

**Performance Test**

**Execution Report**

**SCUBA - External Positions Repository**

**Program / Project No.: 0354**

**Document Version: 1.10/2020-01-30**

**Template Version: 1.20 (2015-10-14)**

***Prepared By:***

**Department: Quality Assurance**

**Group: Technical QA**

RBC Unit: RBC I&TS

© Royal Bank of Canada 2006 - January 2020: All Rights Reserved. Restricted / Confidential / RBC Internal: The information contained in this document is the property of the RBC Financial Group. Except as specifically authorized in writing by the RBC Financial Group, the holder of this document shall keep the information contained herein confidential and shall protect same in whole or in part from disclosure or dissemination to third parties.

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Version Number** | **Date** | **Author** | **Description** |
| 1.0 | 2020-01-29 | Vaibhav Musale | Test Execution Report |
| 1.10 | 2020-01-30 | Vaibhav Musale | Added the schematic representation to clear the difference between queues for message injection delay. Also modified detailed observations based on discussions with the dev team |

distribution

|  |  |  |
| --- | --- | --- |
| **Date** | **Name** | Title |
| 2020-01-29 | **Jean-Christophe FIGUIERE** | App Owner |
| 2020-01-29 | **Alex Lee Figuiere** | Senior Manager |
| 2020-01-29 | **Martial Urbanc** | Delivery Stream Lead |
| 2020-01-29 | Sophie Bannier | Business Owner |
| 2020-01-29 | Christelle Gerard | SCUBA IT Project Manager |
| 2020-01-29 | Fatiha Bekkar | Senior Manager, QE/PE & Technical Services |
| 2020-01-29 | Dinesh Patil | QA Lead |
| 2020-01-29 | Pierre Delatte | Program Manager, Scuba |
| 2020-01-29 | Dimitri Devriendt | Developer |
| 2020-01-29 | Cedric Leburton | Developer |
| 2020-01-29 | Nicolas Viel | Developer |
| 2020-01-29 | Vaibhav Musale | Performance Test Lead |
| 2020-01-29 | Elena Frenkel | DBA |

ReFERENCE DOCUMENTS

The following provides a list of documents that are useful to the creation and review of this Performance Test Execution Report

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | **Date** | **Author** | Title |
| 1.02 | 2019-12-18 | Vaibhav Musale | [SCUBA – EPR Performance Test Plan Dec2019\_v1.02.docx](https://spcollab.fg.rbc.com/team/H30_ACE2/scuba/BLS_PgmMgmt/Phase%202%20Delivery%20Streams/EPR/QA/Performance%20Test/Dec%202019/1%20-%20Docs/SCUBA%20-%20EPR%20Performance%20Test%20Plan%20Dec2019_v1.02.docx) |
| 0.0 | 2020-01-29 | Sumedh Dabhade | Email message for the pictorial representation of message injections |

TABLE OF CONTENTS

[1 Introduction 4](#_Toc31272575)

[2 Executive Summary 5](#_Toc31272576)

[2.1 Results Summary 5](#_Toc31272577)

[2.2 DEFECT Summary 5](#_Toc31272578)

[2.3 Conclusion 5](#_Toc31272579)

[3 test Results 6](#_Toc31272580)

[3.1 STRESS1 – GOAL1 6](#_Toc31272581)

[3.1.1 Scenario Details 6](#_Toc31272582)

[3.1.2 Execution summary 6](#_Toc31272583)

[3.2 STRESS2 – GOAL2 7](#_Toc31272584)

[3.2.1 Scenario Details 7](#_Toc31272585)

[3.2.2 Execution summary 7](#_Toc31272586)

[3.3 GOAL3 8](#_Toc31272587)

[3.3.1 Scenario Details 8](#_Toc31272588)

[3.3.2 Execution summary 8](#_Toc31272589)

[3.4 GOAL4 8](#_Toc31272590)

[3.4.1 Scenario Details 8](#_Toc31272591)

[3.4.2 Execution summary 8](#_Toc31272592)

[4 Performance Test Result Sign-Off 10](#_Toc31272593)

# Introduction

This Performance Execution Report document provides the details in regard to the readiness of this project for the next phase of testing/deployment. Sign-off by all parties involved is required before the project can be released from Performance testing to the next phase of testing/deployment, in accordance with the standards and practices defined by the RBC SDLC.

