Settings updates at runtime

- Update settings via the API
- Check for stored settings
- Remove stored settings
- Background job (high availability set-ups)
- Control access with role-based access control

{{< admonition type="note" >}} This functionality is deprecated and will be removed in a future release. For configuring SAML authentication, please use the new SSO settings API. {{< /admonition >}}

By updating settings at runtime, you can update Grafana settings without needing to restart the Grafana server.

Updates that happen at runtime are stored in the database and override settings from 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`). When no value is provided through any of these options, then the value used will be the application default

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

{{< admonition type="warning" >}} Stored settings override the configuration file, environment variables, and command line arguments, and nothing in the UI or the file says so. If you rotate a credential by editing the file while a stored value exists, Grafana keeps using the stored value and your change has no effect.

Refer to Check for stored settings when a configuration change appears to do nothing. {{< /admonition >}}

Update settings via the API

You can update settings through the Admin API.

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`:

```
{
  "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.

Check for stored settings

Two stores can override your configuration file. Grafana resolves each key in this order: an SSO settings record, then the settings table this API writes, then command line arguments, environment variables, the configuration file, and defaults.

An SSO settings record comes from the SSO Settings API, the SAML and OAuth user interfaces, or Terraform. To check for one:

```
curl -s -u "<USERNAME>:<PASSWORD>" https://<GRAFANA_URL>/api/v1/sso-settings/saml
```

Replace `<USERNAME>` and `<PASSWORD>` with the credentials of a user that has the `settings:read` permission, and `<GRAFANA_URL>` with your Grafana address.

A `"source": "database"` response means a record exists and wins over everything below it. A `"system"` response rules out a record, but not a value in the settings table. To inspect that table key by key, request `GET /api/admin/settings-verbose`, which marks each key `db` or `system`. That endpoint requires Grafana Enterprise.

While an SSO settings record exists, Grafana also ignores lower-precedence values for any key containing `certificate`, `private_key`, or `idp_metadata`, even when the record doesn't set them.

Remove stored settings

For SAML, one request clears both stores:

```
curl -X DELETE -u "<USERNAME>:<PASSWORD>" https://<GRAFANA_URL>/api/v1/sso-settings/saml
```

This clears the SSO settings record and every `auth.saml` key in the settings table, so your configuration file takes over again. It returns `404` and clears nothing when no record exists.

To remove single keys from the settings table instead, send a `removals` list:

```
curl -X PUT -u "<USERNAME>:<PASSWORD>" \
  -H "Content-Type: application/json" \
  -d '{"removals": {"auth.saml": ["private_key"]}}' \
  https://<GRAFANA_URL>/api/admin/settings
```

Don't try to remove a key by leaving it out of a `PUT /api/v1/sso-settings/{provider}` payload. That request replaces the whole record, so anything you omit is deleted.

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 a highly available set-up. 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 role-based access control

If you have role-based access control enabled, you can control who can read or update settings.

Refer to the Admin API for more information.

□□□ #2
□ □□□ □□□ 2026-10-31 16:14:38 UTC
□ □□□ □□□ 2026-10-31 16:14:42 UTC