**Spotify**

# **Introduction**

## Description

This document defines testing approach for Spotify process verification. It briefly describes methods and tools to be used to evaluate and tune performance.

## Purpose

The purpose of this document is to outline the approach for Performance Testing team to follow to assure Performance Acceptance Criteria to be defined and met. Specifically, this document details:

* Performance Acceptance Criteria;
* Performance Testing workflow;
* Load Model;
* Test types to be performed;
* Measurements to be collected;
* Tools and infrastructure.

## Scope

This document provides strategy to carry out all performance analysis activities for the project. It briefly describes resources required, including toolset to accomplish test executions, results analysis, and performance tuning. It covers Performance Acceptance Criteria, explains system interaction models to be tested, and describes scripts to be developed. This strategy doesn’t include functional testing, nor does it guarantee any specific performance results. The primary objectives for this testing effort are to:

* Validate that the Performance Acceptance Criteria are met by the system
* Identify and ensure that performance related defects are addressed prior to deployment

# **Performance acceptance criteria**

## Introduction

Acceptance criteria provide the essentials to be satisfied to accept the application by the end user. Performance efforts associated with two sets of acceptance criteria:

* requirements and objectives;
* engagement criteria.

In the sections below, both types are explained in general and in specific detail for the Spotify performance optimization effort. The efforts will be deemed complete when either all performance criteria are met, or any one of the engagement completion criteria is met.

## Identifying the Performance Testing Objectives

Two major goals is set to be addressed by current plan:

1. Assure that future releases do not cause performance degradations and introduce performance improvements

Systems features for testing:

* Case management actions for Free account
* Case management actions for Premium account

## Non-Functional Requirements must be defined and met for Spotify to go live successfully.

## Defining Performance Pass/Fail Criteria

We need to define what constitutes passing the goal and what constitutes falling short of achieving it. The pass/fail criteria should be unambiguous so that they are measurable in absolute terms. Otherwise, the clients may challenge the pass/fail criteria later. The strategy for Spotify is to achieve step by step the goals set.

Step 1: Define user scenarios.

Step 2: Design workloads based on API endpoint profiles.

Step 3: Define Entry/Exit criteria for each workload.

Step 4: Define proper measurement criteria for each workload.

Step 5: Set the pass/fail criteria for each test run with each workload.

Performance parameters and pass/fail criteria may overlap for a set of workloads.

# **Load model**

## Introduction

Workload is an instrument simulating the real-world environment. It is an integrated part of the test execution process. Once user journeys or API endpoint profiles are known, a detailed workload can be planned. The workload provides in-depth knowledge of behaviors in proposed system. It explains how typical load will spread once the system go live. In addition, the workload helps to understand the requirements in a structured way. Because the system performance capability viewed from different perspectives, like meeting the goals set by the users, impact on the architecture, and growth potential of the system and so on.

# **Workflow**

## Introduction

This section describes the concepts underlying the activities necessary to make performance testing successful within an iterative process, as well as specific, actionable items that can be immediately applied for the Spotify to gain a significant return on this investment. The key to working within an iteration-based work cycle is team coordination. For this reason, performance analysts must be able to adapt what is measured and analyzed per iteration cycle as circumstances change.

## Iterative Performance Testing Activities

This approach can be represented by using the following nine activities below:

* **Activity 1. Understand the Project and Context**. The outcome of this activity is a shared understanding of the project vision and context.
* **Activity 2. Identify Reasons for Testing Performance**. Explicitly identify the reasons for performance testing.
* **Activity 3. Identify the Value Performance Testing Adds to the Project**. Translate the project- and business-level objectives into specific, identifiable, and manageable performance-testing activities.
* **Activity 4. Configure the Test Environment.** Set up the load-generation tools and the system under test, collectively known as the performance test environment.
* **Activity 5. Identify and Coordinate Tasks**. Prioritize and coordinate support, resources, and schedules to make the tasks efficient and successful.
* **Activity 6. How often should run performance tests**. Execute the activities for the current iteration.
* **Activity 7. Analyze Results, compare with previous executions and Report**. Analyze and share results with the team.
* **Activity 8. Revisit Activities 1-3 and Consider Performance Acceptance Criteria**. Between iterations, ensure that the foundational information has not changed. Integrate new information such as customer feedback and update the strategy as necessary.
* **Activity 9. Reprioritize Tasks**. Based on the test results, new information, and the availability of features and components, reprioritize, add to, or delete tasks from the strategy, and then return to activity 5.