The versions on which we have executed the performance test are listed as below:

|  |  |
| --- | --- |
| **Micro Applications** | **Version** |
| rbc-epr-dic-service | master-TWU0\_rbc-epr-dict-service-v33 |
| rbc-epr-zuul-proxy | cs1.3-TWU0\_rbc-epr-zuul-proxy-v2.9 |
| rbc-epr-ui | CS1.3-TWU0\_rbc-epr-ui-v57 |
| rbc-epr-security-static-jms-interface | CS1.3-TWU0\_rbc-epr-security-static-jms-interface-v3 |
| rbc-epr-transaction-jms-interface | CS1.3-TWU0\_rbc-epr-transaction-jms-interface-v56 |
| rbc-epr-business-date-service | master-TWU0\_rbc-epr-business-date-service-v25 |
| rbc-epr-business-logic | cs1.3-TWU0\_rbc-epr-business-logic-v210 |
| rbc-epr-eod-job-trigger-service | CS1.3-TWU0\_rbc-epr-eod-job-trigger-service-v3 |

This Performance Execution artefact:

* Provides an overview of the performance test results
* Presents the details on performance test execution conditions and outcome of each performance test scenario
* Offers recommendations based on the performance test execution results

# Executive Summary

## Results Summary

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Expected Result** | **Actual Result** |
| GOAL1 | **Daily** peak CASH SWIFT messages (volume 500) injected at EPS endpoint should be processed and seen in the EPR front end application | * Messages processed and seen in EPR application in Verified state. |
| GOAL2 | All screens browsed by the script should not show performance slowness | * No distinguishing slowness observed on any screens |
| GOAL3 | Prove that EPR MS SQL DB is performant in processing and storing data | * No issues reported on the database |
| GOAL4 | Validate to confirm there is no degradation in response times after merging 9 micro services to 5 micro services as compared to results in the baseline test document dated 26th July 2019 [Performance Test Execution Report EPR.docx](https://spcollab.fg.rbc.com/team/H30_ACE2/scuba/BLS_PgmMgmt/Phase%202%20Delivery%20Streams/EPR/QA/Performance%20Test/July%202019/5%20-%20Results/Performance%20Test%20Execution%20Report%20EPR.docx) | * There was increase in the response time. However, not distinguishable as all the values were well below 1 sec except for the Login transaction which is on the SSO server and not in control of the EPR dev team |

## DEFECT Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Defect #** | **Severity** | **Status** | **Defect Description** | **Recommendation / Action Plan** |
|  |  |  |  |  |  |

**Final Defect Count Status**

|  |  |
| --- | --- |
| **Status** | **Total Count** |
| Closed | 0 |
| Deferred | 0 |
| Monitored | 0 |
| Total Count | 0 |

## Conclusion

During the performance test, the front end UI transactions did not report any distinguishable slowness.

The daily peak messages (500) injected in one go from Milvus end point to EPR end point to come into verified state in the EPR application took around 5 min. This will be treated as a new baseline for message injections.

At the database, we did not see any degradations.

For more details on this, please refer [section 3](#_test_Results)

# test Results

## STRESS1 – GOAL1

### *Scenario Details*

|  |  |
| --- | --- |
| **Scenario Details** | |
| Test Scenario Description | 500 SWIFT messages injected into **QTWU0.EPREPS.TO.ESB** |
| Performance Goal/Acceptance Criteria | Prove that overall EPR system can process peak SWIFT transaction load received from EPR endpoint |
| Assumptions, Configurations and Prerequisites | Messages to reach the EPR application |
| Test Result Summary | Processing of these 500 injected messages was successful. All the messages reached the EPR application in 5min 09 sec.  For more details on this, please refer section 3.1.2 Execution Summary |

### *Execution summary*

select count(\*) from cash\_transaction where trade\_date = '2020-01-24' and remarks = 'Data Load' and currency\_code = 'EUR' and creation\_date BETWEEN ‘24-01-2020 11:34:15' and ' 24-01-2020 11:45:00'

Database extract in attached excel sheet shows the messages as received in the EPR application



Total time required for 500 messages to move from Milvus end point to EPR end point 🡪 EPR application(verified state) is **5 min 09sec**.

