The limits workflows

Dynatrace has many limits defined in settings objects used by the Settings 2.0 framework.

This repository contains three workflows. They fetch all the available settings objects within a Dynatrace environment.

The **limits_calc_only** workflow collects all the limits and displays the result in JSON format. You will see the result after the workflow is executed when you click on the `ingest` action.

The **limits_calc_and_ingest** and **limits_calc_and_ingest public** workflows ingest the data into Dynatrace as business events. Then, you can query the ingested data using DQL.

Please be careful when running workflows in customer environments! Two workflows, **limits_calc_and_ingest** and **limits_calc_only**, collect both limits that are visible to the customer and some that are only available internally. Therefore, the workflow execution by the customer might fail because it lacks access to the internal settings objects. The third workflow (**limits_calc_and_ingest_public**) collects only settings that are accessible by the customer so that it can run on behalf of the customer.

If you want to remove querying the internal settings object, then change the code of `define_internal_schemas` in the workflows above to following:

```
import { execution } from "@dynatrace-sdk/automation-utils";
```

```
export default async function ({ execution_id }) {
  const schemas: string[] = [];
  return schemas;
}
```

In the `cleanup` and `cleanup_internal` tasks, there is filtering to filter out settings that have no objects.

DQL query and dashboard

The product_limits_dashboard file provides a dashboard template with a table and visualization on a separate tile. This dashboard template can be uploaded to a tenant in the "Dashboards" section without manual setup to quickly get an overview of the workflow results in a user-friendly format. The DQL queries and dashboard settings are already included in the template.

If you prefer a manual setup of the dashboard, this is the basic DQL query you can put on a dashboard to get the results of the workflow execution:

```
// Fetching business events of type "dynatrace.limits" from the
Dynatrace environment
fetch bizevents
| filter event.type == "dynatrace.limits"

// Filtering the results to include only events where the limit
field is not null
| filter isNotNull(limit)

// Removing duplicate entries based on the "schemaId" field
| dedup schemaId, sort: { timestamp desc }

// Selecting specific fields to work with
| fields timestamp, schemaId, displayName,
    descriptionOriginal = description,
    current, limit, level = (current * 1.0/limit) * 100

// This variable will not work without a dashboard variable
LevelThreshold from the dashboard template
/*
// Filter level by threshold
| filterOut if(
    $LevelThreshold != "none",
    ((toLong(level) < toLong($LevelThreshold)) or isNull(level)),
    else: false
)
*/

// Sorting the results by whether "current" is not null and by
percentage of limit used
```

```
| sort isNotNull(current) desc, level desc

// Adding a new field "description" by transforming
"descriptionOriginal"
| fieldsAdd description = replacePattern(descriptionOriginal,
"'#'+", "######")

// Adding a new field "schema" that creates a clickable Markdown
link using the schemaId
| fieldsAdd schema = concat("[", schemaId, "]
(/ui/apps/dynatrace.classic.settings/ui/settings/", schemaId, ")")

// Assigning visual indicators to different levels
// 🟢: Usage < 70%
// 🟠: Usage between 70% and 100%
// 🔴: Usage >= 100%
| fieldsAdd status = if (level < 70, "🟢")
| fieldsAdd status = if (level >= 70, "🟠", else: status)
| fieldsAdd status = if (level >= 100, "🔴", else: status)

// Removing unnecessary fields (timestamp, descriptionOriginal,
and schemaId)
| fieldsRemove timestamp, descriptionOriginal, schemaId

// Specifying the fields to display on the dashboard
| fields displayName, status, level, current, limit, description,
schema
```

Dashboard settings for better readability

If you decided to set up your dashboard manually and used a DQL query above, you can enhance it for better readability. To achieve that, go to the tile settings on your dashboard and click on "Visual" tab located between "Data" and "Tile" tabs. On this tab, you can make some adjustments for better readability.

In the "Columns" section:

1. Toggle "Line wrap", then click "Custom" and open the dropdown below "Custom". In the dropdown, choose "displayName" and "description" columns.
2. Next to "Custom column types" click on "+ Column type". First, choose "description" in the dropdown and then open a dropdown under "ABC" directly to the right from "description" and choose the option "Markdown" in the dropdown list. Then click on "+ Column type" again, choose "schema" in the dropdown and choose the option "Markdown" to the right from "schema".

In the "Cells" section:

1. Among three values next to the "Density", click on "Default".

In the "Units and formats" section:

1. Click the " + " (plus) icon to the right. Choose "level" in the small dropdown to the right that appears on the screen. Then click on the small dropdown to the left. A small pop-up will appear. In the pop-up, click on the dropdown under "Units" (when nothing is chosen, it says "None") and choose the option "Percent" from the list. You can double-check the "Decimals" and change this setting, it is set to "0.00" by default. You can leave it as is.

These and other settings are already included in the dashboard template.

Workflow usage monitoring

To monitor workflow execution, a "dummy task" called `sfm_bookmark` is added. It executes following free of charge query:

```
data record(dt_limits_workfow = 1)
```

This triggeres a log line in our self monitoring environment. With following query you can see the execution by environment:

```
fetch logs, scanLimitGBytes: -1, from: -30d
| filter dt.system.bucket ==
"custom_sen_low_query_frontend_dql_logs"
| filter dql == "data record(dt_limits_workfow = 1)"
| fields timestamp,tenant
```