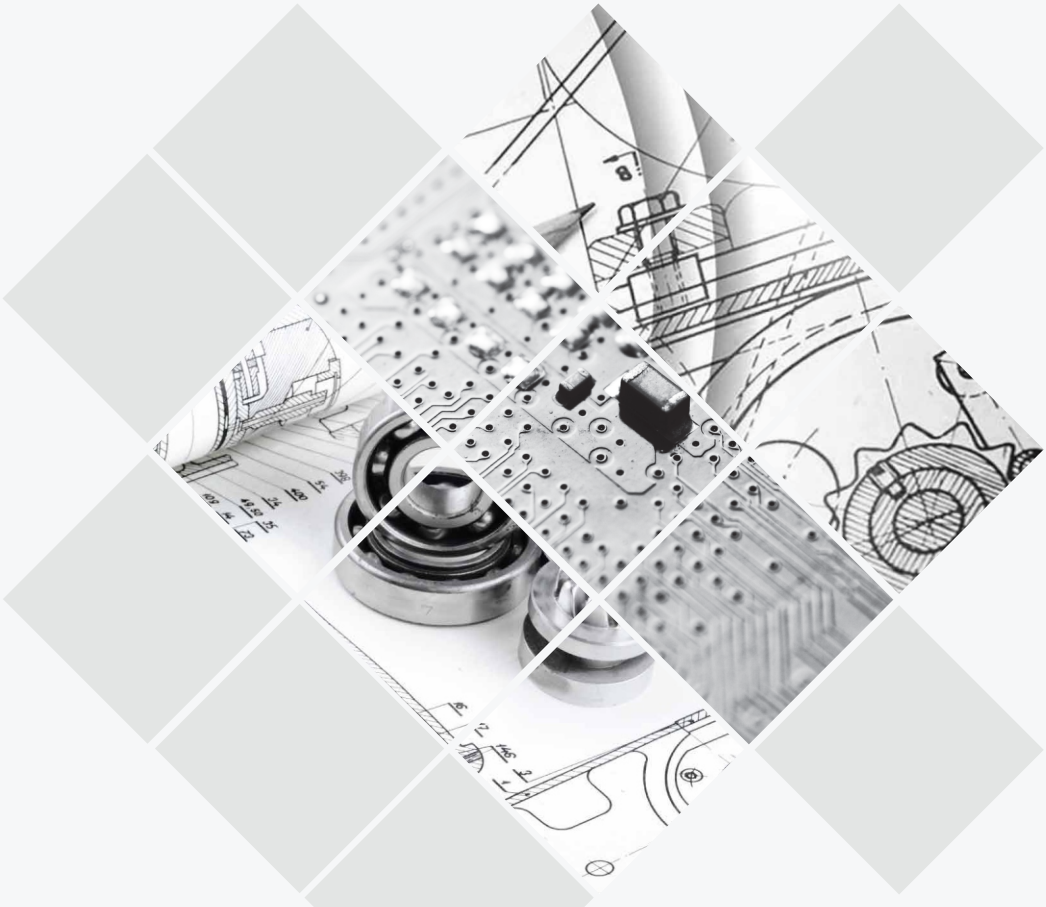
QUEST

BORN TO ENGINEER

# FaSaT

## an interoperable test automation solution

**Flexi any Script any Tool (FaSaT is a test automation framework which provides interoperability among multiple test automation tools and multiple scripting languages.**



**Rejeesh Gopalakrishnan**
**Ajith Michael**
**Dileep K**
**Manjulakshmi C S**
**Ashly Kurian**

QuEST Global

# contents

# FaSaT An Interoperable Test Automation Solution


## Abstract

In software industry, test automation is a key solution for achieving volume verification and validation with optimal costs. Picking up the right automation tool and underlying scripting language has always been a challenge, balancing between cost factors and team's expertise levels in various tools and scripting languages. A real solution would be one that allows full flexibility for team on these two core concern areas – test automation tool and scripting language.