### Activity 1: Understand the Project and Context

This activity takes place only at the beginning of diving to the project. The project vision and context are the foundation for determining what performance testing activities are necessary and valuable. Because the Performance Analyst is not driving these items, the coordination aspect refers more to team education about the performance implications of the project vision and context, and to identifying areas where future coordination will likely be needed for success. A critical part of working with an iteration-based process is asking the correct questions, providing the correct value, and performing the correct task related to each step. Although situations can shift or add more questions, values, or tasks, a sample checklist is provided as a starting point for each step.

**Questions to ask:**

* What are the performance implications of the project vision?
* What are the performance implications of the service the application is intended to provide, or what problem are we trying to solve for the customer?
* How does the team envision performance testing as it relates to the project schedule, structure, and available resources?

**Value provided:**

* Be involved in the product concept.
* Point out any areas of concern immediately.
* Point out assumptions related to available resources, tools, and resource-monitoring instrumentation based on the project vision and context as soon as they arise.

**Tasks accomplished:**

* Ask the whole team questions and provide answers.
* Determine the team’s perception of performance testing.
* Gain a conceptual understanding of the project’s critical performance implications.
* Begin to define equipment and/or resources needed for conducting performance testing.
* Understand resource constrains; for example, budget, people, equipment.
* Understand how the team will coordinate.
* Understand how the team will communicate.

**Coordinate with:**

* Whole team

### Activity 2: Identify Reasons for Testing Performance

The underlying reasons for performance testing on the project is not always obvious and shouldn’t be based on the vision and context alone. Project teams generally do not include performance testing as part of the project unless there is some performance-related risk or concern they feel needs to be mitigated. Explicitly identifying these risks and areas of concern is the next fundamental step in determining what specific performance testing activities will add the most value to the project. Regardless of when a performance tester joins the team, once the project vision and context are understood, it is worth taking the time to verbalize and/or document the overall objectives of the performance-testing effort based on the risks or concerns that the team has. The following checklist should help to accomplish this step.

**Questions to ask:**

* What risk(s) is performance testing intended to mitigate for this project?
* Are there specific contractual, compliance, or customer performance expectations that are already known to be required?
* What performance concerns relating to this project already exist?

**Value provided:**

* Be involved in the product concept.
* Point out any areas of concern immediately.
* Point out resource and instrumentation assumptions based on the project vision and context when they arise.
* Guide the process of collecting/determining performance-testing objectives.
* Capture implied usage scenarios of performance concerns.
* Capture implied performance goals, requirements, targets, and thresholds as they come up in conversation.

**Tasks accomplished:**

* Ask the whole team questions and provide answers.
* Determine the project-level objectives for conducting performance testing.
* Refine estimates of equipment and/or resources required for conducting performance testing.
* Identify disconnects between the objectives of the performance-testing effort and the equipment and resources to be made available.
* Capture implied performance goals, requirements, targets, and thresholds to be fleshed out later.
* Capture implied usage scenarios of concern to be fleshed out later.

**Coordinate with:**

* Whole team

### Activity 3: Identify the Value Performance Testing Adds to the Project

Using information gained from activities 1 and 2, we can now clarify the value added through performance testing, and convert that value into a conceptual performance-testing strategy. The point is to translate the project- and business-level objectives into specific, identifiable, and manageable performance-testing activities. The coordination aspect of this step involves team-wide discussion and agreement on which performance-testing activities are likely to add value or provide valuable information, and if these activities are worth planning for now.

