

# 

# 

**Pekin Insurance GW Performance Test Strategy**

Table of Contents

[Executive Summary: 2](#_Toc205486833)

[Revision History: 2](#_Toc205486834)

[Overview and Objective: 2](#_Toc205486835)

[Performance Testing Scope 2](#_Toc205486836)

[Core Systems: 3](#_Toc205486837)

[Supporting Systems and Risk Overview: **.** 5](#_Toc205486838)

[Different Types of Performance testing 6](#_Toc205486839)

[High Level Performance schedule: 6](#_Toc205486840)

[Performance Approach 8](#_Toc205486841)

[Key Performance Indexes 11](#_Toc205486842)

[Infrastructure Requirements 12](#_Toc205486843)

[Environment: 12](#_Toc205486844)

[Entry and Exit criteria 12](#_Toc205486845)

[Roles and Responsibilities 13](#_Toc205486846)

[Performance practices to be followed: 14](#_Toc205486847)

[Assumptions 15](#_Toc205486848)

[Communication Plan: 15](#_Toc205486849)

[Requirements 16](#_Toc205486850)

[Test Execution Flow 16](#_Toc205486851)

## **Executive Summary:**

This strategy outlines the performance testing approach for Pekin Insurance’s Guidewire cloud environment, with a focus on validating critical business functions under expected load and identifying system bottlenecks. The plan includes testing across Personal and Commercial lines, emphasizes the need for alignment with current business volumes, and introduces realistic user behavior modeling via JMeter and Datadog. The strategy aligns with upcoming deployment milestones and will be used to establish performance benchmarks and support go/no-go decisions.

## **Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Verison** | **Date** | **Revision Author** | **Description of changes** |
| 1.0 | 06/10/2025 |  | Initial Draft |
| 2.0 | 06/30/2025 |  | Reviewed Copy |
| 3.0 |  |  | Signed off |

## **Overview and Objective:**

We are initiating a performance testing effort to evaluate the current systems’ ability to meet expected service level agreements (SLAs) and ensure it can handle projected workloads efficiently. The primary goals of this initiative are to establish a baseline benchmark, assess systems responsiveness and stability, and proactively identify any performance bottlenecks.

It is important to note that the last performance test was conducted over two years ago. Since then, there have been significant changes to system architecture, user volume, and business operations. This makes it critical to revalidate the system’s performance and ensure it aligns with current expectations and growth plans.

An additional Key focus for this project is evaluating the performance of the PEX, as this marks the first-time implementation of the CL products. This will help us ensure the system operates efficiently and reliably under real-world conditions.

**Objectives of this document**:

* Define Performance scope, test iterations and schedule
* Establish testing metrics to track, monitor and report Performance testing
* Define testing / support roles and responsibilities
* Outline testing infrastructure needs including tools and environments

## **Performance Testing Scope**

Performance testing scope for GW cloud migration is defined in the table below. Depending on the availability of documented non-functional requirements (NFRs), the Performance team will take the following approach:

1. NFR defined: compare the actual outcome with NFR and decide on acceptance of performance benchmarks.
2. NFR not defined: execute the tests at predefined load levels and present the findings to Architect/Business/Product Owners to get acceptance.
3. Batch jobs will be executed with the expected largest volume plus 20% additional load.
4. Load tests will be performed with at least 20 concurrent users per Lob specific transactions. Approximately 1,200 concurrent users over the course of one or two hours, encompassing two business lines (Pl and CL) and all product lines (PA, HO, PUMB, CA, DBOP, WC, CPP).
5. Define the load levels based on the current production Business Volume Metrics (BVM). Work with the business and the respective teams to obtain the latest BVM.

