can input strings, such as Terraform input variables, which can use the environment, or from another provider such as Vault.

Prerequisites

To create a property there are a number of dependencies you must first meet:

- **Contract ID**: The ID of the contract under which the property, CP Code, and edge hostnames will live
- **Group ID**: The ID of the group under which the property, CP Code, and edge hostnames will live
- **Edge hostname:** The Akamai edge hostname for your property. You can create a new one or reuse an existing one.
- **Origin hostname:** The origin hostname you want your configuration to point to

- **Product:** The Akamai Product ID (/docs/providers/akamai/g/appendix.html#common-product-ids) for the product you are using (Ion, DSA, etc.)
- **Rules configuration**: The rules.json file contains the base rules for the property. (learn how to leverage the rules.json tree from an existing property here (/docs/providers/akamai/g/faq.html#migrating-a-property-to-terraform))

Retrieving The Contract ID

You can fetch your contract ID automatically using the `akamai_contract` data source (/docs/providers/akamai/d/contract.html). To fetch the default contract ID no attributes need to be set:

```
data "akamai_contract" "default" {

}
```

Alternatively, if you have multiple contracts, you can specify the `group` which contains it:

```
data "akamai_contract" "default" {
  group = "default"
}
```

You can now refer to the contract ID using the `id` attribute: `data.akamai_contract.default.id`.

Retrieving The Group ID

Similarly, you can fetch your group ID automatically using the `akamai_group` data source (/docs/providers/akamai/d/group.html). To fetch the default group ID no attributes need to be set:

```
data "akamai_group" "default" {

}
```

To fetch a specific group, you can specify the `name` argument:

```
data "akamai_group" "default" {
  name = "example"
}
```

You can now refer to the group ID using the `id` attribute: `data.akamai_group.default.id`.

Managing Edge Hostnames

Whether you are reusing an existing Edge Hostname or creating a new one, you use the `akamai_edge_hostname` resource. The following will create the `example.com.edgesuite.net` edge hostname:

```
resource "akamai_edge_hostname" "example" {
    group = "${data.akamai_group.default.id}"
    contract = "${data.akamai_contract.default.id}"
    product = "prd_SPM"
    edge_hostname = "example.com.edgesuite.net"
}
```

> **Note:** Notice that we're using variables from the previous section to reference the group and contract IDs. These will automatically be replaced at runtime by Terraform with the actual values.

This will create a non-secure hostname, to create a secure hostname, you must specify a certificate enrollment ID, using the `certificate` argument:

```
resource "akamai_edge_hostname" "example" {
    group = "${data.akamai_group.default.id}"
    contract = "${data.akamai_contract.default.id}"
    product = "prd_SPM"
    edge_hostname = "example.com.edgesuite.net"
    certificate = "<CERTIFICATE ENROLLMENT ID>"
}
```

This will create a Standard TLS secure hostname, to create an Enhanced TLS hostname, use the `edgekey.net` domain suffix for the `edge_hostname` instead.

> **Note:** This resource does not automatically make the property secure. You will need the `is_secure` flag set to `true` in your rule tree as well — this can be set in your `akamai_property` or `akamai_property_rules` resources, or in your `rules.json` file.

Property Rules

A property contains the delivery configuration, or rule tree, which determines how requests are handled. This rule tree is usually represented using JSON, and is often refered to as `rules.json` .

You can specify the rule tree as a JSON string, using the `rules` argument of the `akamai_property` resource (/docs/providers/akamai/r/property.html#rules).

We recommend storing the rules JSON as a JSON file on disk and ingesting it using Terraforms `local_file` data source. For example, if our file is called `rules.json` , we might create a `local_file` data source called `rules` . We specify the path to `rules.json` using the `filename` argument:

```
data "local_file" "rules" {
    filename = "rules.json"
}
```

We can now use `${data.local_file.rules.content}` to reference the file contents in the `akamai_property.rules` argument.

Creating a Property

The property itself is represented by an `akamai_property` resource (/docs/providers/akamai/r/property.html). Add this new block to your `akamai.tf` file after the provider block.

To define the entire configuration, we start by opening the resource block and give it a name. In this case we're going to use the name "example".

Next, we set the name of the property, contact email id, product ID, group ID, CP code, property hostname, and edge hostnames.