Flexi any Script any Tool (FaSaT) is a test automation framework which provides interoperability among multiple test automation tools and multiple scripting languages. To support multiple scripting languages and test automation tools, the framework provides plug-ins for standard languages and tools. Feature extended for scripting languages that are not originally supported by test automation tool. The scripting style hides underlying test automation tool details.

# FaSaT An Interoperable Test Automation Solution

## Introduction

### Why Test Automation

Have we neglected detailed testing in software products because of time and cost constraints or have we given less weightage for test automation because of these constraints. Majority of the managers are forced to give less weightage for software testing because of these factors. Effective software test automation is the best solution for the above questions. Nowadays all the project managers are thinking for software test automation to increase the software quality.
To get a high ROI, the framework that we develop for test automation should be efficient, structured and easy to work with. An Easy test automation framework yields high ROI, which will reduce the high initial effort that required in general test automation. We consider many factors for selecting the most appropriate test automation tool:

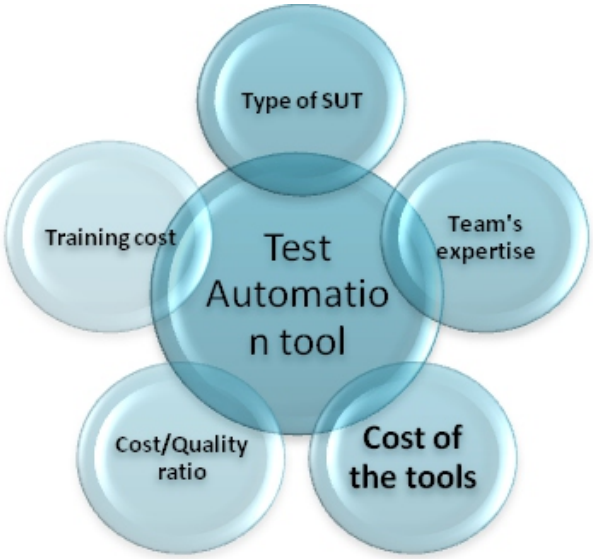



Figure 1 :
Factors Considered for tool selection

### Challenges with Test automation tools

In many cases, an end to end software testing automation can be done only by the usage of several testing tools. Selecting tools which have support for all testing requirements is practically impossible. So testers will limit the test automation to a minimum extend with which the selected compatible tools can provide better solutions. So, there is a high chance for using multiple test automation tools for building an efficient quality test automation framework. Test automation engineers will face below constraints while using multiple test automation tools in a framework.


Figure 2:
Constraints faced with usage of multiple test automation tools

### What could be the optimal solution?

The scenario: With a team of 10 testing members, the scripting expertise may be diverse- three of the members having expertise in VB scripting three others are ok with VB scripting but are very good with j scripting and the rest four are not familiar with VB or J scripting, but are good at c# scripting. This calls for a right decision on the selection of automation tool- one that supports the scripting languages that the team is good at. Even if we select a tool that supports all these three scripting languages, team cannot work in a

# FaSaT An Interoperable Test Automation Solution

mixed mode of scripting i.e. team has to choose one particular scripting language at the beginning and has to stick on to the same. This actually is a compromise on the team's expertise and will raise additional training and learning requirements for a few of the team members on the selected scripting language. Similar is the case with automation tool's expertise.

The solution: Would it be better if an environment can be created wherein the members can work on their preferred scripting languages, and doesn't demand any conversion to a common scripting language. And if these scripts can be ran against any of the automation test tools? Much better! Automation tool selection usually depends on the type of testing (web, database, desktop UI etc). This in turn will demand usage of multiple test automation tools in larger products with extensive functionalities. Hence a framework that hides the automation test tool details/dependency would relieve the project team from complexities in managing the scripts.

A complete solution for above mentioned issues is to develop a test automation framework which can offer support for Interoperability among multiple tools and scripting languages with minimal effort. Considering the above constraints, we came up with Flexi any Script any Tool Test automation framework.

## Introduction to FaSaT

Due to ever evolving requirements of test automation and tools, software service companies like QuEST has realized that there is a need of a new test automation development platform that has unrestricting and flexible features. In a rather successful attempt to address the problems faced by testers to overcome test automation tool integration, compatibility with co-tools, test engineers' tool expertise, QuEST has started thinking about a new testing platform which can afford all these technical burdens. This initiative created the test automation integration platform FaSaT (Flexi any Script any Tool framework).

