NS1 Provider

The NS1 provider exposes resources to interact with the NS1 REST API. The provider needs to be configured with the proper credentials before it can be used. Note also that for a given resource to function, the API key used must have the corresponding permissions set.

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

Example Usage

```
# Configure the NS1 provider
provider "ns1" {
  apikey = "${var.ns1_apikey}"
}

# Create a new zone
resource "ns1_zone" "foobar" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `apikey` - (Required) NS1 API token. It must be provided, but it can also be sourced from the `NS1_APIKEY` environment variable.
- `version` - (Optional, but recommended if you don't like surprises) From output of `terraform init`.

Environment Variables

The provider does check some environment variables:

- `NS1_APIKEY` - (string) Explained above.
- `NS1_ENDPOINT` - (string) For managed clients, this normally should not be set.
- `NS1_IGNORE_SSL` - (boolean) This normally does not need to be set. If set, follows the convention of strconv.ParseBool (https://golang.org/pkg/strconv/#ParseBool).

Data Source: ns1_dnssec

Provides DNSSEC details about a NS1 Zone.

Example Usage

```
# Get DNSSEC details about a NS1 Zone.
resource "ns1_zone" "example" {
  zone   = "terraform.example.io"
  dnssec = true
}

data "ns1_dnssec" "example" {
  zone = "${ns1_zone.example.zone}"
}
```

Argument Reference

- `zone` - (Required) The name of the zone to get DNSSEC details for.

Attributes Reference

In addition to the argument above, the following are exported:

- `keys` - (Computed) - Keys field is documented below.
- `delegation` - (Computed) - Delegation field is documented below.

Keys

`keys` has the following fields:

- `dnskey` - (Computed) List of Keys. Key is documented below.
- `ttl` - (Computed) TTL for the Keys (int).

Delegation

`delegation` has the following fields:

- `dnskey` - (Computed) List of Keys. Key is documented below.
- `ds` - (Computed) List of Keys. Key is documented below.
- `ttl` - (Computed) TTL for the Keys (int).

Key

A `key` has the following (string) fields:

- `flags` - (Computed) Flags for the key.
- `protocol` - (Computed) Protocol of the key.
- `algorithm` - (Computed) Algorithm of the key.
- `public_key` - (Computed) Public key for the key.

Data Source: ns1_zone

Provides details about a NS1 Zone. Use this if you would simply like to read information from NS1 into your configurations. For read/write operations, you should use a resource.

Example Usage

```
# Get details about a NS1 Zone.
data "ns1_zone" "example" {
  zone = "terraform.example.io"
}
```

Argument Reference

- `zone` - (Required) The domain name of the zone.

Attributes Reference

In addition to the argument above, the following are exported:

- `link` - The linked target zone.
- `primary` - The primary ip.
- `additional_primaries` - List of additional IPs for the primary zone.
- `ttl` - The SOA TTL.
- `refresh` - The SOA Refresh.
- `retry` - The SOA Retry.
- `expiry` - The SOA Expiry.
- `nx_ttl` - The SOA NX TTL.
- `dnssec` - Whether or not DNSSEC is enabled for the zone.
- `networks` - List of network IDs for which the zone is available.
- `dns_servers` - Authoritative Name Servers.
- `hostmaster` - The SOA Hostmaster.
- `secondaries` - List of secondary servers. Secondaries is documented below.

Secondaries

A secondary has the following fields:

- `ip` - IPv4 address of the secondary server.
- `port` - Port of the the secondary server. Default `53` .
- `notify` - Whether we send `NOTIFY` messages to the secondary host when the zone changes. Default `false` .
- `networks` - List of network IDs (`int`) for which the zone should be made available. Default is network 0, the primary NSONE Global Network.

ns1_apikey

Provides a NS1 Api Key resource. This can be used to create, modify, and delete api keys.

Example Usage

```
resource "ns1_team" "example" {
  name = "Example team"
}

resource "ns1_apikey" "example" {
  name  = "Example key"
  teams = ["${ns1_team.example.id}"]

  # Configure permissions
  dns_view_zones       = false
  account_manage_users = false
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free form name of the apikey.
- `key` - (Required) The apikeys authentication token.
- `teams` - (Required) The teams that the apikey belongs to.
- `dns_view_zones` - (Optional) Whether the apikey can view the accounts zones.
- `dns_manage_zones` - (Optional) Whether the apikey can modify the accounts zones.
- `dns_zones_allow_by_default` - (Optional) If true, enable the `dns_zones_allow` list, otherwise enable the `dns_zones_deny` list.
- `dns_zones_allow` - (Optional) List of zones that the apikey may access.
- `dns_zones_deny` - (Optional) List of zones that the apikey may not access.
- `data_push_to_datafeeds` - (Optional) Whether the apikey can publish to data feeds.
- `data_manage_datasources` - (Optional) Whether the apikey can modify data sources.
- `data_manage_datafeeds` - (Optional) Whether the apikey can modify data feeds.
- `account_manage_users` - (Optional) Whether the apikey can modify account users.
- `account_manage_payment_methods` - (Optional) Whether the apikey can modify account payment methods.
- `account_manage_plan` - (Optional) Whether the apikey can modify the account plan.

