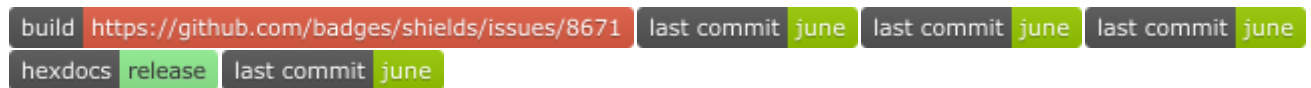


---

## ExAws



A flexible easy to use set of AWS APIs.

Available Services: <https://github.com/ex-aws?q=service&type=&language=>

## Getting Started

ExAws v2.0 breaks out every service into its own package. To use the S3 service, you need both the core `:ex_aws` package as well as the `:ex_aws_s3` package.

As with all ExAws services, you'll need a compatible HTTP client (defaults to `:hackney`) and whatever JSON or XML codecs needed by the services you want to use. Consult individual service documentation for details on what each service needs.

```
1 defp deps do
2   [
3     {:ex_aws, "~> 2.1"},
4     {:ex_aws_s3, "~> 2.0"},
5     {:hackney, "~> 1.9"},
6     {:sweet_xml, "~> 0.6"},
7   ]
8 end
```

With these deps you can use ExAws precisely as you're used to:

```
1 # make a request (with the default region)
2 ExAws.S3.list_objects("my-bucket") |> ExAws.request()
3
4 # or specify the region
5 ExAws.S3.list_objects("my-bucket") |> ExAws.request(region: "us-west-1"
6   )
7
8 # some operations support streaming
9 ExAws.S3.list_objects("my-bucket") |> ExAws.stream!() |> Enum.to_list()
```

## AWS Key configuration

ExAws requires valid AWS keys in order to work properly. ExAws by default does the equivalent of:

```
1 config :ex_aws,
2   access_key_id: [[:system, "AWS_ACCESS_KEY_ID"], :instance_role],
```

---

```
3   secret_access_key: [{:system, "AWS_SECRET_ACCESS_KEY"}, :  
    instance_role]
```

This means it will try to resolve credentials in order:

- Look for the AWS standard `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables
- Resolve credentials with IAM
  - If running inside ECS and a task role has been assigned it will use it
  - Otherwise it will fall back to the instance role

AWS CLI config files are supported, but require an additional dependency:

```
1  {:configparser_ex, "~> 4.0"}
```

You can then add `{:awscli, "profile_name", timeout}` to the above config and it will pull information from `~/.aws/config` and `~/.aws/credentials`

Alternatively, if you already have a profile name set in the `AWS_PROFILE` environment variable, you can use that with `{:awscli, :system, timeout}`

```
1  config :ex_aws,  
2    access_key_id: [{:system, "AWS_ACCESS_KEY_ID"}, {:awscli, "default",  
    30}, :instance_role],  
3    secret_access_key: [{:system, "AWS_SECRET_ACCESS_KEY"}, {:awscli, "  
    default", 30}, :instance_role]
```

For role based authentication via `role_arn` and `source_profile` an additional dependency is required:

```
1  {:ex_aws_sts, "~> 2.0"}
```

Further information on role based authentication is provided in said dependency.

**Session token configuration** Alternatively, you can also provide `AWS_SESSION_TOKEN` to `security_token` to authenticate with session token:

```
1  config :ex_aws,  
2    access_key_id: {:system, "AWS_ACCESS_KEY_ID"},  
3    security_token: {:system, "AWS_SESSION_TOKEN"},  
4    secret_access_key: {:system, "AWS_SECRET_ACCESS_KEY"}
```

---

## Hackney configuration

ExAws by default uses hackney to make HTTP requests to AWS API. You can modify the options as such:

```
1 config :ex_aws, :hackney_opts,  
2   follow_redirect: true,  
3   recv_timeout: 30_000
```

## AWS Region Configuration.

You can set the region used by default for requests.

```
1 config :ex_aws,  
2   region: "us-west-2",
```