FaSaT's internal name is 'Common Test Suite' - CTS. Common Test Suite provides languages interoperability across several test automation tools. Each tool can use scripts written in other languages other than the vendor specification. Test automation scripts written in several programming languages targeted for FaSaT execute in software environments suggested by each test automation tools that are configured in FaSaT.

## What is FaSaT

FaSaT aims at interoperability among multiple test automation tools and multiple scripting languages. The framework accepts test scenarios written in different scripting languages. This mixed language scripting can be across script files or even within a single script file. On top of this, user can target test execution to any of the test automation tools configured in FaSaT.

Multiple scripting language support is availed by specifying a tag (the script tag) within the script. Thus switching between scripting languages are achieved within individual scripts itself.

Again, multiple test automation tool support is availed in a similar manner by specifying automation tool tags. With this support, the framework actually hides the automation tool details from the user. This in turn opens the door for another key advantage – user can stick on to scripting language of his/her of expertise, even when the targeted test automation tool doesn't support this scripting language.

At runtime, all the heterogeneous scripts are converted to an intermediate script. This in turn is converted to a native binary by the framework. Executing this binary invokes the particular test automation tool(s) specified by the user and performs the test scenarios.

# FaSaT An Interoperable Test Automation Solution

## Design

### a) Interoperability

Test engineers have experience on several popular test automation tools. Everyone knows how to interface each tool with the System Under Test (SUT). Emerging number of test automation tools which are tuned to perform well with noted requirements. Each tool shall follow its own interfacing programming syntaxes and semantics. It would be very much difficult for humans to remember all these interfacing syntaxes and switching between the semantics.

FaSaT provides means to interface any configured test automation tools with any preferred programming language syntaxes and unified semantics. Switching between interfacing languages are provides through “script” tag. Flexibility is provided through easy integration of new programming languages to FaSaT through single line configuration files.

### b) Common Test Suite (CTS)

Common Test Suite (CTS) serves as the execution engine for FaSaT. All scripts are executed fully under the supervision of CTS. This guarantee that logging and other platform services are fully available with ease of usage for all the processed test scripts. Cross platform targets and de-compilations are fully managed by CTS. “Code to CTS” makes platform services available to automation scripts. This makes test engineers' jobs quite easy. CTS is the re-distributable part of FaSaT which can be delivered across different environments where the test engineers are very well confident.

### c) Platform Services

Platform services are provided as a post activity and user has to provide very minimal information or hint for availing these features to their automation scripts. Platform service like logging can be fully enabled by providing a single line statement “TestAutomation.Log()”. This method is targeted to CTS. All CTS targeted method calls shall get converted to actual platform invocations on processing the row script by CTS. Global as well as local settings can override default platform configurations. Named and global settings fully serve the custom platform configurations.

### d) Target Services

Platform services are the API(s) available for all configured test automation tools. Regardless of target test automation tools, users have the freedom to use platform services. But there are cases where user needs to directly invoke several API(s) of actual target tools (for optimum speed, performance). This can be made available through Target Services.

FaSaT doesn't promote the usage of Target Services due the fact that any mistake from the test engineer's side may result in un-predicted result. Using these API(s) shall be allowed with much care. Since this facility is prone to get weird result, CTS allows every invocation to these Target Services shall be strictly enclosed between the container tags of “direct”. CTS never process any direct tag contents. All the contents shall be passed directly to the actual test automation tool.

### e) Easy Integration

FaSaT has its own way of friendly integration. Copying the deliverables of FaSaT will create a fully functional test environment.
Users have given a customized way of distribution of FaSaT. This includes only CTS and dependent libraries. This is the minimal installation. This installation also follows copy pasting strategy. Mere installation of these components doesn't make it usable. For using the minimal installation, user shall have to extend his/her companion editor support to CTS.