- `account_manage_teams` - (Optional) Whether the apikey can modify other teams in the account.
- `account_manage_apikeys` - (Optional) Whether the apikey can modify account apikeys.
- `account_manage_account_settings` - (Optional) Whether the apikey can modify account settings.
- `account_view_activity_log` - (Optional) Whether the apikey can view activity logs.
- `account_view_invoices` - (Optional) Whether the apikey can view invoices.
- `monitoring_manage_lists` - (Optional) Whether the apikey can modify notification lists.
- `monitoring_manage_jobs` - (Optional) Whether the apikey can modify monitoring jobs.
- `monitoring_view_jobs` - (Optional) Whether the apikey can view monitoring jobs.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_datafeed

Provides a NS1 Data Feed resource. This can be used to create, modify, and delete data feeds.

Example Usage

```
resource "ns1_datasource" "example" {
  name       = "example"
  sourcetype = "nsone_v1"
}

resource "ns1_datafeed" "uswest_feed" {
  name      = "uswest_feed"
  source_id = "${ns1_datasource.example.id}"

  config = {
    label = "uswest"
  }
}

resource "ns1_datafeed" "useast_feed" {
  name      = "useast_feed"
  source_id = "${ns1_datasource.example.id}"

  config = {
    label = "useast"
  }
}
```

Argument Reference

The following arguments are supported:

- `source_id` - (Required) The data source id that this feed is connected to.
- `name` - (Required) The free form name of the data feed.
- `config` - (Optional) The feeds configuration matching the specification in `feed_config` from /data/sourcetypes.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_datasource

Provides a NS1 Data Source resource. This can be used to create, modify, and delete data sources.

Example Usage

```
resource "ns1_datasource" "example" {
  name       = "example"
  sourcetype = "nsone_v1"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free form name of the data source.
- `sourcetype` - (Required) The data sources type, listed in API endpoint https://api.nsone.net/v1/data/sourcetypes (https://api.nsone.net/v1/data/sourcetypes).
- `config` - (Optional) The data source configuration, determined by its type, matching the specification in `config` from /data/sourcetypes.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_monitoringjob

Provides a NS1 Monitoring Job resource. This can be used to create, modify, and delete monitoring jobs.

Example Usage

```
resource "ns1_monitoringjob" "uswest_monitor" {
  name          = "uswest"
  active        = true
  regions       = ["sjc", "sin", "lga"]
  job_type      = "tcp"
  frequency     = 60
  rapid_recheck = true
  policy        = "quorum"

  config = {
    send = "HEAD / HTTP/1.0\r\n\r\n"
    port = 80
    host = "example-elb-uswest.aws.amazon.com"
  }

  rules {
    value      = "200 OK"
    comparison = "contains"
    key        = "output"
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free-form display name for the monitoring job.
- `job_type` - (Required) The type of monitoring job to be run. See NS1 API docs for supported values.
- `active` - (Required) Indicates if the job is active or temporarily disabled.
- `regions` - (Required) The list of region codes in which to run the monitoring job. See NS1 API docs for supported values.
- `frequency` - (Required) The frequency, in seconds, at which to run the monitoring job in each region.
- `rapid_recheck` - (Required) If true, on any apparent state change, the job is quickly re-run after one second to confirm the state change before notification.
- `policy` - (Required) The policy for determining the monitor's global status based on the status of the job in all regions. See NS1 API docs for supported values.
- `config` - (Required) A configuration dictionary with keys and values depending on the jobs' type.

- `notify_delay` - (Optional) The time in seconds after a failure to wait before sending a notification.
- `notify_repeat` - (Optional) The time in seconds between repeat notifications of a failed job.
- `notify_failback` - (Optional) If true, a notification is sent when a job returns to an "up" state.
- `notify_regional` - (Optional) If true, notifications are sent for any regional failure (and failback if desired), in addition to global state notifications.
- `notify_list` - (Optional) The id of the notification list to send notifications to.
- `notes` - (Optional) Freeform notes to be included in any notifications about this job.
- `rules` - (Optional) A list of rules for determining failure conditions. Job Rules are documented below.

