+++ title = "Legacy snapshot mode" aliases = ["/docs/grafana/latest/plugins/developing/snapshot-mode/"] +++

Legacy snapshot mode

{{< figure class="float-right" src="/static/img/docs/Grafana-snapshot-example.png" caption="A dashboard using snapshot data and not live data." >}}

Grafana has this great feature where you can [save a snapshot of your dashboard]({{< relref "../../../dashboards/json-model.md" >}}). Instead of sending a screenshot of a dashboard to someone, you can send them a working, interactive Grafana dashboard with the snapshot data embedded inside it. The snapshot can be saved on your Grafana server and is available to all your co-workers. Raintank also hosts a snapshot server if you want to send the snapshot to someone who does not have access to your Grafana server.

{{< figure class="float-right" src="/static/img/docs/animated_gifs/snapshots.gif" caption="Selecting a snapshot" >}}

This all works because Grafana saves a snapshot of the current data in the dashboard json instead of fetching the data from a data source. However, if you are building a custom panel plugin then this will not work straight out of the box. You will need to make some small (and easy!) changes first.

Enabling support for loading snapshot data

Grafana automatically saves data from data sources in the dashboard json when the snapshot is created so we do not have to write any code for that. Enabling snapshot support for reading time series data is very simple. First in the constructor, we need to add an event handler for `data-snapshot-load`. This event is triggered by Grafana when the snapshot data is loaded from the dashboard json.

```
constructor($scope, $injector, contextSrv) {
    super($scope, $injector);
    ...
    this.events.on('init-edit-mode', this.onInitEditMode.bind(this));
    this.events.on('data-received', this.onDataReceived.bind(this));
    this.events.on('panel-teardown', this.onPanelTeardown.bind(this));
    this.events.on('data-snapshot-load', this.onDataSnapshotLoad.bind(this));
```

Then we need to create a simple event handler that just forwards the data on to our regular `data-received` handler:

```
onDataSnapshotLoad(snapshotData) {
  this.onDataReceived(snapshotData);
}
```

This will cover most use cases for snapshot support. Sometimes you will want to save data that is not time series data from a Grafana data source and then you have to do a bit more work to get snapshot support.

Saving custom data for snapshots

Data that is not time series data from a Grafana data source is not saved automatically by Grafana. Saving custom data for snapshot mode has to be done manually.

{{< figure class="float-right" src="/static/img/docs/Grafana-save-snapshot.png" caption="Save snapshot" >}}

Grafana gives us a chance to save data to the dashboard json when it is creating a snapshot. In the 'data-received' event handler, you can check the snapshot flag on the dashboard object. If this is true, then Grafana is creating a snapshot and you can manually save custom data to the panel json. In the example, a new field called snapshotLocationData in the panel json is initialized with a snapshot of the custom data.

```
onDataReceived(dataList) {
  if (!dataList) return;

  if (this.dashboard.snapshot && this.locations) {
    this.panel.snapshotLocationData = this.locations;
  }
```

Now the location data is saved in the dashboard json but we will have to load it manually as well.

Loading custom data for snapshots

The example below shows a function that loads the custom data. The data source for the custom data (an external API in this case) is not available in snapshot mode so a guard check is made to see if there is any snapshot data available first. If there is, then the snapshot data is used instead of trying to load the data from the external API.

```
loadLocationDataFromFile(reload) {
  if (this.map && !reload) return;

  if (this.panel.snapshotLocationData) {
    this.locations = this.panel.snapshotLocationData;
    return;
  }
```

It is really easy to forget to add this support but it enables a great feature and can be used to demo your panel.

If there is a panel plugin that you would like to be installed on the Raintank Snapshot server then please contact us via Slack or GitHub.