### f) De-Compilation

Test automation scripts written in different programming languages are not easy to understand by an external person, he who is the reviewer or a test manager or customer. Scripts written in any programming language syntaxes and semantics can be easily de-compiled to any

# FaSaT An Interoperable Test Automation Solution

configured interfacing language. This unified script format makes the review and re-works easier for a new test engineer as well.

### g) Cross Platform Support

Software requirements are going to the heights of hybrid systems. These systems may have support for several platforms. Or in several cases, these systems may run distributed in different platforms. FaSaT doesn't limit the scope of end to end testing of these distributed software systems. FaSaT can generate target binaries for many well-known platforms. User can have the facility to extend the platform support for FaSaT. In some cases, test engineers develop automation scripts in one platform and execute them in some other platforms. Cross compilation facility of FaSaT makes these tasks quite simple.

### h) Dynamic Automation Tool Selection

Since hybrid systems are very common in engineering domain software, providing an end-to-end test automation solution involve multiple test automation tools. For a business test automation to be meaningful, relate heterogeneous software modules within same script methods. This requires switching between one test automation tools to another inside the same script. Using the platform service “TestAutomation.Use()”, test engineers can switch between different test automation tools inside same script.

```
<Use>Selenium</Use>
<Script Language="Vbscript">
    Function TC1(x As String, y As Integer) As Boolean
        this.Use "Selenium"
        Dim text as String
        TestAutomation.SetHost "d1291-w2k8-vm4.tvm.nestgroup.net"
        TestAutomation.Url = "/EMS/Login.aspx"
        TestAutomation.Maximize
        this.Use "AutoIT"
        TestAutomation.Click "ibutLogin", CommonTypes.By.Id
        TestAutomation.Wait 17000
        TestAutomation.SetText "ctl00_ContentPlaceHolder1_txtBxReasonForResignation", "hellojhgh", CommonTypes.By.Id
        text=TestAutomation.GetText("ctl00_ContentPlaceHolder1_heading", CommonTypes.By.Id)
        If text=="Resignation Request" Then
            TestAutomation.Log "Navigation Success"
            TC1=true
        End If
        TestAutomation.Log "Navigation Failed"
        TC1=false
    End Function
</Script>
```

Figure 3: **Snippet showing Automation Tool/Scripting Language selection**

### i. Tool Independent Scripting

With dynamic automation tool selection, user has given the flexibility to switch between test automation tools on the fly. This brings the scenario of inter-communication between the heterogeneous tools. By vendor, inter-communications may or may not be provided. Test engineers are advised to implement communication between different test automaton tools through CTS, which will handshake the communication effectively.
Exceptional cases can be handled by the usage of target services. FaSaT team doesn't promote the usage of any vendor provided communication strategies.

