Technical Design Document

Posting QE Test Results into HPQC

Document Version: 1.1

Summary of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description | Created by | Date |
| 1.0 | Initial Draft | Jayapal.K | 04/14/2012 |
| 1.1 | - Renamed class names.  - Added/updated use case/ sequence/class diagrams. | Jayapal.K | 05/31/2012 |

Table of Contents

[1 Introduction 4](#_Toc326343235)

[1.1 Purpose 4](#_Toc326343236)

[1.2 Audience 4](#_Toc326343237)

[1.3 Definitions 4](#_Toc326343238)

[2 System Architecture Overview 5](#_Toc326343239)

[2.1 System components 5](#_Toc326343240)

[3 Diagrams 7](#_Toc326343241)

[3.1 Use case realizations 7](#_Toc326343242)

[3.2 Sequence Diagrams 7](#_Toc326343243)

[3.2.1 Online result posting 7](#_Toc326343244)

[3.2.2 Offline result posting 8](#_Toc326343245)

[3.3 Class Diagrams 9](#_Toc326343246)

[3.3.1 Online & offline result posting 9](#_Toc326343247)

[4 Benefits 11](#_Toc326343248)

[5 References 12](#_Toc326343249)

# Introduction

## Purpose

Different QE Teams are using different automation frameworks to execute QE tests and update the results in different format & different file systems according to their requirements. All QE teams will be migrating to HPQC that is a centralized repository to store all QE results and each team needs an interface to post the results into HPQC from the automation framework.

This technical design document proposes a common connector to integrate a test automation framework into HPQC to post the results and also provide an interface to upload the intermittent results into HPQC.

## Audience

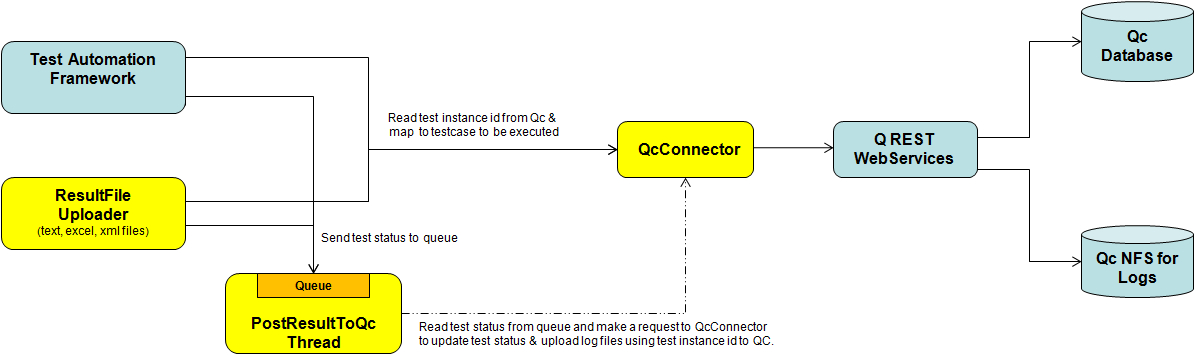
This document is intended for QE teams who want to post test results and upload the log files directly into HPQC without any manual intervention.

## Definitions

|  |  |
| --- | --- |
| HPQC | HP Quality Center is a test management tool. |
| Test Lab | It is a HPQC module where user executes the test cases and stores the results. |
| Test Instance | A test case imported from Test Plan to Test Lab module is called an instance of the test case. It is possible to have multiple instances of the same test case. |
| Test Run | When a test instance is run every time, test instance has a new row to store status, start date & time, duration, etc for that run in Test Lab so that user can see the test execution status for that run at any point of time. |
| Test Set | It is a collection of test instances grouped together in Test Lab. |
| Test Suite | It is a collection of tests grouped together. This term is used in automation framework. |
| Test/TestInstance/TestSet/TestRun Id | It is a unique id to identify a specific Test/TestInstance/TestSet/ TestRun in HPQC. |

# System Architecture Overview

The diagram below provides an illustration of the system architecture along with various system components that will be used in architecting QC Connector framework.



## System components

Interaction of software components along with its responsibilities is explained below.

**Qc REST Web Services** - A set of API REST Web services are available for test automation frameworks to read test instances data from Quality Center as well as post test results using test instance IDs back to Quality Center

**Qc Database & NFS** – Database is a repository to store Test plans, TDSs, test results, etc and NFS for storing log & screenshot files.