Alternatively, the region can be set in an environment variable:

```
1 config :ex_aws,  
2   region: {:system, "AWS_REGION"}
```

## JSON Codec Configuration

The default JSON codec is Jason. You can choose a different one:

```
1 config :ex_aws,  
2   json_codec: Poison
```

## Path Normalization

Paths that include multiple consecutive /'s will by default be normalized to a single slash. There are cases when paths need to be literal (S3) and this normalization behaviour can be turned off via configuration:

```
1 config :ex_aws,  
2   normalize_path: false
```

## Direct Usage

ExAws can also be used directly without any specific service module.

You need to figure out how the API of the specific AWS service works, in particular:

- 
- Protocol (JSON or query).
  - Path (depends on the service and the specific operation, usually “/”).
  - Service name (used to generate the request signature, as described here).
  - Request body, query params, HTTP method, and headers (depends on the service and specific operation).

You can look for this information in the service’s API reference at [docs.aws.amazon.com](https://docs.aws.amazon.com) or, for example, in the Go SDK API models at [github.com/aws/aws-sdk-go](https://github.com/aws/aws-sdk-go) (look for a `api-*.json` file).

The protocol dictates which operation module to use for the request. If the protocol is JSON, use `ExAws.Operation.JSON`, if it’s query, use `ExAws.Operation.Query`.

## Examples

### Redshift DescribeClusters

```
1 action = :describe_clusters
2 action_string = action |> Atom.to_string |> Macro.camelize
3
4 operation =
5   %ExAws.Operation.Query{
6     path: "/",
7     params: %{"Action" => action_string},
8     service: :redshift,
9     action: action
10  }
11
12 ExAws.request(operation)
```

### ECS RunTask

```
1 data = %{
2   taskDefinition: "hello_world",
3   launchType: "FARGATE",
4   networkConfiguration: %{
5     awsvpcConfiguration: %{
6       subnets: ["subnet-1a2b3c4d", "subnet-4d3c2b1a"],
7       securityGroups: ["sg-1a2b3c4d"],
8       assignPublicIp: "ENABLED"
9     }
10  }
11 }
12
13 operation =
14   %ExAws.Operation.JSON{
15     http_method: :post,
16     headers: [
17       {"x-amz-target", "AmazonEC2ContainerServiceV20141113.RunTask"},
18       {"content-type", "application/x-amz-json-1.1"}
19     ],
```

---

```
20     path: "/",
21     data: data,
22     service: :ecs
23 }
24
25 ExAws.request(operation)
```

## Highlighted Features

- Easy configuration.
- Minimal dependencies. Choose your favorite JSON codec and HTTP client.
- Elixir streams to automatically retrieve paginated resources.
- Elixir protocols allow easy customization of Dynamo encoding / decoding.
- Simple. ExAws aims to provide a clear and consistent elixir wrapping around AWS APIs, not abstract them away entirely. For every action in a given AWS API there is a corresponding function within the appropriate module. Higher level abstractions like the aforementioned streams are in addition to and not instead of basic API calls.

That's it!

## Retries

ExAws will retry failed AWS API requests using exponential backoff per the “Full Jitter” formula described in <https://www.awsarchitectureblog.com/2015/03/backoff.html>

The algorithm uses three values, which are configurable:

```
1 # default values shown below
2
3 config :ex_aws, :retries,
4   max_attempts: 10,
5   base_backoff_in_ms: 10,
6   max_backoff_in_ms: 10_000
```

- `max_attempts` is the maximum number of possible attempts with backoffs in between each one
- `base_backoff_in_ms` corresponds to the `base` value described in the blog post
- `max_backoff_in_ms` corresponds to the `cap` value described in the blog post

---

## Testing

If you want to run `mix test`, you'll need to have a local `dynamodb` running on port 8000. See [Setting up DynamoDB Local](#).

The redirect test will intentionally cause a warning to be issued.

## License

The MIT License (MIT)

Copyright (c) 2014-2020 CargoSense, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.