

# How do we do automated testing

## Table of Contents

[Credits](#)

[Overview](#)

[Automate test case](#)

[Test case Libraries](#)

[Test case organization](#)

[Test cases delivery](#)

[Automate test process](#)

[Test case management – Product Studio, TCMRegister](#)

[Test harness – Perseus](#)

[Check-in validation system – CVS](#)

[Test daily run system](#)

[Test report](#)

[Glossary](#)

## Credits

|                            |                  |
|----------------------------|------------------|
| <b>Author:</b>             | <b>Song Xu</b>   |
| <b>Technical Reviewer:</b> | <b>Haiyun He</b> |
| <b>UA Reviewer:</b>        | <b>John Dong</b> |

## Overview

Automated testing is a widely adopted practice in Microsoft. In HMC team, 70% test cases are automated. It saves much time for us to do regression tests and run test pass at regular intervals.

Automated testing is not only about how to automate test cases, but also about automating the test process which includes test case management, test case execution, test results analysis, test report generation, and building a daily-run system. The HMC test team's work covers all these areas. What's more, we also develop some separate testing tools to improve our engineering process.

There is no free lunch with no exception to automated testing – it needs additional time to develop and maintain test code, let alone that not all the test cases can be automated. So we (test team) will get early involved in the project, and investigate what has to be automated, and review developers' design spec to make sure that the design is testable. Afterwards, we will work out a test spec to finalize our test methodology in this project.

## Automate test case

After we decide to use automated testing, the first thing is to automate the test cases we designed.

The most important part of HMC test cases is to set up the environment and do back-end verification. So we developed C# libraries to access resources, such as Active Directory, Exchange, and SharePoint. Because the HMC test environment includes several servers which the test cases need to access, we use Dot Net Remoting technology in our libraries.

For module independence, we have two levels of library. The low level one called Olympus focuses on non-HMC related operations, and the high level one called CommonHelper is a kind of wrapper of MPS requests, which can be used to easily generate and send a MPS request.

For some UI related test cases, we use Microsoft internal UI automation library such as KAF, or Mita. And our principal is to access resource directly instead of using UI automation as much as we can, because UI is more prone to change.

All HMC test cases will be run by Perseus (a test harness), so the test cases need meet the requirements of Perseus test case (such as return type of test case function).

## Test case Libraries

### Olympus

Olympus library uses Remoting technology. Before running test cases, Olympus.Remote components will be installed on all the servers which test cases want to access, running as windows services. The request to access remote resource will be transferred from the host server on which test cases are running to the destination server.

Olympus library can also log and parse parameter xml which is used by test cases.

### Common Helper

Common helper wraps different level of MPS requests as C# class and functions for test cases to invoke easily. We have different commonhelper for different HMC versions. For HMC4.5, our common helper is called BogongCommonHelper.

### UI automation library

Microsoft has many excellent UI automation libraries for internal use. We are using KAF which is at first used by Windows Live team. Some of our scenario test cases need to take advantage of UI automation.

## Test case organization

We group test cases into several levels. Firstly, we have test modules, and each module has several test suites, and lastly, each test suite has several test cases. For example, Hosted exchange namespace maps to a test module, and each procedure of hosted exchange namespace maps to a test suite, and test cases for a procedure will live in a corresponding test suite. The following list shows an example.

Hosted Exchange namespace (test module)

    {CreateMailbox} (test suite)

        CrtMB\_AllParamsValid\_Pos\_aRA (test case)

CrtMB\_MandatoryParamsValid\_Pos (test case)

{DeleteMailbox} (test suite)

In most cases, test module has its module setup and cleanup, and test suite has its suite setup and cleanup.

