Example SLO Document

This document describes the SLOs for the Example Game Service.

Status	**Published**
Author	Steven Thurgood
Date	2018-02-19
Reviewers	David Ferguson
Approvers	Betsy Beyer
Approval Date	2018-02-20
Revisit Date	2019-02-01

Service Overview

The *Example Game Service* allows Android and iPhone users to play a game with each other. The app runs on users' phones, and moves are sent back to the *API* via a REST API. The *data store* contains the states of all current and previous games. A *score pipeline* reads this table and generates up-to-date league tables for today, this week, and all time. League table results are available in the app, via the API, and also on a public *HTTP server*.

The SLO uses a four-week rolling window.

SLIs and SLOs

Category	**SLI**	**SLO**
API		
Availability	The proportion of successful requests, as measured from the load balancer metrics. Any HTTP status other than 500–599 is considered successful. count of "api" http_requests which do not have a 5XX status code divided by count of all "api" http_requests	97% success
Latency	The proportion of sufficiently fast requests, as measured from the load balancer metrics. "Sufficiently fast" is defined as < 400 ms, or < 850 ms. count of "api" http_requests with a duration less than or equal to	90% of requests < 400 ms 99% of requests < 850 ms

Category	SLI	SLO
	"0.4" seconds divided by count of all "api" http_requests count of "api" http_requests with a duration less than or equal to "0.85" seconds divided by count of all "api" http_requests	
HTTP server		
Availability	The proportion of successful requests, as measured from the load balancer metrics. Any HTTP status other than 500–599 is considered successful. count of "web" http_requests which do not have a 5XX status code divided by count of all "web" http_requests	99%
Latency	The proportion of sufficiently fast requests, as measured from the load balancer metrics. “Sufficiently fast” is defined as < 200 ms, or < 1,000 ms.	90% of requests < 200 ms 99% of requests < 1,000 ms

Category	SLI	SLO
	count of "web" http_requests with a duration less than or equal to "0.2" seconds divided by count of all "web" http_requests count of "web" http_requests with a duration less than or equal to "1.0" seconds divided by count of all "web" http_requests	
Score pipeline		
Freshness	The proportion of records read from the league table that were updated recently. “Recently” is defined as within 1 minute, or within 10 minutes. Uses metrics from the API and HTTP server: count of all data_requests for "api" and "web" with freshness less than or equal to 1 minute divided by count of all data_requests count of all data_requests for "api" and "web" with freshness less than or equal to 10 minutes divided by count of all data_requests	90% of reads use data written within the previous 1 minute. 99% of reads use data written within the previous 10 minutes.

Category	SLI	SLO
Correctness	The proportion of records injected into the state table by a correctness prober that result in the correct data being read from the league table. A correctness prober injects synthetic data, with known correct outcomes, and exports a success metric: count of all data_requests which were correct divided by count of all data_requests	99.99999% of records injected by the prober result in the correct output.
Completeness	The proportion of hours in which 100% of the games in the data store were processed (no records were skipped). Uses metrics exported by the score pipeline: count of all pipeline runs that processed 100% of the records divided by count of all pipeline runs	99% of pipeline runs cover 100% of the data.

Rationale

Availability and latency SLIs were based on measurement over the period 2018-01-01 to 2018-01-28. Availability SLOs were rounded down to the nearest 1% and latency SLO timings were rounded up to the nearest 50 ms. All other numbers were picked by the author and the services were verified to be running at or above those levels.

No attempt has yet been made to verify that these numbers correlate strongly with user experience.[1]

Error Budget

Each objective has a separate error budget, defined as 100% minus (–) the goal for that objective. For example, if there have been 1,000,000 requests to the API server in the previous four weeks, the API availability error budget is 3% (100% – 97%) of 1,000,000: 30,000 errors.

We will enact the error budget policy (see Example Error Budget Policy) when any of our objectives has exhausted its error budget.

Clarifications and Caveats

- Request metrics are measured at the load balancer. This measurement may fail to accurately measure cases where user requests didn't reach the load balancer.
- We only count HTTP 5XX status messages as error codes; everything else is counted as success.
- The test data used by the correctness prober contains approximately 200 tests, which are injected every 1s. Our error budget is 48 errors every four weeks.

[1] Even if the numbers in the SLO are not strongly evidence-based, it is necessary to document this so that future readers can understand this fact, and make their decisions appropriately. They may decide that it is worth the investment to collect more evidence.

PREVIOUS
Conclusion

NEXT
Appendix B - Example Error Budget Policy