+++ title = "Settings updates at runtime" description = "Settings updates at runtime" keywords = ["grafana", "runtime", "settings"] weight = 500 +++

Settings updates at runtime

> ***Note:*** *Available in Grafana Enterprise v8.0+.*

Settings updates at runtime allows you to update Grafana settings with no need to restart the Grafana server.

Updates that happen at runtime are stored in the database and override settings from the other sources (arguments, environment variables, settings file, etc). Therefore, every time a specific setting key is removed at runtime, the value used for that key is the inherited one from the other sources in the reverse order of precedence (`arguments > environment variables > settings file`), being the application default the value used when no one provided through one of these, at least.

Currently, **it only supports updates on the `auth.saml` section.**

Update settings via the API

You can update settings through the [Admin API]({{< relref "../http_api/admin.md#update-settings" >}}).

When you submit a settings update via API, Grafana verifies if the given settings updates are allowed and valid. If they are, then Grafana stores the settings in the database and reloads Grafana services with no need to restart the instance.

So, the payload of a `PUT` request to the update settings endpoint (`/api/admin/settings`) should contain (either one or both):

- An `updates` map with a key, and a value per section you want to set.
- A `removals` list with keys per section you want to unset.

For example, if you provide the following `updates` :

```
{
  "updates": {
    "auth.saml": {
      "enabled": "true",
      "single_logout": "false"
    }
  }
}
```

it would enable SAML and disable single logouts. And, if you provide the following `removals` :

```
{
  "auth.saml": ["allow_idp_initiated"]
}
```

it would remove the key/value setting identified by `allow_idp_initiated` within the `auth.saml` . So, the SAML service would be reloaded and that value would be inherited for either (settings `.ini` file, environment variable, command line arguments or any other accepted mechanism to provide configuration).

Therefore, the complete HTTP payload would looks like:

```
{
  "updates": {
    "auth.saml": {
      "enabled": "true",
      "single_logout": "false"
    }
  },
  "removals": {
    "auth.saml": ["allow_idp_initiated"]
  }
}
```

In case any of these settings cannot be overridden nor valid, it would return an error and these settings won't be persisted into the database.

Background job (high availability set-ups)

Grafana Enterprise has a built-in scheduled background job that looks into the database every minute for settings updates. If there are updates, it reloads the Grafana services affected by the detected changes.

The background job synchronizes settings between instances in high availability set-ups. So, after you perform some changes through the HTTP API, then the other instances are synchronized through the database and the background job.

Control access with fine-grained access control

If you have [Fine-grained access Control]({{< relref "../enterprise/access-control/_index.md" >}}) enabled, you can control who can read or update settings. Refer to the [Admin API]({{< relref "../http_api/admin.md#update-settings" >}}) for more information.