**Questions to ask:**

* What performance-testing activities will help achieve the performance-testing objectives?
* What performance-testing activities are needed to validate any contractual, compliance, project, or customer performance criteria or expectations that are known?
* What performance-testing activities will help address currently known performance concerns?

**Value provided:**

* Ensure team-wide support of performance-testing activities.
* Ensure that the team has adequate warning about performance-testing activities that will require the support of additional team members.
* Determine if resource and instrumentation assumptions are adequate.
* Guide the process of determining how performance-testing objectives will be measured.
* Capture additional implied usage scenarios of performance concerns.
* Capture additional implied performance goals, requirements, targets, and thresholds as they come up in conversation.

**Tasks accomplished:**

* Ask the whole team questions and provide answers.
* Determine a conceptual project-level strategy for determining if the objectives for conducting performance testing have been met.
* Refine estimates of equipment and/or resources required for conducting performance testing.
* Identify disconnects between the objectives of the performance-testing effort and the equipment and resources to be made available.
* Capture additional implied performance goals, requirements, targets, and thresholds to be fleshed out later.
* Capture additional implied usage scenarios of concern to be fleshed out later.

**Coordinate with:**

* Whole team

### Activity 4: Configure the Test Environment

With a conceptual strategy in place, the tools and resources should be prepared to execute the strategy as features and components become available for test. This step should be performed as soon as possible, so that the team has this resource from the beginning. And finally – performance acceptance environment should be close to production environment.

Set up the load-generation tools and the system under test — collectively known as the performance test environment — and ensure that this environment will meet engineering needs. The coordination component of this step typically involves asking managers and administrators to obtain and/or configure equipment and other resources that are not under the direct control of the team or performance tester.

**Questions to ask:**

* Who administrates the performance-testing environment of the application under test?
* Who administrates the load-generation tool/environment?
* Who configures and operates resource monitors for the application under test?
* Is special permission needed prior to generating load of a certain volume?
* Who can reset the application under test?
* What other components require special coordination?
* What security or authentication considerations are there for simulating multiple users?
* What coordination needs to be done to enable the use of recording and/or monitoring software?

**Value provided:**

* Ensure that the load-generation and performance-test environments are ready when the team needs them.
* Ensure that the entire team knows who to contact for help with performance-testing environment support.
* Ensure that performance testing support staff knows what they are supporting.

**Tasks accomplished:**

* Performance-test environment configured and ready to begin testing.
* Load-generation environment configured and ready to begin testing.
* Support responsibilities assigned.
* Special permissions, time of day for high load tests, etc., determined.

**Coordinate with:**

* System administrators
* Network support
* Database administrators
* Infrastructure support
* Managers of those above
* Development team

### Activity 5: Identify and Coordinate Tasks

Performance testing tasks do not happen in isolation. The performance specialist needs to work with the team to prioritize and coordinate support, resources, and schedules to make the tasks efficient and successful.

During the pre-iteration planning meeting, look at where the project is now and where you want to be to determine what should and can be done next. Performance analyst is driven by the goals that have been determined for this cycle during planning. It also includes signing up for the activities that will be accomplished during this cycle.

**Questions to ask:**

* What is the performance goal for this cycle?
* Where is the project in terms of the overall project performance goals?
* Has the system achieved all performance objectives?
* Has tuning been accomplished since the last iteration?
* What analysis, reports, or retesting will add value during this iteration?
* Who requires pairing to do performance testing?
* How much time is available?
* How much time does each task take?
* What is the most critical activity?

**Value provided:**

* Provide insight on how the overall project is achieving its goal.
* Provide insight on what can be measured and reported on in this cycle.
* Provide insight on any critical issues that may have arisen from the last iteration cycle.
* Make suggestions to other team members.
* Transfer lessons learned as they emerge from the test.
* Pair with developers to improve performance unit testing.
* Help reuse unit tests.
* Help reuse functional tests.

**Tasks accomplished:**

* Estimate how much work is achievable.
* Determine if anyone needs to be paired out.
* Prioritize achievable work.
* Identify primary and alternate tasks for this cycle.

