Go App Engine packages

This repository supports the Go runtime on *App Engine standard*. It provides APIs for interacting with App Engine services. Its canonical import path is `google.golang.org/appengine`.

See https://cloud.google.com/appengine/docs/go/ for more information.

File issue reports and feature requests on the GitHub's issue tracker.

Upgrading an App Engine app to the flexible environment

This package does not work on *App Engine flexible*.

There are many differences between the App Engine standard environment and the flexible environment.

See the documentation on upgrading to the flexible environment.

Directory structure

The top level directory of this repository is the `appengine` package. It contains the basic APIs (e.g. `appengine.NewContext`) that apply across APIs. Specific API packages are in subdirectories (e.g. `datastore`).

There is an `internal` subdirectory that contains service protocol buffers, plus packages required for connectivity to make API calls. App Engine apps should not directly import any package under `internal`.

Updating from legacy (`import "appengine"`) packages

If you're currently using the bare `appengine` packages (that is, not these ones, imported via `google.golang.org/appengine`), then you can use the `aefix` tool to help automate an upgrade to these packages.

Run `go get google.golang.org/appengine/cmd/aefix` to install it.

1. Update import paths

The import paths for App Engine packages are now fully qualified, based at `google.golang.org/appengine`. You will need to update your code to use import paths starting with that; for instance, code importing `appengine/datastore` will now need to import `google.golang.org/appengine/datastore`.

2. Update code using deprecated, removed or modified APIs

Most App Engine services are available with exactly the same API. A few APIs were cleaned up, and there are some differences:

- `appengine.Context` has been replaced with the `Context` type from `golang.org/x/net/context`.
- Logging methods that were on `appengine.Context` are now functions in `google.golang.org/appengine/log`.
- `appengine.Timeout` has been removed. Use `context.WithTimeout` instead.

- `appengine.Datacenter` now takes a `context.Context` argument.
- `datastore.PropertyLoadSaver` has been simplified to use slices in place of channels.
- `delay.Call` now returns an error.
- `search.FieldLoadSaver` now handles document metadata.
- `urlfetch.Transport` no longer has a Deadline field; set a deadline on the `context.Context` instead.
- `aetest` no longer declares its own Context type, and uses the standard one instead.
- `taskqueue.QueueStats` no longer takes a maxTasks argument. That argument has been deprecated and unused for a long time.
- `appengine.BackendHostname` and `appengine.BackendInstance` were for the deprecated backends feature. Use `appengine.ModuleHostname` and `appengine.ModuleName` instead.
- Most of `appengine/file` and parts of `appengine/blobstore` are deprecated. Use Google Cloud Storage if the feature you require is not present in the new blobstore package.
- `appengine/socket` is not required on App Engine flexible environment / Managed VMs. Use the standard `net` package instead.

Key Encode/Decode compatibiltiy to help with datastore library migrations

Key compatibility updates have been added to help customers transition from google.golang.org/appengine/datastore to cloud.google.com/go/datastore. The `EnableKeyConversion` enables automatic conversion from a key encoded with cloud.google.com/go/datastore to google.golang.org/appengine/datastore key type.

Enabling key conversion

Enable key conversion by calling `EnableKeyConversion(ctx)` in the `/_ah/start` handler for basic and manual scaling or any handler in automatic scaling.

1. Basic or manual scaling

This start handler will enable key conversion for all handlers in the service.

```
http.HandleFunc("/_ah/start", func(w http.ResponseWriter, r *http.Request) {
    datastore.EnableKeyConversion(appengine.NewContext(r))
})
```

2. Automatic scaling

`/_ah/start` is not supported for automatic scaling and `/_ah/warmup` is not guaranteed to run, so you must call `datastore.EnableKeyConversion(appengine.NewContext(r))` before you use code that needs key conversion.

You may want to add this to each of your handlers, or introduce middleware where it's called. `EnableKeyConversion` is safe for concurrent use. Any call to it after the first is ignored.