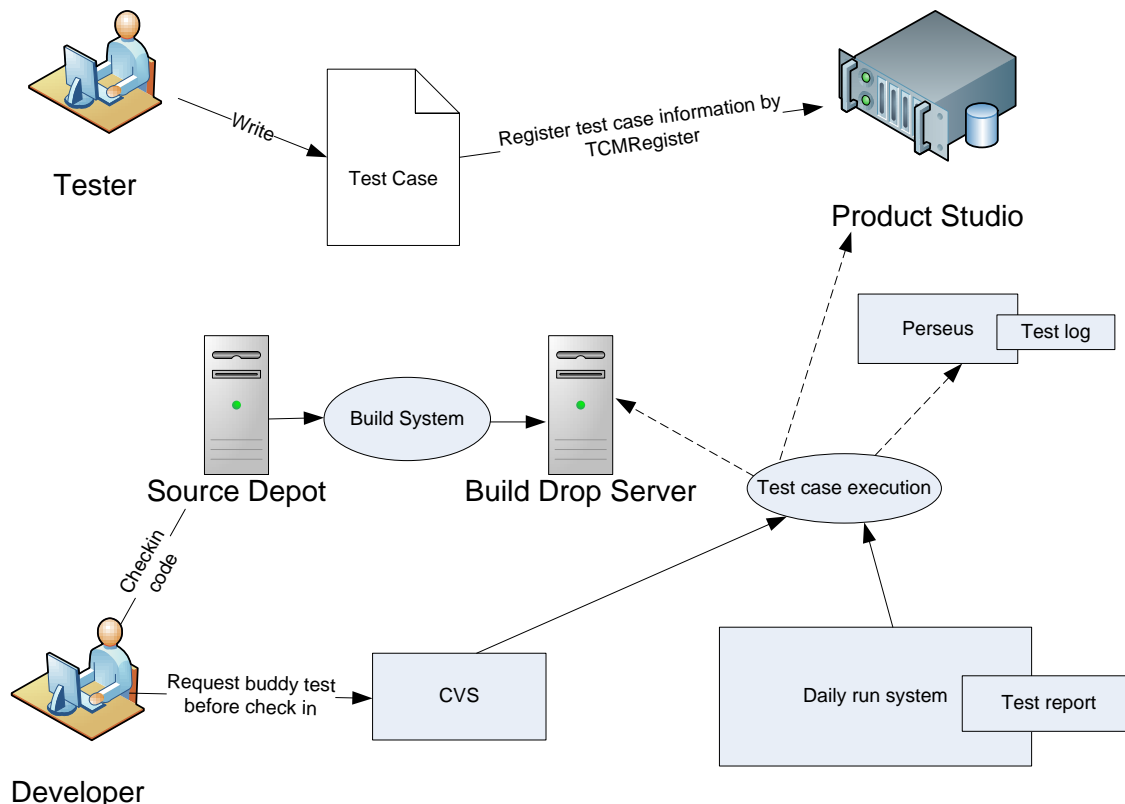
## Test cases delivery

All test cases are built into a single MSI by the daily build system. So we only need to install the MSI on test server, and then all test cases are there.

## Automate test process

Now test cases are automated. But there is still much work for testers to do. We need to manage the test cases, execute the test cases, generate test report, and archive test result. We try to make all these processes automated as much as we can to ease our daily work and improve our engineering excellence.

The following picture shows the main testing process in HMC team.



## Test case management – Product Studio, TCMRegister

There are about 20 thousand test cases to run before we ship a HMC product. So how to manage these test cases efficiently is very important. It should be easy to retrieve specified set of test cases, such as all Exchange test cases, all Priority 0 test cases. In HMC, we use Product Studio (PS) to store the information of test cases. All properties of test case such as priority, title, description will be marked as C# attributes, and

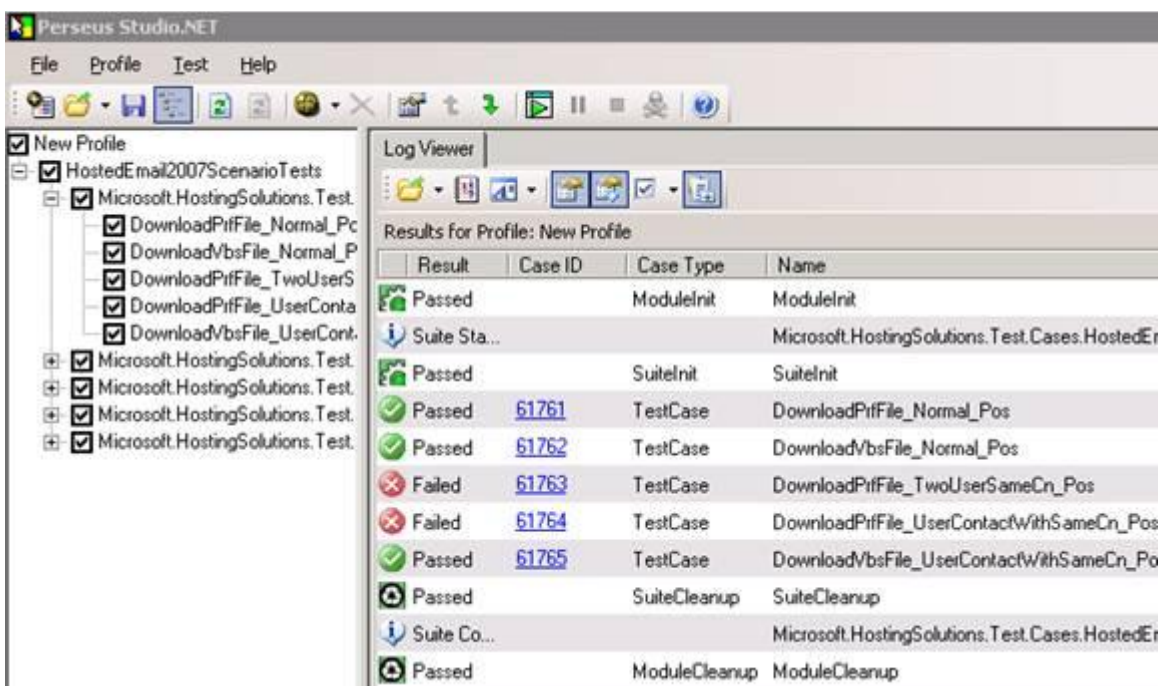
then will be uploaded to Product Studio by TCMRegister tool (a Microsoft internal tool). After that, we can use PS client tool to query test cases by providing the PS query statement.