Monitoring Job Rules (`rules`) support the following:

- `key` - (Required) The output key.
- `comparison` - (Required) The comparison to perform on the the output.
- `value` - (Required) The value to compare to.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_notifylist

Provides a NS1 Notify List resource. This can be used to create, modify, and delete notify lists.

Example Usage

```
resource "ns1_notifylist" "nl" {
  name = "my notify list"
  notifications {
    type = "webhook"
    config = {
      url = "http://www.mywebhook.com"
    }
  }

  notifications {
    type = "email"
    config = {
      email = "test@test.com"
    }
  }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free-form display name for the notify list.
- `notifications` - (Optional) A list of notifiers. All notifiers in a notification list will receive notifications whenever an event is send to the list (e.g., when a monitoring job fails). Notifiers are documented below.

Notify List Notifiers (`notifications`) support the following:

- `type` - (Required) The type of notifier. Available notifiers are indicated in /notifytypes endpoint.
- `config` - (Required) Configuration details for the given notifier type.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_record

Provides a NS1 Record resource. This can be used to create, modify, and delete records.

Example Usage

```
resource "ns1_zone" "example" {
  zone = "terraform.example.io"
}

resource "ns1_datasource" "ns1" {
  name       = "ns1_source"
  sourcetype = "nsone_v1"
}

resource "ns1_datafeed" "foo" {
  name      = "foo_feed"
  source_id = "${ns1_datasource.ns1.id}"
  config = {
    label = "foo"
  }
}

resource "ns1_datafeed" "bar" {
  name      = "bar_feed"
  source_id = "${ns1_datasource.ns1.id}"
  config = {
    label = "bar"
  }
}

resource "ns1_record" "www" {
  zone   = "${ns1_zone.tld.zone}"
  domain = "www.${ns1_zone.tld.zone}"
  type   = "CNAME"
  ttl    = 60
  meta   = {
    up   = true
  }

  regions {
    name = "east"
    meta = {
      georegion = "US-EAST"
    }
  }

  regions {
    name = "usa"
    meta = {
      country = "US"
    }
  }

  answers {
```

```
    answer  = "sub1.${ns1_zone.tld.zone}"
    region  = "east"
    meta    = {
      up    = "{\"feed\":\"${ns1_datafeed.foo.id}\"}"
    }
  }

  answers {
    answer = "sub2.${ns1_zone.tld.zone}"
    meta   = {
      up   = "{\"feed\":\"${ns1_datafeed.bar.id}\"}"
      connections = 3
    }
  }

  filters {
    filter = "select_first_n"

    config = {
      N = 1
    }
  }
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The zone the record belongs to.
- `domain` - (Required) The records' domain.
- `type` - (Required) The records' RR type.
- `ttl` - (Optional) The records' time to live.
- `link` - (Optional) The target record to link to. This means this record is a 'linked' record, and it inherits all properties from its target.
- `use_client_subnet` - (Optional) Whether to use EDNS client subnet data when available(in filter chain).
- `meta` - (Optional) meta is supported at the `record` level. Meta is documented below.
- `regions` - (Optional) One or more "regions" for the record. These are really just groupings based on metadata, and are called "Answer Groups" in the NS1 UI, but remain `regions` here for legacy reasons. Regions must be sorted alphanumerically by name, otherwise Terraform will detect changes to the record when none actually exist. Regions are documented below.
- `answers` - (Optional) One or more NS1 answers for the records' specified type. Answers are documented below.
- `filters` - (Optional) One or more NS1 filters for the record(order matters). Filters are documented below.

Answers

`answers` support the following:

- `answer` - (Required) Space delimited string of RDATA fields dependent on the record type.

  A:

  ```
  answer = "1.2.3.4"
  ```

  CNAME:

  ```
  answer = "www.example.com"
  ```

  MX:

  ```
  answer = "5 mail.example.com"
  ```

  SRV:

  ```
  answer = "10 0 2380 node-1.example.com"
  ```

  SPF:

  ```
  answer = "v=DKIM1; k=rsa; p=XXXXXXXX"
  ```

- `region` - (Optional) The region (Answer Group really) that this answer belongs to. This should be one of the names specified in `regions`. Only a single `region` per answer is currently supported. If you want an answer in multiple regions, duplicating the answer (including metadata) is the correct approach.
- `meta` - (Optional) meta is supported at the `answer` level. Meta is documented below.

Filters

`filters` support the following:

- `filter` - (Required) The type of filter.
- `disabled` - (Optional) Determines whether the filter is applied in the filter chain.
- `config` - (Optional) The filters' configuration. Simple key/value pairs determined by the filter type.

