cron.xml Reference for App Engine SDK-Based Tooling

Note: This page provides a reference for the XML format of the Cron configuration file that is used by tooling based on the App Engine SDK for Java; for example the `appcfg` tool. If you are using Cloud SDK-based tooling, such as the `gcloud` command line, or Cloud SDK-based Maven (https://cloud.google.com/appengine/docs/standard/java/tools/using-maven), Gradle (https://cloud.google.com/appengine/docs/standard/java/tools/gradle), Eclipse (https://cloud.google.com/eclipse/docs/), or IntelliJ (https://cloud.google.com/tools/intellij/docs/) plugins, refer to the Cron reference in YAML format (https://cloud.google.com/appengine/docs/standard/java/config/cronref-yaml).

The `cron.xml` defines scheduled tasks for your application. This topic provides details on the options available to you. See Scheduling Tasks with Cron (https://cloud.google.com/appengine/docs/standard/java/config/cron) for how-to information related to cron jobs.

Example

The following is an example `cron.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<cronentries>
  <cron>
    <url>/recache</url>
    <description>Repopulate the cache every 2 minutes</description>
    <schedule>every 2 minutes</schedule>
  </cron>
  <cron>
```

```
    <url>/weeklyreport</url>
    <description>Mail out a weekly report</description>
    <schedule>every monday 08:30</schedule>
    <timezone>America/New_York</timezone>
  </cron>
  <cron>
    <url>/weeklyreport</url>
    <description>Mail out a weekly report</description>
    <schedule>every monday 08:30</schedule>
    <timezone>America/New_York</timezone>
    <target>version-2</target>
  </cron>
</cronentries>
```

For an XSD describing the format, check the file **`docs/cron.xsd`** in the SDK.

Syntax

The **`cron.xml`** file should reside in the **`WEB-INF`** directory of your application alongside **`appengine-web.xml`** : **`cron.xml`** configures scheduled tasks for your Java application.

Cron job definitions

Element	Description
`<description>`	Optional. The description is visible in the GCP Console and the development server's admin interface.
`<retry-parameters>`	Optional. If a cron job's request handler returns a HTTP status code that is not in the range 200–299 (inclusive) App Engine considers the job to have failed. By default, failed jobs are not retried. You can cause failed jobs to be retried by including a retry-parameters block in your configuration file. See the Cron retries (#retry) section for more information.
`<schedule>`	Required. Cron schedules are specified using a simple English-like format, such as **`every 5 minutes from 10:00 to 14:00`**. See schedule format (#schedule_format) for more information.

Element	Description
`<target>`	The **`target`** string is prepended to your app's hostname. It is usually the name of a service. The cron job will be routed to the version of the named service that is configured for traffic. **Warning:** Be careful if you run a cron job with traffic splitting (https://cloud.google.com/appengine/docs/java/splitting-traffic) enabled. The request from the cron job is always sent from the same IP address, so if you've specified IP address splitting, the logic will route the request to the same version every time. If you've specified cookie splitting, the request will not be split at all, because there is no cookie accompanying the request. If there is no service with the name assigned to **`target`**, the name is assumed to be a version of the default service, and App Engine will attempt to route the job to that version. For more information on specifying the version, see the appengine-web.xml reference (https://cloud.google.com/appengine/docs/standard/java/config/appref#version).
`<timezone>`	The **`timezone`** should be the name of a standard zoneinfo (https://wikipedia.org/wiki/List_of_zoneinfo_time_zones) time zone name. If you don't specify a timezone, jobs run in UTC (also known as GMT).
`<url>`	Required. The **`<url>`** field is just a URL in your application. If the **`<url>`** element contains the special XML characters **&**, **<**, **>**, **'**, or **"**, you should escape them.

Schedule format

Cron schedules are specified using a simple English-like format.

The following are examples of schedules:

```
every 12 hours
every 5 minutes from 10:00 to 14:00
every day 00:00
every monday 09:00
2nd,third mon,wed,thu of march 17:00
1st monday of sep,oct,nov 17:00
1 of jan,april,july,oct 00:00
```

You can use either the full weekday name or the short form (sun, mon, tue, wed, thu, fri, sat). If you don't need to run a recurring job at a specific time, but instead only need to run it at regular intervals, use the form:

```
every N (hours|mins|minutes) ["from" (time) "to" (time)]
```

The brackets are for illustration only, and quotes indicate a literal.

- *N* specifies a number.
- *hours* or *minutes* (you can also use *mins*) specifies the unit of time.
- *time* specifies a time of day, as HH:MM in 24 hour time.

By default, an interval schedule starts the next interval after the last job has completed. If a *from...to* clause is specified, however, the jobs are scheduled at regular intervals independent of when the last job completed. For example:

```
every 2 hours from 10:00 to 14:00
```

This schedule runs the job three times per day at 10:00, 12:00, and 14:00, regardless of how long it takes to complete.

You can use the literal `synchronized` to specify simple intervals, for example:

```
every 2 hours synchronized
```

Important: You must use an interval that evenly divides 24 hours. If you use an interval for `synchronized` that doesn't evenly divide 24 hours, it is invalid and you'll get an error.

If you want more specific timing than is provided for by `synchronized`, you can specify the schedule as:

```
("every"|ordinal) (days) ["of" (monthspec)] (time)
```

Where:

- *ordinal* specifies a comma separated list of "1st", "first" and so forth (both forms are ok)
- *days* specifies a comma separated list of days of the week (for example, "mon", "tuesday", with both short and long forms being accepted); "every day" is equivalent to "every mon,tue,wed,thu,fri,sat,sun"

- *monthspec* specifies a comma separated list of month names (for example, "jan", "march", "sep"). If omitted, implies every month. You can also say "month" to mean every month, as in "1,8,15,22 of month 09:00".
- *time* specifies the time of day, as HH:MM in 24 hour time.

Note: You cannot specify a schedule that includes both regular intervals and specific timing.

Cron requests

Request headers

Requests from the Cron Service will contain a HTTP header:

```
X-Appengine-Cron: true
```

The `X-Appengine-Cron` header is set internally by Google App Engine. If your request handler finds this header it can trust that the request is a cron request. If the header is present in an external user request to your app, it is stripped. The exception being requests from logged in administrators of the application, who are allowed to set the header for testing purposes.

Originating IP address

Google App Engine issues Cron requests from the IP address 0.1.0.1.

Cron retries

If a cron job's request handler returns a status code that is not in the range 200–299 (inclusive) App Engine considers the job to have failed. By default, failed jobs are not retried. You can cause failed jobs to be retried by including a `<retry-parameters>` block in your configuration file.

Here is a sample `cron.xml` file that contains a single cron job configured to retry up to five times (the default) with a starting backoff of 2.5 seconds that doubles each time.

```
<cronentries>
  <cron>
    <url>/retry</url>
    <description>Retry on jsdk</description>
    <schedule>every 10 minutes</schedule>
    <retry-parameters>
      <min-backoff-seconds>2.5</min-backoff-seconds>
      <max-doublings>5</max-doublings>
    </retry-parameters>
  </cron>
</cronentries>
```

Cron retries syntax

The retry parameters are described in the table below.

Element	Description
`<job-retry-limit>`	The maximum number of retry attempts for a failed cron job not to exceed '5'. If specified with `` `<job-age-limit>` ``, App Engine retries the cron job until both limits are reached. When omitted from the parameters, the limit is set to '5' by default.
`<job-age-limit>`	The time limit for retrying a failed cron job, measured from when the cron job was first run. The value is a number followed by a unit of time, where the unit is s for seconds, m for minutes, h for hours, or d for days. For example, the value 5d specifies a limit of five days after the cron job's first execution attempt. If specified with `<job-retry-limit>`, App Engine retries the cron job until both limits are reached.
`<min-backoff-seconds>`	The minimum number of seconds to wait before retrying a cron job after it fails.
`<max-backoff-seconds>`	The maximum number of seconds to wait before retrying a cron job after it fails.
`<max-doublings>`	The maximum number of times that the interval between failed cron job retries will be doubled before the increase becomes constant. The constant is: 2**(`max_doublings` - 1) * `min_backoff`.

Cron and app versions

If the `target` parameter has been set for a job, the request is sent to the specified version. Otherwise, Cron requests are sent to the version of your app that is configured to receive traffic.

Cron support in the GCP Console

The GCP Console Task queues page (https://console.cloud.google.com/appengine/taskqueues) has a tab that shows the tasks that are running cron jobs.

You can also visit the Logs page (https://cloud.google.com/appengine/docs/standard/java/logs/) see when cron jobs were added or removed.

Cron support in the development server

The development server doesn't automatically run your cron jobs. You can use your local desktop's cron or scheduled tasks interface to trigger the URLs of your jobs with curl (http://curl.haxx.se/) or a similar tool.

Deadlines

The cron timeout deadline depends on the instance class and scaling type that is configured for your app:

Automatic scaling

Timeout is about 10 minutes.

Basic scaling and manual scaling

Timeout can be up to 24 hours.

For more information, see Scaling types and instance classes (https://cloud.google.com/appengine/docs/standard/java/an-overview-of-app-engine#Java_Instance_scaling_and_class)

Limits

Free applications can have up to 20 scheduled tasks. Paid applications can have up to 250 scheduled tasks.

Deploying cron jobs

You can use the `appcfg.sh` tool to deploy your cron jobs to App Engine.

Option 1: Upload your whole application

To upload your whole application, which also updates the Cron Service with the entries from your `cron.xml` file, run the following command:

```
./appengine-java-sdk/bin/appcfg.sh -A your-app-id -V app-version updat
```

Option 2: Upload only your cron updates

To update just the cron configuration without uploading the rest of the application, run the following command:

```
./appengine-java-sdk/bin/appcfg.sh -A your-app-id -V app-version updat
```

Deleting all cron jobs

To delete all cron jobs:

1. Edit the contents of the `cron.xml` file to:

```
<?xml version="1.0" encoding="UTF-8"?>
<cronentries/>
```

2. Deploy the `cron.xml` file to App Engine.

Last updated February 8, 2018.