Finally, we setup the property rules: first, we should specify the `rule format` argument (/docs/providers/akamai/r/property.html#rule_format), as well as passing the `rules.json` data to `rules` argument.

Once you're done, your property should look like this:

```
resource "akamai_property" "example" {
    name = "xyz.example.com"                        # Property Name
    contact = ["user@example.org"]                  # User to notify of de/activations
    product  = "prd_SPM"                            # Product Identifier (Ion)
    group    = "${data.akamai_group.default.id}"    # Group ID variable
    contract = "${data.akamai_contract.default.id}" # Contract ID variable
    hostnames = {                                   # Hostname configuration
        # "public hostname" = "edge hostname"
        "example.com" = "example.com.edgesuite.net"
        "www.example.com" = "example.com.edgesuite.net"
    }
    rule_format = "v2018-02-27"                     # Rule Format
    rules = "${data.local_file.rules.content}"      # JSON Rule tree
}
```

> **Note:** If you are creating a secure property (using TLS), you need to set the `is_secure` attribute to true unless it already set in your `rules.json` . If specified in the property resource, it will *override* the value in `rules.json` .

Initialize the Provider

Once you have your configuration complete, save the file. Then switch to the terminal to initialize terraform using the command:

```
$ terraform init
```

This command will install the latest version of the Akamai provider, as well as any other providers necessary (such as the local provider). To update the Akamai provider version after a new release, simply run `terraform init` again.

Test Your Configuration

To test your configuration, use `terraform plan` :

```
$ terraform plan
```

This command will make Terraform create a plan for the work it will do based on the configuration file. This will not actually make any changes and is safe to run as many times as you like.

Apply Changes

To actually create our property, we need to instruct terraform to apply the changes outlined in the plan. To do this, in the terminal, run the command:

```
$ terraform apply
```

Once this completes your property will have been created. You can verify this in Akamai Control Center (https://control.akamai.com) or via the Akamai CLI (https://developer.akamai.com/cli). However, the property configuration has not yet been activated, so let's do that next!

Activate your property

To activate your property we need to create a new `akamai_property_activation` resource (/docs/providers/akamai/r/property_activation.html). This resource manages the activation for a property, allowing you to specify which network and what version to activate.

You will need to set the `property` ID and `version` arguments, which can both be set from the `akamai_property` resource. You should then set the `network` to `STAGING` or `PRODUCTION`. You should also set the `contact` email address.

Lastly, you need to affirm that you wish to activate the property, by setting the `activate` argument to `true`.

```
resource "akamai_property_activation" "example" {
    property = "${akamai_property.example.id}"
    version = "${akamai_property.example.version}"
    network = "STAGING"
    contact = ["user@example.org"]
    activate = true
}
```

Test & Deploy Property Activation

Like the property itself, we should test our configuration with this command:

```
$ terraform plan
```

This time you will notice how the property is not being modified while the activation is being added to the plan.

Again, as with our property configuration we can apply our changes using:

```
$ terraform apply
```

This will activate the property on the staging network.

akamai_cp_code

The `akamai_cp_code` resource allows you to create or re-use CP Codes.

If the CP Code already exists it will be used instead of creating a new one.

Example Usage

Basic usage:

```
resource "akamai_cp_code" "cp_code" {
  name = "My CP Code"
  contract = "${akamai_contract.contract.id}"
  group = "${akamai_group.group.id}"
  product = "prd_SPM"
}
```

Argument Reference

The following arguments are supported:

- `name` — (Required) The CP Code name
- `contract` — (Required) The Contract ID
- `group` — (Required) The Group ID
- `product` — (Required) The Product ID

akamai_dns_record

The `akamai_dns_record` provides the resource for configuring a dns record to integrate easily with your existing DNS infrastructure to provide a secure, high performance, highly available and scalable solution for DNS hosting.