## STRESS2 – GOAL2

### *Scenario Details*

|  |  |
| --- | --- |
| **Scenario Details** | |
| Test Scenario Description | 10 users browsing screens of EPR application CASH stream for 1 hour   |  | | --- | | A\_LandingPage | | B\_LogIn | | C\_ViewMovements | | D\_ViewPositions | | E\_PositionsByCurrencyFilter | | F\_Movement\_fromPositionScreen | | G\_ViewTransaction | | H\_ViewTransaction\_DateFilter | |
| Performance Goal/Acceptance Criteria | 1 hour steady state execution to be analyzed and shared for response times with no delay in response time |
| Assumptions, Configurations and Prerequisites | Scenario details as below  User ramp up: 1 user every 30 sec  Think time: 3 sec  Test duration: 1 hour  Iteration criteria: start new iteration after every 120 sec  User ramp down: 2 users every 30 sec  500 messages to be injected once into **QTWU0.EPREPS.TO.ESB** after all 10 users are logged on to the application. |
| Test Result Summary | Response time for all the above said screens did not see any performance bottleneck |

Note: For transactions having intermittent spikes of high response times, if the spikes are between 5 to 10%, they have been clipped off as a part of analysis.

### *Execution summary*

Response times for the 1 hour steady state execution is as below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Transactions** | **Min**  **(sec)** | **Avg**  **(sec)** | **Max**  **(sec)** | **Std. Deviation** | **90 Percent (sec)** | **Pass** | **Fail** |
| A\_LandingPage | 0.57 | 0.83 | 1.28 | 0.15 | 1.06 | 156 | 0 |
| B\_LogIn | 12.29 | 13.1 | 14.38 | 0.34 | 13.53 | 156 | 0 |
| C\_ViewMovements | 0.36 | 0.48 | 0.89 | 0.10 | 0.64 | 155 | 0 |
| D\_ViewPositions | 0.31 | 0.43 | 0.70 | 0.1 | 0.57 | 154 | 0 |
| E\_PositionsByCurrencyFilter | 0.18 | 0.38 | 1.44 | 0.22 | 0.64 | 154 | 1 |
| F\_Movement\_fromPositionScreen | 0.54 | 0.74 | 1.14 | 0.11 | 0.9 | 154 | 0 |
| G\_ViewTransaction | 0.31 | 0.45 | 0.91 | 0.13 | 0.64 | 154 | 0 |
| H\_ViewTransaction\_DateFilter | 0.30 | 0.53 | 2.71 | 0.31 | 0.8 | 152 | 1 |

## GOAL3

### *Scenario Details*

During the message injection of 500 sample messages and the execution of 1 hour of UI performance test, the database needs to be monitored for performance issues, if any.

Database trace has been scheduled to capture the transactions.

### *Execution summary*

No performance issues observed during the run as confirmed by the dba.

Observatoins by DBA attached as below



## GOAL4

### *Scenario Details*

For the same message injection of 500 samples (day volume) and UI load test, this goal is to compare the test runs between July 2019 & Jan 2020 runs.