# FaSaT An Interoperable Test Automation Solution

## Architecture

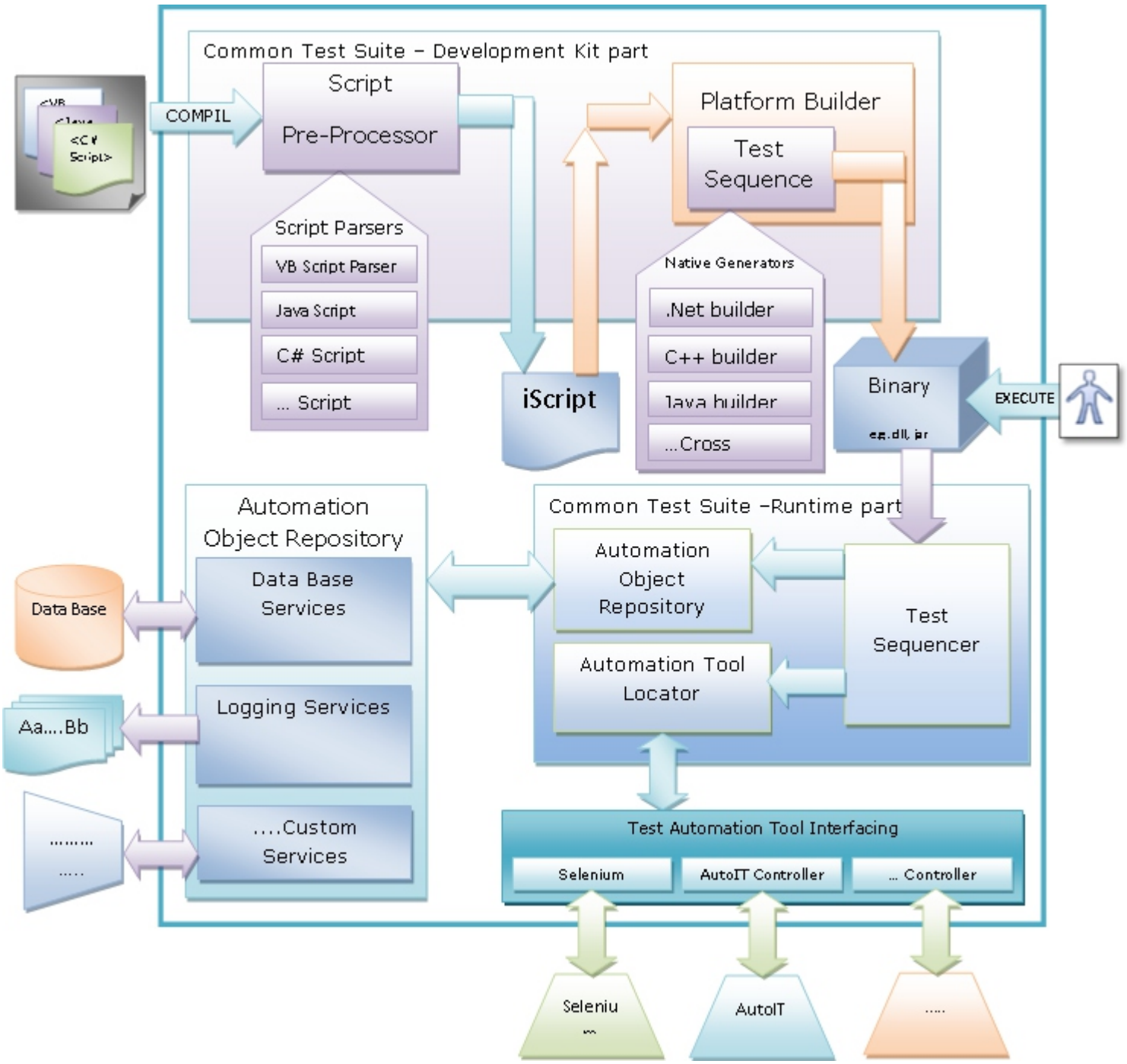



Figure 4: **FaSaT Architechure Diagram**

# FaSaT An Interoperable Test Automation Solution

### a. Script Pre-Processor

The purpose of script pre-processor is to provide a scripting syntax and semantics neutral platform for easy test automation development and maintenance. CTS shall use this module for processing anonymous scripts received as input from user. Script pre-processor shall convert all the heterogeneous scripting syntaxes to a unified model named "iScript" (acronym for "intermediate script") which is an xml format.

By implementing the scripting syntax and semantics interoperability within the CTS with the help of script pre-processors, this functionality will not be tied to a single scripting syntax but will be available across many scripting syntaxes as configured by user.

FaSaT framework provides support for typical scripting languages widely used across various test automation tools. So that users familiar with test automation tools can find an easy way in to FaSaT.

User can easily extend the power of script pre-processor to their friendly script syntaxes though script pre-processor plug-in(s) of CTS. Even if user can add more script pre-processors to CTS, user cannot customize script execution order and priority. Script execution is fully controlled under the supervision of CTS.

### b. Intermediate Script – iScript

Unified model of test automation scripts is supported by FaSaT. This is a pure xml file built by script pre-processors by parsing input scripting files. Experienced users can write down this file directly so that, execution speed can be improved by excluding the latency injected by script pre-processors. FaSaT team is not highly encouraging this activity since user typos may result in weird behavior. The syntaxes followed in iScript are not standardized. This is because; iScrpt inputs are managed and processed only by CTS. So every syntaxes and semantics defined for iScript are very close to CTS design and implementation.