## Test harness – Perseus

Now it is time to execute our test cases. HMC team uses Perseus as test harness, which is an internal tool from Exchange team. Perseus can run different type of test cases, such as Dot Net assembly, Windows Script Host file. The HMC test cases are all of type Dot Net assembly.

Perseus will load a test case configuration file which represents the test cases to be executed, (we have another tool to generate the configuration file from PS query file), and then we can select some or all the test cases to run. When test running is complete, you can read the log from Perseus client.

Here is a snapshot of Perseus.



## Check-in validation system – CVS

Some of test cases are marked as BVT (build verification tests), which is a small set of test cases yet with high priority. We use these test cases to quickly verify the basic functionalities of the product. In HMC every code check-in is required to pass BVT test cases before checking in.

To automate this process and run BVT tests, we developed the check-in validation system, which can help developers and testers to do buddy build and BVT test automatically. It can greatly improve the check-in quality and work efficiency.

Here is a snapshot of web client of CVS.

# Checkin Validation System

Welcome! FAREAST\songxu

Project:

Buddy Build

|                      |   |  |
|----------------------|---|--|
| <b>SD package:</b>   | <input type="text"/>  | <input type="button" value="Browse..."/> |
| <b>Branch:</b>       | HMC_Bogong  |  |
| <b>Build Path:</b>   | <input checked="" type="radio"/> Max <input type="radio"/> Min <input type="radio"/> Dev <input type="radio"/> Test <input type="radio"/> Specify |  |
|                      | Specify: \\depot\HMC_Bogong \ <input type="text"/>  |  |
| <b>Build flavor:</b> | <input checked="" type="radio"/> Debug <input type="radio"/> Release  |  |

Buddy Test

|                     |  |
|---------------------|--|
| <b>Dev binary:</b>  | <input type="text" value="4.5.246.0"/>                                 |
| <b>Test binary:</b> | <input type="text" value="Buddy Build"/>                               |
| <b>PSQ file:</b>    | <input checked="" type="radio"/> BVT <input type="radio"/> Specify     |
|                     | Specify: <input type="text"/> <input type="button" value="Browse..."/> |

## Test daily run system

After test code is complete, we will run daily test pass against every official build. The process is that dev team checks in code and then builder team build official build and then test team runs test pass against the new build.

To automate some part of the process, we developed the test daily run system. The system can manage lots of physical test servers, run test job by schedule on target server, and have a well-formed test report. It will pick up latest or given official build, prepare test environment (restore and start virtual images), install the product build, install the test build, execute given test cases, generate test report, and send the report via e-mail to subscribers.

## Test report

Our test report is built into test daily run system. It is a web-based report, which shows statistic information, such as the pass rate. Also it can show the detail execution log of specified test cases. User can easily retrieve the test result by date, test owner or test module on web client.

The following tables show an example test report:

| HMC 4.5 - Daily Tests - 4.5.228.0 |                |
|-----------------------------------|----------------|
| <b>Product</b>                    | HMC 4.5        |
| <b>Run Time</b>                   | 1.10:37:48     |
| <b>Build</b>                      | 04.05.0228.000 |
| <b>Pass Rate</b>                  | 100.00 %       |

## Components Test Results

| Component Name                                      | Total | Grand Total | Passed | Failed | Pass Rate | Net Pass Rate |
|---|-------|-------------|--------|--------|-----------|---------------|
| <b>Active Directory</b>                             | 2329  | 2367        | 2329   | 0      | 100.00 %  | 98.39 %       |
| <a href="#">Active Directory Provider Namespace</a> | 483   | 483         | 483    | 0      | 100.00 %  | 100.00 %      |
| <a href="#">Hosted Active Directory Namespace</a>   | 497   | 502         | 497    | 0      | 100.00 %  | 99.00 %       |
| <a href="#">Hosted Active Directory Webservice</a>  | 481   | 492         | 481    | 0      | 100.00 %  | 97.76 %       |
| <a href="#">Managed Active Directory Namespace</a>  | 463   | 483         | 463    | 0      | 100.00 %  | 95.86 %       |
| <a href="#">Managed Active Directory Webservice</a> | 405   | 407         | 405    | 0      | 100.00 %  | 99.51 %       |

## Glossary

| Name         | Description  |
|--------------|--|
| HMC          | Hosted Messaging and Collaboration   |
| AD           | Active directory   |
| MPS          | Microsoft Provisioning System  |
| Test harness | A collection of software which has a test execution engine to execute a set of test cases. |
| Olympus      | A test library developed and used by HMC team  |
| Perseus      | A test harness developed by Exchange test team   |