Example Usage

Basic usage:

```
# A record
resource "akamai_dns_record" "origin" {
    zone = "origin.org"
    name = "origin.example.org"
    recordtype =  "A"
    active = true
    ttl =  30
    target = ["192.0.2.42"]
}

# CNAME record
resource "akamai_dns_record" "www" {
    zone = "example.com"
    name = "www.example.com"
    recordtype =  "CNAME"
    active = true
    ttl =  600
    target = "origin.example.org.edgesuite.net"
}
```

Argument Reference

The following arguments are supported:

- `name` — (Required) The name of the record. The name is an owner name, that is, the name of the node to which this resource record pertains.
- `zone` — (Required) Domain zone, encapsulating any nested subdomains.
- `recordType` — (Required) The DNS record type.
- `active` — (Required,Boolean) Whether the record is active.
- `ttl` — (Required,Boolean) The TTL is a 32-bit signed integer that specifies the time interval that the resource record may be cached before the source of the information should be consulted again. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached. Zero values can also be used for extremely volatile data.
- `target` — (Required) A domain name that specifies the canonical or primary name for the owner. The owner name is an alias.

akamai_dns_zone

The `akamai_dns_zone` provides the resource for configuring a dns zone to integrate easily with your existing DNS infrastructure to provide a secure, high performance, highly available and scalable solution for DNS hosting.

Example Usage

Basic usage:

```
resource "akamai_dns_zone" "demozone" {
    contract = "ctr_XXX"
    group = 100

    zone = "example.com"
    type =  "primary"
    masters = [
      "1.2.3.4",
      "1.2.3.5"
    ]

    comment =  "some comment"
    signandserve = true
}
```

Argument Reference

The following arguments are supported:

- `contract` — (Required) The contract ID.
- `group` — (Required) The currently selected group ID.
- `zone` — (Required) Domain zone, encapsulating any nested subdomains.
- `type` — (Required) Whether the zone is primary or secondary.
- `masters` — (Required) The names or addresses of the customer's nameservers from which the zone data should be retrieved.
- `comment` — (Required) A descriptive comment.
- `sign_and_serve` — (Required) Whether DNSSEC Sign&Serve is enabled.

akamai_edge_hostname

The `akamai_edge_hostname` provides the resource for configuring a secure edge hostname that determines how requests for your site, app, or content are mapped to Akamai edge servers.

An edge hostname is the CNAME target you use when directing your end user traffic to Akamai. In a typical DNS CNAME, your www.customer.com (http://www.customer.com) hostname corresponds to an edge hostname of www.customer.com.edgesuite.net (http://www.customer.com.edgesuite.net).

Example Usage

Basic usage:

```
resource "akamai_edge_hostname" "terraform-demo" {
    product  = "prd_####"
    contract = "ctr_####"
    group    = "grp_####"
    edge_hostname = "www.example.org.edgesuite.net"
}
```

Argument Reference

The following arguments are supported:

- `contract` — (Required) The contract ID.
- `group` — (Required) The group ID.
- `product` — (Required) The product ID.
- `edge_hostname` — (Required) One or more edge hostnames (must be <= to the number of public hostnames).
- `ipv4` — (Optional) Whether the property supports IPv4 to origin. (Default: `true`).
- `ipv6` — (Optional) Whether the property supports IPv6 to origin. (Default: `false`).
- `certificate` — (Optional) The certificate enrollment ID.

Attributes Reference

The following attributes are returned:

- `ip_behavior` — Whether the hostname uses `IPV4` , `IPV6` or `IPV6_COMPLIANCE` .

akamai_property_activation

The `akamai_property_activation` provides the resource for activating a property in the appropriate environment. Once you are satisfied with any version of a property, an activation deploys it, either to the Akamai staging or production network. You activate a specific version, but the same version can be activated separately more than once.

Example Usage

Basic usage:

```
resource "akamai_property_activation" "example" {
     property = "${akamai_property.example.id}"
     network  = "STAGING"
     activate = "${var.akamai_property_activate}"
     contact  = ["user@example.org"]
}
```

Argument Reference

The following arguments are supported:

- `property` — (Required) The property ID.
- `version` — (Optional) The version to activate. When unset it will activate the latest version of the property.
- `network` — (Optional) Akamai network to activate on. Allowed values `staging` or `production` (Default: `staging`).
- `activate` — (Optional, boolean) Whether to activate the property on the network. (Default: `true`).
- `contact` — (Required) One or more email addresses to inform about activation changes.

