roduct or feature is in a pre-release state and might change or have limited support. For more information, see the pr
 stages (/products/#product-launch-stages).

Service Monitoring adds the following resources to the Monitoring API:

- `services` (/monitoring/api/ref_v3/rest/v3/services)
- `services.serviceLevelObjectives` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives)

This page illustrates primary uses of the new API resources. For a general overview of the structures used here, see Constructs in the API (/monitoring/service-monitoring/api-structures). Comprehensive reference material appears under Stackdriver Monitoring API v3 (/monitoring/api/ref_v3/rest/).

This document collectively refers to these additions as the Service Monitoring API.

The Service Monitoring API can be used to manage services and service-level objectives (SLOs). This page focuses on custom services and service-level indicators (SLIs). Services running on App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints are automatically detected, and the SLIs for them are predefined. To define SLOs, you can use either the Anthos Service Mesh console or the Service Monitoring API. For more information on creating SLOs by using the Anthos Service Mesh dashboard, see Anthos Service Mesh documentation: Creating SLOs (/service-mesh/docs/observability/create-slo).

You can also define your own services and their SLOs. To do these, you must use the Service Monitoring API. There is no UI support available.

The examples on ths page focus primarily on custom services and SLIs.

Several methods in the Service Monitoring API use identifiers for different elements, particularly project IDs and service IDs. These IDs adhere to the following rules:

- Length: between 1 and 63 characters
- Characters: only lower-case letters, number, and hyphens
- Initial character: lower-case letter
- Terminal character: lower-case letter or a number, but not a hyphen

The regex for these rules is as follows: `[a-z](?:[-a-z0-9]{0,61}[a-z0-9])?`

This section describes the conventions and setup used for invoking the Service Monitoring API by using the `curl` tool. If you are using this API through a client library, you can skip this section.

The examples on this page access the Service Monitoring API by using the `curl` tool to send HTTP requests to REST endpoints. Use the following information on authentication and invoking `curl` to set the variables used in the invocations.

1. Create an environment variable to hold the ID of your Stackdriver Monitoring Workspace. These examples use `PROJECT_ID`:

2. Authenticate to the Cloud SDK:

3. To avoid having to specify your project ID with each command, set it as the default by using Cloud SDK:

4. Create an authorization token and capture it in an environment variable. These examples call the variable `ACCESS_TOKEN`:

You have to periodically refresh the access token. If commands that worked suddenly report that you are unauthenticated, re-issue this command.

5. To verify that you got an access token, echo the `ACCESS_TOKEN` variable:

Each `curl` invocation includes a set of arguments, followed by the URL of a Service Monitoring API resource. The common arguments include values specified by the `PROJECT_ID` and `ACCESS_TOKEN` environment variables. You might also have to specify other arguments, for example, to specify the type of the HTTP request (for example, `-X DELETE`). The default request is `GET`, so the examples don't specify it.

Each `curl` invocation has this general structure:

```
curl --http1.1 --header "Authorization: Bearer ${ACCESS_TOKEN}" <other_args>
https://monitoring.googleapis.com/v3/projects/${PROJECT_ID}/<request>
```

For example, to list all the services in your project, issue the following `GET` request:

This returns an array of service descriptions, with entries like the following, an Istio service called “currencyservice”:

See `Service` (/monitoring/api/ref_v3/rest/v3/services#resource-service) for more information about the structure of a service.

The `Service` (/monitoring/api/ref_v3/rest/v3/services#resource-service) resource acts as the root element for organizing your services. Aspects of a particular service, like its SLOs for example, are organized under the name of the service.

Services on App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints are created automatically for you. You can do things like add SLOs to these services by using the Anthos Service Mesh console or the Service Monitoring API.

Services you create manually are called *custom services*. Custom services can be created, deleted, retrieved, and updated only by using the Service Monitoring API.

After you have identified or created a service, you can add SLOs to it. Managing SLOs (#managing-slos) covers commands to manipulate SLOs.

Each service has fully-qualified identifier called the *resource name*. This identifier is stored in the `name` field of the service description, for example:

Embedded in the resource name is a shorter ID for the service, the part of the resource name after the substring `projects/[PROJECT_NUMBER]/services/`

If you created your own service with the resource name `projects/[PROJECT_NUMBER]/services/my-test-service`, the service ID is `my-test-service`.

For brevity and accuracy, many `curl` examples assume the service ID is held in the environment variable `SERVICE_ID` so you can refer to it repeatedly.

To retrieve information about all the services in your project, invoke the `services.list` (/monitoring/api/ref_v3/rest/v3/services/list) method. To retrieve information about a specific service by service ID, use the `services.get` (/monitoring/api/ref_v3/rest/v3/services/get) method:

If you are not using an environment where services are automatically created (that is, App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints), then you can create services by using the `services.create` (/monitoring/api/ref_v3/rest/v3/services/create) method.

