Data requests

BackendSrv handles all outgoing HTTP requests from Grafana. This document explains the high-level concepts used by `BackendSrv`.

Canceling requests

This section describes how canceling requests work in Grafana. While data sources can implement their own cancellation concept, we recommend that you use the method we describe here.

A data request can take a long time to finish. During the time between when a request starts and finishes, the user can change context. For example, the user may navigate away or issue the same request again.

If we wait for canceled requests to complete, it might create unnecessary load on data sources.

Grafana uses a concept called *request cancelation* to cancel any ongoing request that Grafana doesn't need.

Before Grafana 7.2

Before Grafana can cancel any data request, it has to identify that request. Grafana identifies a request using the property `requestId` passed as options when you use BackendSrv.

The cancellation logic is as follows:

- When an ongoing request discovers that an additional request with the same `requestId` has started, then Grafana will cancel the ongoing request.
- When an ongoing request discovers that the special "cancel all requests" `requestId` was sent, then Grafana will cancel the ongoing request.

After Grafana 7.2

Grafana 7.2 introduced an additional way of canceling requests using RxJs. To support the new cancellation functionality, the data source needs to use the new `fetch` function in BackendSrv.

Migrating the core data sources to the new `fetch` function is an ongoing process that you can read about in this issue.

Request queue

Depending on how the web browser implements the protocol for HTTP 1.1, it will limit the number of parallel requests, lets call this limit *max_parallel_browser_request*.

Unless you have configured Grafana to use HTTP2, the browser limits parallel data requests according to the browser's implementation. For more information on how to enable HTTP2, refer to Configuration.

Because there is a *max_parallel_browser_request* limit, if some of the requests take a long time, they will block later requests and make interacting with Grafana very slow.

Before Grafana 7.2

Not supported.

After Grafana 7.2

Grafana uses a *request queue* to process all incoming data requests in order while reserving a free "spot" for any requests to the Grafana API.

Since the first implementation of the request queue doesn't take into account what browser the user uses, the *request queue* limit for parallel data source requests is hard-coded to 5.

> ***Note:*** *Grafana instances configured with HTTP2 will have a hard coded limit of 1000.*