## **Core Systems:**

The scope of the performance testing will cover all the critical business functions across Insurance suite applications for all Personal Line and Commercial Line products.

|  |  |  |
| --- | --- | --- |
| **Area of Testing** | **Scope** | **Scope-Details** |
| PC | Core Functionality | Quick Quote |
| Quote Submission |
| Requoting |
| Issue Policy |
| Bind Policy |
| Change-Reinstate-Cancel Transaction |
| Search Policy |
| Search Account |
| View Account Summary Page |
| Renewal Processing |
| Adding notes |
| Adding Activities |
| Search for a contact |
|  | Comp rater (PA, HO) | Quote creation |
| BC | Core Functionality | Invoice Batch Job |
| Invoice Due Batch Job |
| Change Payment Method |
|  |  | Automatic disbursement |
| CC | Core Functionality | Create Claim |
| Capture first notice of loss |
| Search for a policy |
| Bulk Invoice |
| Adding Notes |
| Adding Activities |
| Search for a claim |
| Search for a contact |
| Search for a check |
| Search for a recovery |
| View claim summary page |
| Close a Claim |
| PE | Core Functionality | Quote creation  Policy Change |

Additional scenarios presented by the business, and the decisions made regarding them from the perspective of performance testing feasibility.

**Billing Center:**

In the context of BC scenarios, the business has outlined the following additional scenarios:

1. The Paperless Bill Manager
2. Check Equity
3. Workflow
4. Commission Batch Jobs

Please note that the scenarios involving Check Equity, Workflow, and Commission Batch Jobs necessitate clock advancement for testing. Consequently, these have been classified as non-testable.

Regarding the Paperless Bill Manager, it is essential to acknowledge the email associated with it. However, performing an email acknowledgment is not feasible within the scope of performance testing. We are currently exploring alternative methods to generate paperless account data and will decide on inclusion based on our findings.

**Claim Center:**

The Claims business has proposed an additional optional scenario: Bulk invoice. This scenario is currently under review for feasibility due to its requirement for file transfer. We will consider moving forward with this scenario if it meets feasibility standards and if time permits.

## **Supporting Systems and Risk Overview:**

|  |  |
| --- | --- |
| **Area of Testing** | **Scope** |
| Sitefinity | Excluded |
| OnBase | Excluded |
| Payment Centers | OneInc |

Reasons for Excluding Sitefinity and OnBase from the Scope:

**Sitefinity**

The scenario considered for this GW Performance test involves selecting the product on Sitefinity and navigating to the PE for quote creation.

Currently, there is no performance environment available for Sitefinity. Conducting this test in a mixed environment would not provide accurate results. Should the need arise to perform testing for Sitefinity in the future, we can certainly include it in the scope.

**OnBase**

In document processing, costs are incurred for each document processed. Considering that we have production data in the performance environment, the potential volume of documents may result in considerable expense. To mitigate these costs, the OnBase instance has been disabled. If the need arises to conduct performance testing on the documents, we can revisit this decision.

**Payment Centers**

The performance environment is currently unavailable for Payment Center applications. Additionally, the Payment Center application is under evaluation as part of the Alpha-MYPI project. This test can be re-executed in the future if there are any changes to the application or if customer volume increases. This will allow us to assess performance and fine-tune the application to ensure optimal performance.

**Different Types of Performance testing**

|  |  |
| --- | --- |
| **Types of Testing** | **Objectives** |
| Profile | Identify the performance of application transactions, with minimal load on the system. For example, a single virtual user is assigned to each test scenario, and all transactions are executed at least once. Response times are theoretically the best the system can achieve; however, throughput and stability are not the objective. |
| Load Testing | This test is designed to evaluate system performance against the target transaction load to establish a performance baseline. It measures expected throughput and transactions response times under typical usage conditions, closely simulating the system’s normal behavior. |
| Stress Testing | Analyze system performance and behavior during continuous and excessive load testing to evaluate system stability and reliability during spikes. This usage pattern mimics a flood of users who may sign-on to the system for certain activities, such as Renewals. It also helps identify the maximum load for acceptable performance and stability. |
| Endurance Test | Identify system performance at 80-90% of expected volume for extended periods to evaluate long term stability and expose resource leaks.  Endurance testing is performed to evaluate the system’s stability and performance under a sustained load over an extended period. It is designed to uncover issues such as memory leaks, performance degradation, resource exhaustion, and potential system crashes that may not surface during shorter test cycles.  **Test Duration:**  **Short Duration Test**: Typically runs for 6 to 8 hours to simulate a standard business day and detect early signs of degradation.  **Standard Duration Test**: Runs for 12 to 14 hours to evaluate a system’s behavior over prolonged use, closely mirroring real-world usage patterns and identifying long term issues. |