### c. Platform Builder

This part of CTS processes unified iScript syntaxes to system buildable solutions. These solutions can be built for target platform where automation scripts are executed. Platform builder generate buildable solutions with the help of pre-configured native generators. Native generators can be extended by user to give more flexibility for distributing binary across various platforms.

Depending on the configuration, platform builder shall generate one of .dll, .jar, etc. These binaries can be executed in the target environment either with or without the help of self starting executables.

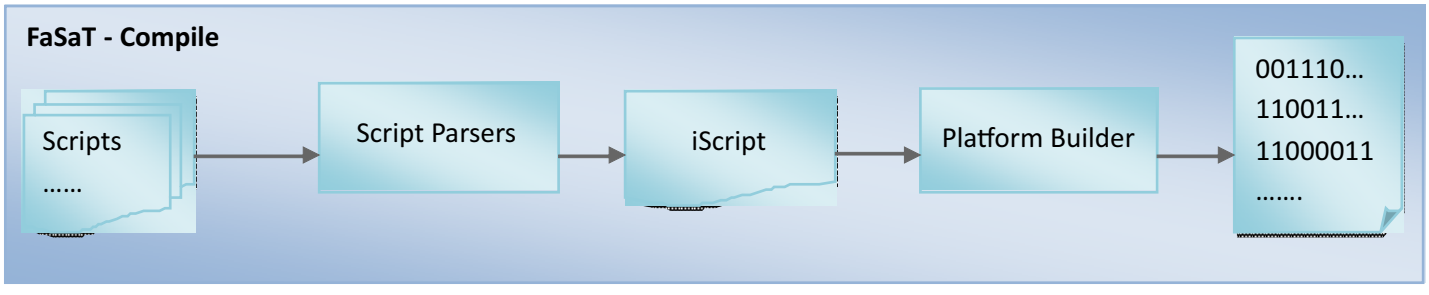



Figure 5: FaSaT Compile - Flow Diagram

# FaSaT An Interoperable Test Automation Solution

### a. Automation Object Repository

This is the factory which produces all CTS interfacing objects. One such typical example is “TestAutomation”. Objects corresponding to AOR are self instantiated and so no need to add instantiation steps inside the scripts. Providing instantiation on top of all scripts doesn't cause any critical issues. But changing the AOR objects during the fly will give weird behavior. So FaSaT team doesn't highly promote the explicit instantiation of AOR objects.

AOR objects can be accessed by name throughout the scripts. All the dependency across AOR objects shall be adjusted by CTS. So that user can concentrate in the test automation of business functionalities rather than looking across the coding headaches.

### b. Automation Tool Locator

Automation Tool Locator is CTS part who is an agent to locate actual test automation tool to be set as current target. ATL takes the highest usage during execution of test automation. During compilation AOR objects are mapped with ATL corresponding code. In order for keeping little dependency from tool to framework, every test automation tools are considered as external services.

Tool instantiation and life are managed completely by ATL. There is no need to explicitly manage test automation tools.
Test engineers who were writing test scripts can see ATL managed objects through AOR objects. While platform builder converts iScript to concrete solutions, all AOR object references are converted to ATL compatible code. AOR services like “TestAutomation.Use()” will switch between test automation tools during test automation. This statement will be compiled to ATL compatible code so as ATL will look in the list of configured test automation tools and select the right one. This will be returned to test sequencer. After execution of this code, every call to AOR objects will result in current ATL returned test automation tool.

### c. Test Sequencer

This is the target platform binary which is fully capable of running independently on the target environment. This can be one of the .dll, .jar, etc. Test sequencer shall execute test automation with the help of the services provided by CTS through the usage of ATL and AOR.

Distribution of test sequencer without development environment needs, just copy and paste the binaries generated to target folder and do execute in the conventional methods. The only prerequisite for test sequencer is that all the configured test automation tools shall be installed and fully functional at target environment.

Automation execution speed is improved by “compile if newer” if newer facility of platform builder. This facility is by default enabled for CTS. There might have cases where this option need to be enabled is included script files got modified then it is needed to change this option to “compile always”.