If you manually create services, you then have to manually add the SLOs and other service-monitoring artifacts to the service structure by using the Service Monitoring API. For an overview of these structures, see Constructs in the API (/monitoring/service-monitoring/api-structures).

To create a service, you must specify a display name for the service and a field named `custom` with an empty object. You can optionally specify the service ID you want the service to have.

To delete a custom service, invoke the **`services.delete`** (/monitoring/api/ref_v3/rest/v3/services/delete) method and specify the service ID.

SLOs are associated with services. You can create SLOs using the Anthos Service Mesh console for App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints, or by using the Service Monitoring API. You must use the Service Monitoring API to create SLOs for custom services.

You can also use the Service Monitoring API to retrieve the SLOs associated with a service, and to delete SLOs from a service.

To manage SLOs for a service, you must have the service ID. SLOs are named based on the service they belong to. Service IDs (#service-ids) describes how to recognize service IDs.

To retrieve information about all the SLOs associated with a service, invoke the **`services.serviceLevelObjectives.list`** (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives/list) method. To retrieve information about a specific SLO by name, use the **`services.serviceLevelObjectives.get`** (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives/get) method:

The following is an availability SLO retrieved from the “currencyservice” service:

This SLO is built on an availability SLI, it sets a target performance goal of 98 percent, and it measures compliance over a calendar week. For more information on availability SLIs, see Service-level indicators (/monitoring/service-monitoring/#defn-sli).

See `ServiceLevelObjective` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives#resource-servicelevelobjective) for more information about the structure of SLOs.

Each SLO has a unique identifier within the service, consisting of the string following `serviceLevelObjectives` in the SLO's `name` field. In the following example:

the SLO ID is the string `3kavNVTtTMuzL7KcXAxqCQ`.

To retrieve information about this particular SLO, request the SLO by ID.

To create an SLO by using the Service Monitoring API, you must create a `ServiceLevelObjective` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives#resource-servicelevelobjective) object and pass it to the `serviceLevelObjectives.create` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives/create) method. The structure representing an SLO has a number of embedded structures, including an available SLI.

The structure of an SLO has a number of embedded structures, including one for the value of the `serviceLevelIndicator` field.

- For App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints services, the SLIs are pre-defined. You can use the Anthos Service Mesh console to create SLOs; all you have to do is specify the performance goals and compliance periods.

  You can also define SLOs by using the Service Monitoring API.

- For custom services, you must define SLOs by using the Service Monitoring API. To do this, you must create a value for the `serviceLevelIndicator` field, as well. See Creating a service-level indicator (/monitoring/service-monitoring/identifying-custom-sli) for some techniques.

The basic skeleton for building the SLO is as follows:

You need to specify the following:

- Display name: a description of the SLO
- A service-level indicator, which is one of the three types:
    - `BasicSli` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives#basicsli)
    - `RequestBasedSli` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives#requestbasedsli)
    - `WindowsBasedSli` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives#windowsbasedsli)
- The performance goal (a percentage)
- The compliance period, which one of two types:
    - A rolling period of some length (in seconds)
    - A calendar period

For more information about SLIs, performance goals, and compliance periods, see Concepts in servide monitoring (/monitoring/service-monitoring).

For App Engine, Istio on Google Kubernetes Engine, and Cloud Endpoints services, the SLI will be a basic SLI.

For custom services, you have to create a request-based SLI or a windows-based SLI. See Creating a service-level indicator (/monitoring/service-monitoring/identifying-custom-sli) for some techniques.

After you have the SLI, you can build the SLO. The following example defines an SLO for a service that uses a basic SLI. You might have several SLOs on a single SLI, for example, one for 95%

availability and one for 99% availability. The following example is an SLO for 95% availability over a calendar week:

This example show an SLO for 75% availability over a rolling 3-day period:

To delete an SLO, invoke the `services.serviceLevelObjectives.delete` (/monitoring/api/ref_v3/rest/v3/services.serviceLevelObjectives/delete) method and specify the ID of the SLO in your project:

SLO data is stored in time series, so you can use the `timeSeries.list` (/monitoring/api/ref_v3/rest/v3/projects.timeSeries/list) method to retrieve it. However, this data isn't stored in standard metric types, so you can't use the standard mechanism of specifying a metric-type filter to the `timeSeries.list` method.

Instead, SLO time series are retrieved by specifying another type of filter, a **time-series selector**, to the `timeSeries.list` method in the `filter` parameter. See Retrieving SLO data (/monitoring/service-monitoring/timeseries-selectors) for information on using these selectors.

You also use time-series selectors to set up alerting policies programmatically. See Alerting on your burn rate (/monitoring/service-monitoring/alerting-on-budget-burn-rate) for more information.