## **High Level Performance schedule:**

This Schedule is created based on the Mammoth and PEX release plan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Task Name** | **Start** | **Finish** |
| 1 | Performance Test plan/ strategy | 04/15/2025 | 07/01/2025 |
| 2 | Gather NFR and determine the load distribution | 04/20/2025 | 07/01/2025 |
| 3 | Performance Scripting | 07/01/2025 | 07/30/2025 |
| 4 | Performance Environment Set up | 07/02/2025 |  |
|  | Benchmark Round 1 | 08/11/2025 | 08/12/2025 |
| 6 | Analyze the results | 08/12/2025 | 08/15/2025 |
| 6 | Benchmark- Round2 | 08/18/2025 | 08/19/2025 |
| 7 | Analyze the Test Results | 08/19/2025 | 08/22/2025 |
| 8 | Benchmark- Round3 | 08/22/2025 | 08/25/2025 |
| 9 | Analyze the Test Results | 08/25/2025 | 08/28/2025 |
| 10 | Benchmark- Round4 | 09/03/2025 | 09/03/2025 |
| 11 | Analyze the Test Results | 09/04/2025 | 09/08/2025 |
| 12 | Final Test summary Report | 09/15/2025 |  |

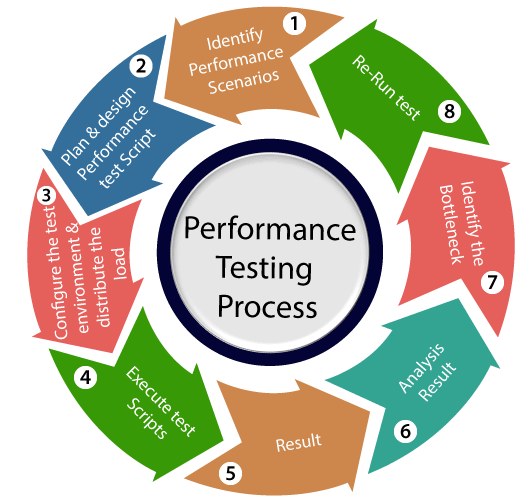
## 

## 

## 

## **Performance Approach**

Test preparation, execution, closure of activities, and deliverables for Performance are outlined below.



**Identify**:

Performance scope is defined, gathering Nonfunctional requirements, identifying the business-critical transactions for the test cases, and the Key Performance Indexes to be monitored.

**Plan and Design Performance script**:

Set up the tool, Generation of performance test scripts based on the identified business transactions.

**Configure the test environment:**

After writing the test scripts, arrange the testing environment before the execution.

Ensure required user roles/permissions are in place.

Confirm necessary monitoring tools are enabled and configured.

Document any known environment difference that could affect test interpretation.

**Validate Data and System availability:**

**Data Validation Checklist:**

Confirm the correct data sets e.g., masked production-like data are loaded. Note: Currently GW does not have the provision to load masked data. Hence, we are only loading unmasked data.

Ensure data volume reflects production –like conditions (e.g. number of customers, transactions, products).

Validate the data covers all test scenarios.

Identify test data refresh procedures if multiple test rounds are planned. Note: Ideally, we aim to refresh the environment with production-like data before each round of testing. However, considering the volume of production data, if refreshing before every round becomes too cumbersome, we will load the data once before the major test phases.

The first few rounds will focus on testing the Mammoth release code and the subsequent tests will be based on the PEX code. Therefore, we plan to refresh the environment twice: once before testing the Mammoth release and again before starting tests on the PEX code.

**System Availability validation:**

Confirm all dependent systems (APIs, databases, third-party services) are accessible.

Ensure environments are stable and reflect production architecture.

Define a process to check system health before each test run (e.g., smoke tests, monitoring alerts)

**Execute test scripts**

After distributing the load, we will execute, validate, and monitor the test scripts.