**QcConnector** - This is a core component and directly interacts with REST web services to read/create/update information related to test into QC. It acts as a bridge between QC and other components and does the following for any QC functionality that is requested.

* Transfer the input parameters into HTTP request.
* Send the http request to relevant web services and receives response.
* Convert response back to components in a way they understand.

**PostResultToQcThread** - This component maintains an queue through which test results are added and reads test status from queue and posts the status & uploads log files into QC. It runs in separate thread.

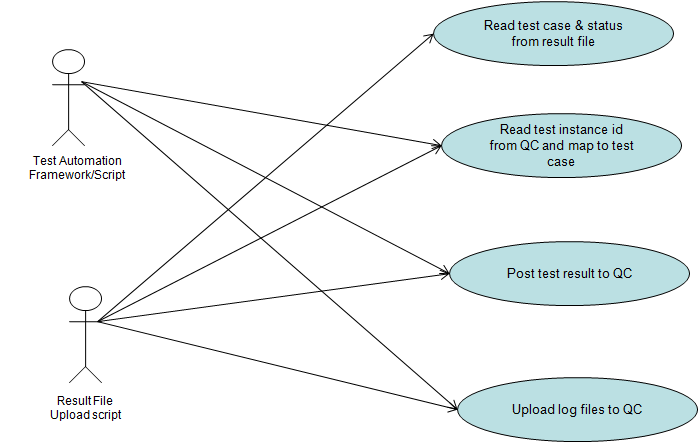
**Test Automation Framework** - Executes QE tests in automated fashion and sends test status to Queue when a test completes and immediately goes to next test. It doesn’t wait for test status to be processed into QC that saves reasonable time.

**ResultFile Uploader** - It is an offline task component that reads a test result file(text, excel, xml), finds the result rows and posts the results into QC using QcConnector & PostResultToQcThread components.

# Diagrams

## Use case realizations

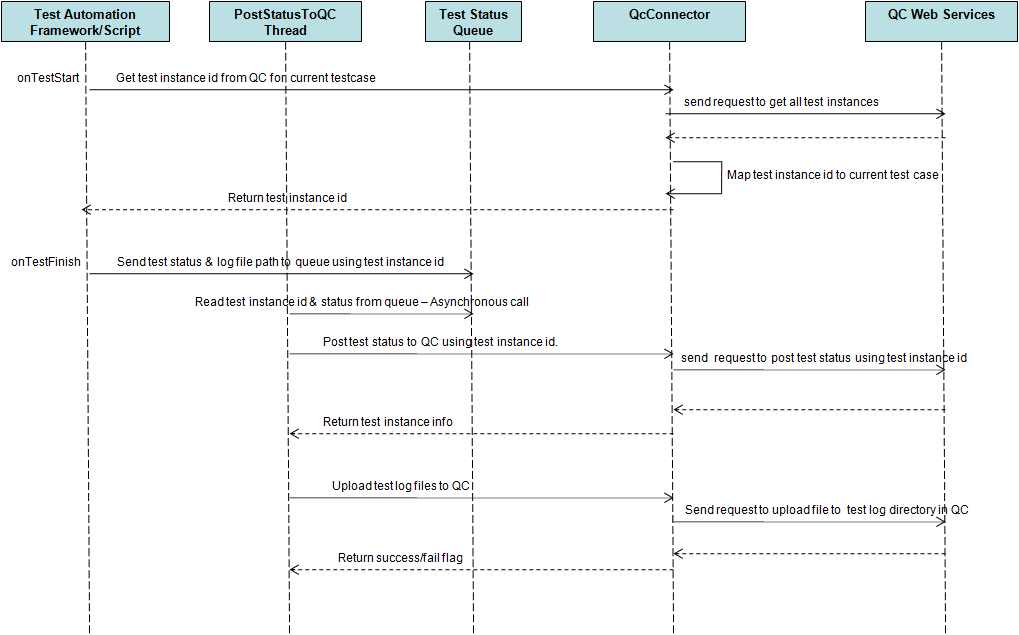
This diagram represents use case scenarios for i) test framework and ii) offline result upload script having goals as related to posting results into QC.