Regions

`regions` support the following:

- `name` - (Required) Name of the region (or Answer Group).
- `meta` - (Optional) meta is supported at the `regions` level. Meta is documented below. Note that `Meta` values for `country`, `ca_province`, `georegion`, and `us_state` should be comma separated strings, and changes in ordering

will not lead to terraform detecting a change.

Meta

Metadata (`meta`) is a bit tricky at the moment. For "static" values it works as you would expect, but when a value is a `datafeed` , it should be represented as "escaped" JSON. See the Example Usage above for illustration of this.

Note that variables are still supported in the escaped JSON format. Note also that we intend to fix up this "escaped" JSON stuff as soon as possible, so please bear with us and plan accordingly.

Since this resource supports import, you may find it helpful to set up some `meta` fields via the web portal or API, and use the results from import to ensure that everything is properly escaped and evaluated.

See NS1 API (https://ns1.com/api#get-available-metadata-fields) for the most up-to-date list of available `meta` fields.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

Import

```
terraform import ns1_record.<name> <zone>/<domain>/<type>
```

So for the example above:

```
terraform import ns1_record.www terraform.example.io/www.terraform.example.io/CNAME
```

ns1_team

Provides a NS1 Team resource. This can be used to create, modify, and delete teams. The credentials used must have the `manage_teams` permission set.

Example Usage

```
# Create a new NS1 Team
resource "ns1_team" "example" {
  name = "Example team"

  # Configure permissions
  dns_view_zones       = false
  account_manage_users = false
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free form name of the team.
- `dns_view_zones` - (Optional) Whether the team can view the accounts zones.
- `dns_manage_zones` - (Optional) Whether the team can modify the accounts zones.
- `dns_zones_allow_by_default` - (Optional) If true, enable the `dns_zones_allow` list, otherwise enable the `dns_zones_deny` list.
- `dns_zones_allow` - (Optional) List of zones that the team may access.
- `dns_zones_deny` - (Optional) List of zones that the team may not access.
- `data_push_to_datafeeds` - (Optional) Whether the team can publish to data feeds.
- `data_manage_datasources` - (Optional) Whether the team can modify data sources.
- `data_manage_datafeeds` - (Optional) Whether the team can modify data feeds.
- `account_manage_users` - (Optional) Whether the team can modify account users.
- `account_manage_payment_methods` - (Optional) Whether the team can modify account payment methods.
- `account_manage_plan` - (Optional) Whether the team can modify the account plan.
- `account_manage_teams` - (Optional) Whether the team can modify other teams in the account.
- `account_manage_apikeys` - (Optional) Whether the team can modify account apikeys.
- `account_manage_account_settings` - (Optional) Whether the team can modify account settings.
- `account_view_activity_log` - (Optional) Whether the team can view activity logs.

- `account_view_invoices` - (Optional) Whether the team can view invoices.
- `monitoring_manage_lists` - (Optional) Whether the team can modify notification lists.
- `monitoring_manage_jobs` - (Optional) Whether the team can modify monitoring jobs.
- `monitoring_view_jobs` - (Optional) Whether the team can view monitoring jobs.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_user

Provides a NS1 User resource. Creating a user sends an invitation email to the user's email address. This can be used to create, modify, and delete users. The credentials used must have the `manage_users` permission set.

Example Usage

```
resource "ns1_team" "example" {
  name = "Example team"

  dns_view_zones       = false
  account_manage_users = false
}

resource "ns1_user" "example" {
  name     = "Example User"
  username = "example_user"
  email    = "user@example.com"
  teams    = ["${ns1_team.example.id}"]
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The free form name of the user.
- `username` - (Required) The users login name.
- `email` - (Required) The email address of the user.
- `notify` - (Required) Whether or not to notify the user of specified events. Only `billing` is available currently.
- `teams` - (Required) The teams that the user belongs to.
- `dns_view_zones` - (Optional) Whether the user can view the accounts zones.
- `dns_manage_zones` - (Optional) Whether the user can modify the accounts zones.
- `dns_zones_allow_by_default` - (Optional) If true, enable the `dns_zones_allow` list, otherwise enable the `dns_zones_deny` list.
- `dns_zones_allow` - (Optional) List of zones that the user may access.
- `dns_zones_deny` - (Optional) List of zones that the user may not access.
- `data_push_to_datafeeds` - (Optional) Whether the user can publish to data feeds.
- `data_manage_datasources` - (Optional) Whether the user can modify data sources.
- `data_manage_datafeeds` - (Optional) Whether the user can modify data feeds.