**Result**

After executing the test scripts, gather the test results. The maximum, average and minimum duration of the RPS will be captured for every scenario.

**Analyze result**

Analyze the test results by comparing it to the pre-defined NFR to determine if it meets with the response time or not.

**Identify the Bottleneck**

Identify the bottlenecks (bug or performance issue) if there are any. The bottleneck could occur because of these aspects like the problem in code, hardware issue (hard disk, RAM Processor), network issues, and software issue (operating system). After finding the bottleneck, perform tuning (fix or adjustment) to resolve this bottleneck.

**Re-run test**

Once resolving the bottlenecks, re-run the test scripts and check the results to ensure it meets with the goal.

## **Key Performance Indexes**

**Client Side**

|  |  |  |
| --- | --- | --- |
| **Metric** | **Units of Measure** | **Description** |
| **Transaction Response Time**   * **Minimum** * **Average** * **Maximum** | **Seconds** | * Response time is the total time it takes after the client sends a request till it gets a response |
| **Peak Response Time** | **Seconds** | * Similar to the previous metric, Peak Response Time, measures the round trip of a request/response cycle. However, the peak will tell us what the LONGEST cycle at this point in the test is. * The Peak Response Time exposes potential problematic resources or application anomalies. * For instance, it can help detect an expensive database query involved in fulfilling a certain request that could cause rendering of a page to take much longer. |
| **Throughput** | **Kilobytes Per Second** | * It represents the amount of data that the virtual users receive from the server at any given second. We can compare this graph to the response-time graph to see how the throughput affects transaction performance. |
| **Hits/second** | **Hits/second** | * The number of HTTP requests made by virtual users to the Web server during each second of the scenario run. |
| **Number of Passed/ Failed Transaction** | **Transaction** | * Total number of transactions Passed and Failed during the test execution |
| **Running Concurrent User** | **# of Concurrent Users** | * The number of concurrent users trying to access the Web Service at any instance in an interval of time |
| **Errors per second** | **Seconds** | * The average number of errors that occurred during each second of the run. It is to be expected that some errors may occur when processing requests, especially underload. Most of the time you will see errors begin to be reported when the load has reached a point that exceeds the web application's ability to deliver what is necessary |
| **90% percentile** | **Seconds** | * 90% times of the response time |

**Server side**

We will be collaborating with the GW Technical Lead to obtain the following metrics.

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Units of Measure** | **Description** |
| CPU & memory utilization | Web Server, Database server, Application Server | * The amount of CPU time & memory used by the Web Service while processing the request. |
| Network Latency | Network statistics | The time it takes from when the first request is sent until the first byte is received |

## **Infrastructure Requirements**

|  |  |  |
| --- | --- | --- |
| Infrastructure **Components** | **Purpose** | **Hardware Details** |
| JMeter | To create script, generate load execute | Standard Pekin IT VDI with Os Windows 10 |
| Datadog | Monitoring tool | Part of GCC |

## Environment:

The server configuration and the data consistency of the performance environment will be identical to the production server.

The database to be loaded to the performance environment would be a replica of production. Therefore, the volume and distribution of data to production will be achieved in the performance test environment.

|  |  |  |
| --- | --- | --- |
| **Environment Name** | **Environment Type** | **Purpose** |
| Performance Environment | Uplifted | Performance Testing |

**Risk: Masked Data Load and the Mitigation plan:**

Currently, GW lacks the capability to load masked data, resulting in the loading of only unmasked data. As most users access the applications using the SU account, the SU password is being changed in the performance environment. Only the test accounts utilized for performance testing will remain enabled, while all other accounts will be disabled.

## **Entry and Exit criteria**

**Entry criteria**

* Non- functional requirements are clearly defined and signed off, wherever applicable
* Functional testing has been completed for the features in scope of performance testing; test results are available in Jira for review
* Performance Test scenarios for features in scope have been developed and tested as individual script.
* Workload configuration and setup is done for running performance test.
* No outstanding Severity 1 and 2 defects, unless agreed upon by all stakeholders.
* The performance environment is ready.
* All appropriate Test data is available and ready to use or acquired in case of dependencies on other teams
* The scope is finalized and under change control.
* The testing resources, environments and tools are available to begin testing.