**Coordinate with:**

* Managers and stakeholders
* Developers and administrators
* Infrastructure and test environment support
* Users or user representatives

### Activity 6: How often performance tests should be run

Conduct tasks in one- to two-day segments. See them through to completion, but be willing to take important detours along the way if an opportunity to add additional value presents itself. Step 5 defines what work the team members will sign up for in this iteration. Now it is time to execute the activities for this iteration

**Questions to ask:**

* Have recent test results or project updates made this task valuable compared to other tests we could be conducting right now?
* What additional team members should be involved with this task?
* Are there other important tasks that can be conducted in parallel with this one?
* Do the preliminary results make sense?
* Is the test providing the data we expected?

**Value provided:**

* Evaluate algorithm efficiency.
* Monitor resource usage trends.
* Measure response times.
* Collect data for scalability and capacity planning.
* Transfer lessons learned as they emerge from the test.
* Improve performance unit testing by pairing performance testers with developers.

**Tasks accomplished:**

* Conduct tests.
* Collect data.
* Validate test assumptions and techniques.
* Potentially tune while testing.
* Pair with other team members; this does not mean only working with a developer or tester but can also mean working with a writer to capture his or her understanding of how the system performance works, or working with the customer directly.

**Coordinate with:**

* Developers and administrators
* Infrastructure and test environment support
* Users or user representatives
* Managers and stakeholders
* Other performance testers who are not on the project

### Activity 7: Analyze Results, compare with previous executions and Report

To keep up with an iterative process, results need to be analyzed and shared quickly. If the analysis is inconclusive, retest at the earliest possible opportunity to give the team maximum time to react to performance issues. As the project is wrapped for final shipping, it is usually worth having a meeting afterward to collect and pass along lessons learned. In most cases, it is valuable to have a daily or every-other-day update to share information and coordinate next tasks.

**Questions to ask:**

* Do the preliminary results make sense?
* Is the test providing the data we expected?
* Is the data valuable?
* Are more tests required to derive meaning from the data?
* Is tuning required? If so, do we know what to tune?
* Do the results indicate that there are additional tests that we need to execute that have not been planned for?
* Do the results indicate that any of the tests we are planning to conduct are no longer necessary?
* Have any performance objectives been met?
* Have any performance objectives been rendered obsolete?

**Value provided:**

* Evaluate algorithm efficiency.
* Monitor resource usage trends.
* Measure response times.
* Collect data for scalability and capacity planning.
* Transfer lessons learned as they emerge from the test.

**Tasks accomplished:**

* Analyze data collaboratively.
* Determine the meaning of the results.
* Share data with the whole team.
* Import lessons learned into future iteration planning.

**Coordinate with:**

* Developers and administrators
* Managers and stakeholders
* Users or user representatives
* Other performance testers who are not on the project

### Activity 8: Revisit Activities 1-3 and Consider Performance Acceptance Criteria

Between iterations, ensure that the foundational information has not changed. Integrate new information, such as customer feedback, and update the strategy as necessary.

**Questions to ask:**

* Have the performance implications of the project vision changed?
* Have the performance implications of the service we are trying to provide changed, or has the problem we are trying to solve for the customer changed?
* Have the project schedule, structure, or available resources changed?
* Have the performance-testing objectives changed?
* Have the performance-testing activities needed to validate any contractual, compliance, project, or customer performance criteria or expectations changed?
* What performance-testing activities will help address currently known performance concerns?

**Value provided:**

* Update resource and instrumentation assumptions and needs.
* Point out any areas of concern.
* Point out resource and instrumentation needs and/or risks.
* Update performance-testing objectives.
* Enhance and update usage scenarios of performance concerns.
* Enhance and update performance goals, requirements, targets, and thresholds.
* Ensure that the team has adequate warning about upcoming performance-testing activities that will require the support of additional team members.