### *Execution summary*

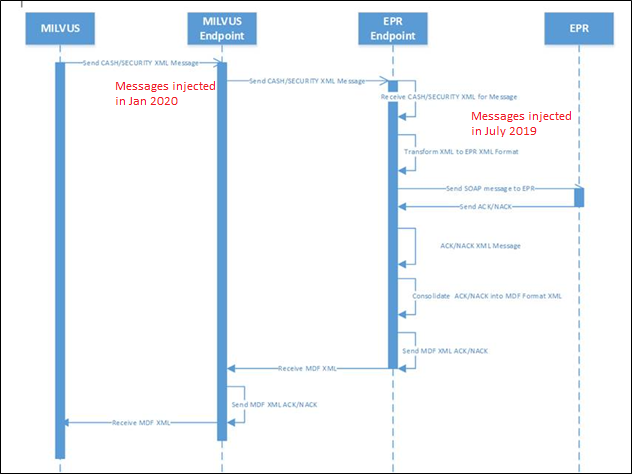
UI Observations: From the two runs, the transactions which were similar have been compared as below. We observed an increase in response time in each of the transactions. However, none of the transactions took more than 1 sec each. The delay observed in the Login transactions is on the SSO server which as confirmed by the dev team is not in their control to fine tune.

|  |  |  |
| --- | --- | --- |
| **Transaction Name** | **Jul'19** | **Jan'20** |
| A\_LandingPage | 2.61 sec | 0.83 sec |
| B\_LogIn | 9.15 sec | 13.1 sec |
| C\_ViewMovements | 0.30 sec | 0.48 sec |
| D\_ViewPositions | 0.25 sec | 0.43 sec |
| E\_ViewTransactions | 0.17 sec | 0.45 sec |

Message injections: We observed some delay in the messages reaching to EPR application when compared to the July 2019 run.

The primary reason for this being that the queue QTWU0.IN.CASH (QTWU0.EPREPS.TO.EPR.MILVUS.CASH) in which the injection was done in July 2019 was directly in to the EPR application.

Whereas during this Jan 2020 run, the messages have been injected into QTWU0.EPREPS.TO.ESB as the QTWU0.EPREPS.TO.EPR.MILVUS.CASH was not available in the SOA Management tool from where we inject messages. When we inject messages into this queue, it traverses through MilvusEndpoint, EPR Endpoint and then into the EPR application in verified state. Hence this additional delay has been observed. This 5min should be considered as new baseline.



**Note**: This execution has been done via HP Performance center. The scripts, test scenario and result files are stored in Performance center.

Result IDs used for analysis are 4833 & 4836.

# Performance Test Result Sign-Off

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Sign Off** | **Date** |
| **App Custodian** | Jean-Christophe FIGUIERE |  |  |
| **Delivery Stream Manager** | **Urbanc Martial** |  |  |
| **Senior Manager** | **Alex Lee** |  |  |
| **Business Owner** | **Sophie Bannier** |  |  |
| **Sr. Manager, QE/PE** | **Fatiha Bekkar** |  |  |
| **Project Manager** | Christelle Gerard |  |  |

|  |  |  |
| --- | --- | --- |
| **Type** | **Purpose** | **Notes** |
| Performance test | To determine or validate speed, scalability, and/or stability. | A performance test is a technical investigation done to determine or validate the responsiveness, speed, scalability, and/or stability characteristics of the product under test. |
| Volume test | To verify application behavior under normal and peak load conditions. | Volume 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 volume test enables you to measure response times, throughput rates, and resource-utilization levels, and to identify your application’s breaking point, assuming that the breaking point occurs below the peak load condition. |
| Stress test | To determine or validate an application’s behavior when it is pushed beyond normal or peak load conditions. | The goal of stress testing is to reveal application bugs that surface only under high load conditions. These bugs can include such things as synchronization issues, race conditions, and memory leaks. Stress testing enables you to identify your application’s weak points, and shows how the application behaves under extreme load conditions. |
| Availability and Reliability test | To determine the sustainability an application over a period of time and under a defined workload. | An Availability and reliability test is a type of performance test focused on determining or validating the performance characteristics of the product under test when subjected to workload models and load volumes anticipated during production operations over an extended period of time.  Availability and reliability testing may be used to calculate Mean Time Between Failure (MTBF), Mean Time To Failure (MTTF), and similar metrics. |
| Scalability test | To determine how many users and/or transactions a given system will support and still meet performance goals. | Scalability testing is conducted 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 you to identify a scaling strategy in order to determine whether you should scale up or scale out. |