**Exit Criteria**

* Test phase execution completed (100% executed, >=90% pass)
* All severity 1 and 2 defects have been fixed unless agreed upon by Stakeholders.
* All non-functional performance requirements have been met.
* The stakeholders approve of any deviation from the performance testing requirements.
* All performance tests results are fully documented.
* All test cases/scenarios have been executed.
* Performance Testing Completion report has been published and signed off by appropriate stakeholders

## **Roles and Responsibilities**

|  |  |
| --- | --- |
| **Role** | **Description of a primary responsibility** |
| Test Manager | * Review and provide inputs for program level Performance approach * Provide and confirm the SLA’s * Planning co-ordination, logistics and support required for Performance across supporting projects and Guidewire core and adjacent stream * Ensure collaboration and communication among Performance testing teams and other development teams * Review Performance results and provide feedback |
| Performance Lead | * Write performance test plan * Identify the performance test cases * Create ramp up models for the test run * Co-ordinate with dev team, app admin team and DBA * Running the performance test * Create the schedule for performance testing * Create analysis reports of the performance testing |
| Performance Test Engineer | * Automate performance test scripts * Identify and prepare test data * Execute the performance tests * Create analysis reports * Identifies/logs defects |
| Solution Architect (GW) | * Define the nonfunctional requirements for the program * Resolve queries around nonfunctional requirement * Review performance test plan and key performance indices * Support in isolating key performance bottlenecks and resolution |

## **Performance practices to be followed:**

Performance Testing will adhere to the standards outlined in IS-TST-1101 Performance Testing Standards document.

* They are parameterized to enable different data sets.
* Script data values, such as names, addresses, and coverages, must be unique to each script to ensure the system does not access data cached by the application or database.
* They use multiple user IDs to prevent virtual users from using the same ID.
* They configure virtual users with the same roles used in production.
* They don't use the super user account, or any users with the super user role.
* Scripts are not configured to be executed as quickly as possible. Instead, they incorporate time delays that emulate actual user behavior in production.
* The test data is sufficiently varied to avoid two virtual users accessing the same entity simultaneously.
* The script sets transaction start and stops timers or flags to provide accurate metrics.
* Tests include realistic messaging. If the messages need to be mocked, they include realistic delays to simulate end-point processing.

## **Assumptions**

|  |
| --- |
| Assumptions |
| 1.Functional testing of application has been completed (at least one round) and there are no critical defects outstanding.  2. Nonfunctional requirements are clearly defined for key business transactions to be monitored during performance testing. In the absence of predefined nonfunctional requirements, baselines will be set up as actuals depending on product owner approvals.  3. Scrum Masters will plan capacity for performance defect triage fixes and any support during design and execution.  4. Appropriate training / support will be provided for performance testers on system functionality in scope.  5. A stable production-like environment is made available for creating performance test scripts and execution.  6. The configuration data (such as User roles etc.) is appropriately loaded, and the environment is ready for execution.  7. Support from other streams (supporting project QA) is available for any functional clarifications. |

## **Communication Plan:**

#### Interim updates:

Interim test results and findings will be shared with core technical team, including QA engineers, developers, Dev leads, test managers and DevOps/release managers, to support ongoing tuning and issue resolution.

#### Final Test Summary:

A consolidated performance test report will be shared with key stakeholders, including the product management teams, Business Owners, and IT leadership. This report will outline overall performance metrics, any identified risks, and recommendations.

## **Communication Channels:**

Updates will be shared via email and during scheduled meetings.

## **Requirements:**





## **Test Execution Flow:**

Execute the Load Test on Jmeter

Discuss the Failure with the Development / Infra Team

Pass

Execution Status

Successful Run

Analyze the Failure

Script

Issue

Fail

Rectify the Performance Script

Yes

No

New / Reopen Defect

Existing Defect

Raise a new defect / Update an existing defect