Attribute Reference

The follwing attributes are returned:

- `status` — the current activation status

akamai_property

The `akamai_property` resource represents an Akamai property configuration, allowing you to create, update, and activate properties on the Akamai platform.

Example Usage

Basic usage:

```
resource "akamai_property" "example" {
    name    = "terraform-demo"
    contact = ["user@example.org"]

    product  = "prd_SPM"
    contract = "ctr_####"
    group    = "grp_####"
    cp_code  = "cpc_#####"

    hostnames = {
      "example.org" = "example.org.edgesuite.net"
      "www.example.org" = "example.org.edgesuite.net"
      "sub.example.org" = "sub.example.org.edgesuite.net"
    }

    rule_format = "v2018-02-27"
    rules       = "${data.local_file.terraform-demo.content}"
    variables   = "${akamai_property_variables.origin.json}"
}
```

Argument Reference

The following arguments are supported:

Property Basics

- `account` — (Required) The account ID.
- `contract` — (Optional) The contract ID.
- `group` — (Optional) The group ID.
- `product` — (Optional) The product ID. (Default: `prd_SPM` for Ion)
- `name` — (Required) The property name.
- `contact` — (Required) One or more email addresses to inform about activation changes.
- `hostnames` — (Required) A map of public hostnames to edge hostnames (e.g. `{"example.org" = "example.org.edgesuite.net"}`)

- `is_secure` — (Optional) Whether the property is a secure (Enhanced TLS) property or not.

Property Rules

- `rules` — (Required) A JSON encoded string of property rules (see: `akamai_property_rules` (/docs/providers/akamai/d/property_rules.html))
- `rule_format` — (Optional) The rule format to use (more (https://developer.akamai.com/api/core_features/property_manager/v1.html#getruleformats)).

In addition the specifying the rule tree in it's entirety, you can also set the default CP Code and Origin explicitly. *This will override your JSON configuration*.

- `cp_code` — (Required) The CP Code id or name to use (or create).
- `origin` — (Optional) The property origin (an origin must be specified to activate a property, but may be defined in your rules block).
  - `hostname` — (Required) The origin hostname.
  - `port` — (Optional) The origin port to connect to (default: 80).
  - `forward_hostname` — (Optional) The value for the Hostname header sent to origin. (default: `ORIGIN_HOSTNAME`).
  - `cache_key_hostname` — (Optional) The hostname uses for the cache key. (default: `ORIGIN_HOSTNAME`).
  - `compress` — (Optional, boolean) Whether origin supports gzip compression (default: `false`).
  - `enable_true_client_ip` — (Optional, boolean) Whether the X-True-Client-IP header should be sent to origin (default: `false`).

You can also define property manager variables. *This will override your JSON configuration*.

- `variables` — (Optional) A JSON encoded string of property manager variable definitions (see: `akamai_property_variables` (/docs/providers/akamai/r/property_variables.html))

Attribute Reference

The following attributes are returned:

- `account` — the Account ID under which the property is created.
- `version` — the current version of the property config.
- `production_version` — the current version of the property active on the production network.
- `staging_version` — the current version of the property active on the staging network.
- `edge_hostnames` — the final public hostname to edge hostname map

akamai_property_variables

The `akamai_property_variables` allows you to implement dynamic functionality. You can perform conditional logic based on the variable's value, and catch any unforeseen errors that execute on the edge at runtime.

Typical uses for variables include:

- Simplify configurations by reducing the number of rules and behaviors.
- Improve self serviceability by replacing or extending advanced metadata.
- Automate redirects, forward path rewrites, HTTP header and cookie manipulation.
- Move origin functionality to the edge.

Example Usage

Basic usage:

```
resource "akamai_property_variables" "origin" {
  variables {
     variable {
        name        = "PMUSER_ORIGIN"
        value       = "origin.example.org"
        description = "Origin Hostname"
        hidden      = true
        sensitive   = true
     }
  }
}
```

Argument Reference

The following arguments are supported:

The `variables` block may contain many `variable` blocks which support the following arguments:

- `name` — (Required) The name of the variable.
- `value` — (Required) The default value to assign to the variable
- `description` — (Optional) A human-readable description
- `hidden` — (Optional) Whether to hide the variable when debugging requests
- `sensitive` — (Optional) Whether to obscure the value when debugging requests