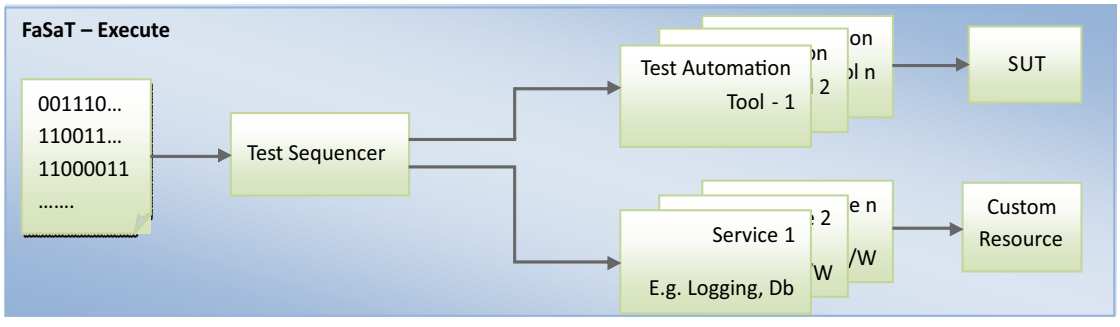



Figure 6: FaSaT Execute - Flow Diagram

# FaSaT An Interoperable Test Automation Solution

## Take Away

- Easy scripting: Does not demand expertise in test automation tools or scripting languages
- Flexibility: Supports all standard test automation tools and scripting languages
- Minimal effort for setting up test automation framework
- Automation tool switching within single script
- Scripting language switching within single script
- De-compilation: Conversion back to any supported scripting languages
- Applicable for web, windows, data driven testing

## Case Study

FaSaT is a Complete Test Automation solution framework with inbuilt inter-operability facility. It was really challenging for the team to provide a best automation solution for a product which required image based verifications along with typical object based scenarios and verifications. For an optimal solution, it demanded usage of multiple test automation tools that follow different scripting approaches and scripting languages.

With the inter-operable features of FaSaT, it was easy to use multiple tools and scripting languages in a single script file. As a result, test team could effectively utilize the capabilities of different automation tools. Also it helped to make the best use of scripting language expertise of each resource in team.

Tools chosen were TestComplete and Sikuli. With this combination, object based scenarios were fully handled by TestComplete and sophisticated image comparison requiring scenarios were taken care by Sikuli. About 70% of tests were written against Testcomplete (a combination of J Script and VB script) and remaining 30% were written against Sikuli (J script). The easy and successful integration has proved the effectiveness of FaSaT in conquering real challenges with test automation.

## Return on Investment

The multi-dimensional advantages gained after implementing this framework across organization as listed below.

No need for highly experienced test automation engineers: As this framework doesn't require much expertise on different automation tools, we can eliminate the cost associated with highly experienced resources.

Reduced training costs: As the framework is very simple to understand, there is no need for a detailed training which reduced the training cost.

Less maintenance cost: Maintenance cost for the test scripts would be on a lesser side as we are using name mapping techniques which simplifies the scripting complexity. Also the scripts are neutral to the target test automation tool which again eliminates the intricacies associated with specific tools

Open source adaptive: As this framework is interoperable with multiple test automation tools, we can eliminate the cost for buying full-fledged test automation tool available in market and can go for multiple open source alternatives that are best in each feature areas.

## Future Work

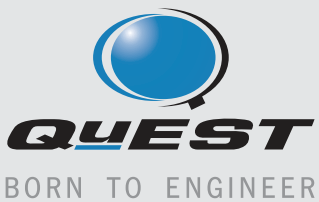
Following areas are identified with a scope for future work:

- Recording facility can be extended by adopting target tool's recording functionality.
- Flowchart based test sequencing: Engineers can configure test sequences by drag and drop operation on several control boxes in sequence form and can execute the test scenarios. Test script building will be handled by the framework.
- Enhanced reporting facility
- Life cycle based test automation
- Template based test sequences

## FaSaT An Interoperable Test Automation Solution

### Bibliography

http://en.wikipedia.org/wiki/.NET_Framework
http://en.wikiversity.org/wiki/Introduction_to_Microsoft.NET




BORN TO ENGINEER

www.quest-global.com