**Tasks accomplished:**

* Enhance and update understanding of the project’s critical performance implications.
* Update resource constraints; for example, budget, people, and equipment.
* Update/improve how the team will coordinate.
* Update/improve how the team will communicate.
* Revise performance-testing strategy.
* Refine estimates of equipment and/or resources required for conducting performance testing.
* Identify incompatibilities or conflicts between the objectives of the performance-testing effort and the equipment and resources to be made available.
* Capture additional performance goals, requirements, targets, and thresholds.
* Capture additional usage scenarios of concern.
* Report current performance-testing status.

**Coordinate with:**

* Whole team

### Activity 9: Reprioritize Tasks

Based on the test results, new information, and the availability of features and components, reprioritize, add to, or delete tasks from the strategy, and then return to activity 5.

**Questions to ask:**

* What performance-testing activities will help address currently known performance concerns?
* What is the performance goal for this cycle?
* Where is the project in terms of the overall project performance goals?
* Has the system achieved all its performance objectives?
* Has tuning been accomplished since the last iteration?
* What analysis, reports, or retesting will add value during this iteration cycle?
* Who requires pairing to do performance testing?
* How much time is available?
* How much time does each task take?
* What is the most critical activity?

**Value provided:**

* Provide insight on how the overall project is achieving its goal.
* Provide insight on what can be measured and reported on in this cycle.
* Provide insight on any critical issues that may have arisen from the last iteration.
* Make suggestions to other team members.
* Transfer lessons learned as they emerge from the test.
* Pair with developers to improve performance unit testing.
* Help reuse unit tests.
* Help reuse functional tests.

**Tasks accomplished:**

* Report current performance-testing status.
* Estimate how much work is achievable.
* Determine if anyone needs to be paired out.
* Prioritize achievable work.
* Identify primary and alternate tasks for this cycle.

**Coordinate with:**

* Managers and stakeholders
* Product owners

# **Performance test types (with priorities)**

## Server-side (API) tests

### Validation test

Priority: Highest

The pipe-clean test is a preparatory task that serves to validate each performance test script in the performance test environment. The test is normally executed for a single use case as a single virtual user for a set period or for a set number of iterations. This execution should ideally be carried out without any other activity on the system to provide a best-case measurement. You can then use the metrics obtained as a baseline to determine the amount of performance degradation that occurs in response to increasing numbers of users and to determine the server and network footprint for each scripted use case.

Baseline tests need to be ran for every iteration.

**Purpose**

 To check that system is ready for performance testing and scripts are OK

### Capacity test

Priority: Highest

Capacity of a system is the highest level of load it can take and handle without:

* Significant response times increase
* Stability decrease

In other words, capacity is the measure of how powerful the system is.

A capacity test complements load testing by determining server’s ultimate failure point. You perform capacity testing in conjunction with capacity planning, which you use to plan for future growth, such as an increased user base or increased volume of data. For example, to accommodate future loads, you need to know how many additional resources (such as processor capacity, memory usage, disk capacity, or network bandwidth) are necessary to support future usage levels. Capacity testing helps to identify a scaling strategy to determine whether system should scale up or scale down.

**Purpose**

To determine how many users and/or transactions a given system will support and still meet performance goals, to find out server capacity, stability under incremental load and scalability of the system. Also, capacity testing results are key points to create performance tests of another types, for example fixed-load or stress.

### Response time (fixed load) test

Priority: High

Fixed load testing is conducted to verify that your application can meet your desired performance objectives; these performance objectives are often specified in a service level agreement (SLA). A load test enables you to measure response times, throughput rates, and resource-utilization levels, to verify application behavior under normal and peak load conditions.

Define the load level vs capacity:

* Low-load (~10% of capacity)
* Mid-load (~45% of capacity)
* High-load (~80% of capacity)
* Or some defined level of load (e.g. production-like, expected, etc.)

Duration should be:

* long enough to make results statistically meaningful
* short enough to avoid biased errors

Benchmark tests should be used as a part of Continuous execution pipeline

**Purpose**

* to get response times (and some other metrics) statistics under different levels of load and compare them against target/previous release (build, sprint, etc.) results
* to check that system under load is stable for a particular period
* (optional) system resources/application profiling for problematic transactions

The goal of checking system stability is correct for any performance test with fixed level of load.