## Sequence Diagrams

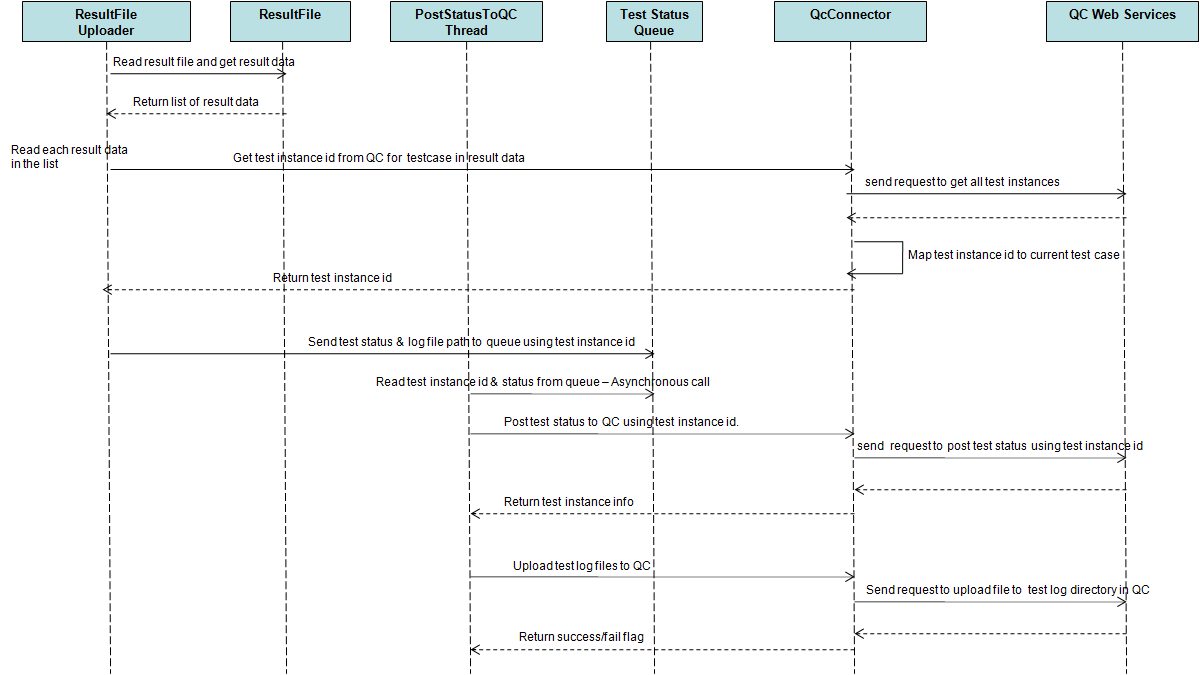
### Online result posting

This diagram shows interactions between components in the sequential order on how test result is posted from test automation framework to QC.