- `account_manage_users` - (Optional) Whether the user can modify account users.
- `account_manage_payment_methods` - (Optional) Whether the user can modify account payment methods.
- `account_manage_plan` - (Optional) Whether the user can modify the account plan.
- `account_manage_teams` - (Optional) Whether the user can modify other teams in the account.
- `account_manage_apikeys` - (Optional) Whether the user can modify account apikeys.
- `account_manage_account_settings` - (Optional) Whether the user can modify account settings.
- `account_view_activity_log` - (Optional) Whether the user can view activity logs.
- `account_view_invoices` - (Optional) Whether the user can view invoices.
- `monitoring_manage_lists` - (Optional) Whether the user can modify notification lists.
- `monitoring_manage_jobs` - (Optional) Whether the user can modify monitoring jobs.
- `monitoring_view_jobs` - (Optional) Whether the user can view monitoring jobs.

Attributes Reference

All of the arguments listed above are exported as attributes, with no additions.

ns1_zone

Provides a NS1 DNS Zone resource. This can be used to create, modify, and delete zones.

Example Usage

```
# Create a new DNS zone
resource "ns1_zone" "example" {
  zone = "terraform.example.io"
  ttl  = 600
}

# Create a new primary zone
resource "ns1_zone" "example_primary" {
  zone     = "terraform-primary.example.io"
  secondaries {
    ip     = "2.2.2.2"
  }
  secondaries {
    ip     = "3.3.3.3"
    port   = 5353
    notify = true
  }
}

# Create a new secondary zone
resource "ns1_zone" "example_primary" {
  zone     = "terraform-primary.example.io"
  primary  = "2.2.2.2"
  additional_primaries = ["3.3.3.3", "4.4.4.4"]
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The domain name of the zone.
- `link` - (Optional) The target zone(domain name) to link to.
- `primary` - (Optional) The primary zones' IP. This makes the zone a secondary. Conflicts with `secondaries`.
- `additional_primaries` - (Optional) List of additional IPs for the primary zone. Conflicts with `secondaries`.
- `ttl` - (Optional/Computed) The SOA TTL.
- `refresh` - (Optional/Computed) The SOA Refresh. Conflicts with `primary` and `additional_primaries` (default must be accepted).
- `retry` - (Optional/Computed) The SOA Retry. Conflicts with `primary` and `additional_primaries` (default must be accepted).

- `expiry` - (Optional/Computed) The SOA Expiry. Conflicts with `primary` and `additional_primaries` (default must be accepted).
- `nx_ttl` - (Optional/Computed) The SOA NX TTL. Conflicts with `primary` and `additional_primaries` (default must be accepted).
- `dnssec` - (Optional/Computed) Whether or not DNSSEC is enabled for the zone. Note that DNSSEC must be enabled on the account by support for this to be set to `true`.
- `networks` - (Optional/Computed) List of network IDs for which the zone is available. If no network is provided, the zone will be created in network 0, the primary NS1 Global Network.
- `secondaries` - (Optional) List of secondary servers. This makes the zone a primary. Conflicts with `primary` and `additional_primaries`. Secondaries is documented below.

Secondaries

A zone can have zero or more `secondaries`. Note how this is implemented in the example above. A secondary has the following fields:

- `ip` - (Required) IPv4 address of the secondary server.
- `port` - (Optional) Port of the the secondary server. Default `53`.
- `notify` - (Optional) Whether we send `NOTIFY` messages to the secondary host when the zone changes. Default `false`.
- `networks` - (Computed) - List of network IDs (`int`) for which the zone should be made available. Default is network 0, the primary NSONE Global Network. Normally, you should not have to worry about this.

Attributes Reference

In addition to all arguments above, the following attributes are exported:

- `dns_servers` - (Computed) Authoritative Name Servers.
- `hostmaster` - (Computed) The SOA Hostmaster.

A note on making Primary or Secondary changes to zones

Switching a zone to being a secondary forces a new resource. In other words, the zone will first be destroyed, then recreated as a secondary. Editing or removing the `primary` key, or directly changing a secondary zone to a primary (by removing the `primary` and `additional_primaries` keys, and setting `secondaries`) is supported "in place". However, in these situations we do not alter records on the zone. You may need to amend records, or finagle them into Terraform state. As a particular example, if you change a secondary zone to be primary (or just not-a-secondary) before a zone transfer has occurred, you can end up with no records on the zone.

Currently, this provider does not support zones being both Primary and Secondary. If that functionality is important for your workflow, please open an issue or contact support, so we can prioritize the work accordingly.

Import

```
terraform import ns1_zone.<name> <zone>
```

So for the example above:

```
terraform import ns1_zone.example terraform.example.io
```