### Offline result posting

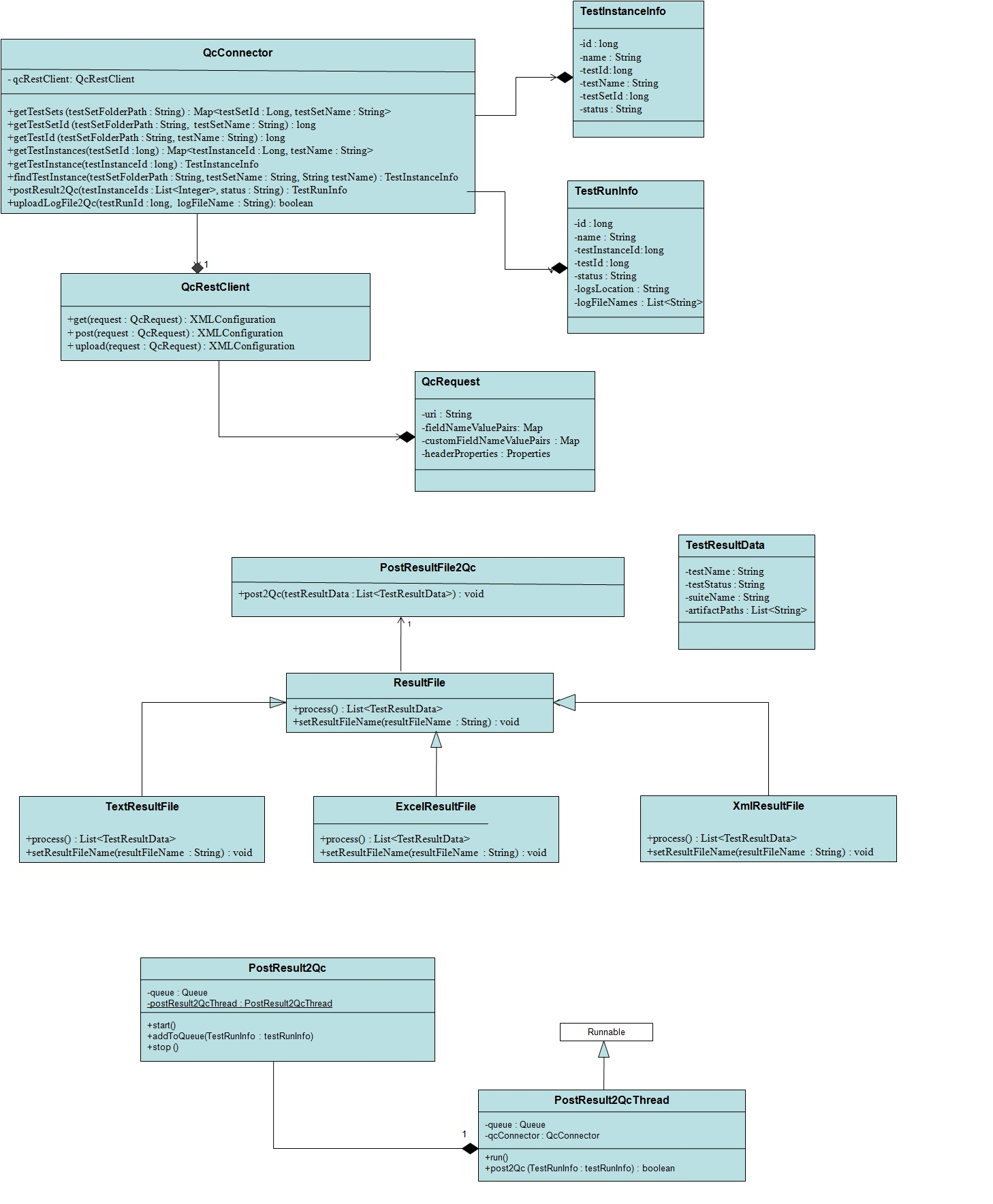
This diagram show interactions between components on how test result is posted from offline result upload script to QC.



## Class Diagrams

### Online & offline result posting

The diagram below describes the structure of QC connector framework by showing classes, their attributes, methods & the relationships among the classes.



QcRestClient – This class makes call directly to REST webservices API to access/store information related to test into QC. It contains methods like get, put & delete to have clear separation of REST specific call.

QcConnector – This class is an entry point for a client to perform any operations as related to create/update test information into QC. It makes use of QcRestClient by invoking appropriate get/put methods.

TestInstanceInfo, TestRunInfo and TestResultData – These are plain classes that have test related attributes and used to exchange data between classes.

For online result upload (test framework to QC),

* When framework starts a test suite(containing list of tests to be executed), it reads all test instances associated with the test suite from QC using QcConnector class and caches them.
* When a test in the suite starts, the framework maps the test to testinstance id from the list of test instances.
* When execution of the test is complete, the framework sends test status, instance id & log file to Queue using PostResult2Qc class and will go to next test without need of waiting for result to process into QC.
* PostResult2QcThread class monitors the queue. As on when a test status is added to the queue and it reads the result from queue and post the results into QC.

For offline result upload,

* ResultFile is used as interface having process method that returns list of ResultFieldData.
* TextResultFile, ExcelResultFile and XmlResultFiles are implementations of ResultFile. The process method has to parse the result log file and return as list of ResultFieldData.
* PostResultFile2Qc is the entry point and uses process method of ResultFile to process the log and then finally it will post the results using PostResult2Qc class by invoking start, addToQueue & stop methods.

# Benefits

There are some benefits from this framework.

- Eliminate QIR.

- Having common framework for different QE teams especially using Java based framework.

- No/Minimal design change required to integrate with a test framework.

- Asynchronous result posting and log file upload, i.e, test framework need not to wait for test results to be processed into QC and saves execution time.

# References

<https://wiki.eng.vmware.com/QEOps/InfTools/HPQC-WebServices>