Automation Script 1.0 Beta Overview

**Introduction**

Automation script is a custom [DSL](http://en.wikipedia.org/wiki/Domain_Specific_Language) developed for automating Windows-based applications, even for creating fully automated tests. It’s XML-based. It’s not bound to [CCF](http://en.wikipedia.org/wiki/Customer_Care_Framework) hence it can be used from any other .NET application. It’s composed by a series of steps, which may (or not) contain some pre-requisites that must be met in order to proceed to the next step. Its main purpose is to provide a starting point for creating automation-based solutions, without forcing the developer to do some of the Win32 plumbing but helping him to focus on the business requirement.

**Contents**

[Description 2](#_Toc231725364)

[What is Automation Script? 2](#_Toc231725365)

[Why Automation Script? 2](#_Toc231725366)

[How does it work? 2](#_Toc231725367)

[Benefits 4](#_Toc231725367)

# Description

## What is Automation Script?

In a few words and to keep it simple, the best definition would be: A custom [DSL](http://en.wikipedia.org/wiki/Domain_Specific_Language) designed to helping developers who require performing any automation on an existing application.

## Why Automation Script?

As previously mentioned, Automation Script is not bound to [CCF](http://en.wikipedia.org/wiki/Customer_Care_Framework) even when it can be used together, which is contrary to [HAT](http://msdn.microsoft.com/en-us/library/dd632242(v=MSDN.10).aspx) that’s dependant on [CCF](http://en.wikipedia.org/wiki/Customer_Care_Framework). By abstracting the developer from Win32API calls, the developer can focus on the business requirements instead of Win32 plumbing. This is a beta release and there are still a bunch of features that can be added to it.

## How does it work?

The principle is very simple; essentially, it’s a parser/dispatcher factory. It’s a parser because every Automation Script is a collection of steps, pre-requisites and conditions and they’re all expressed as [XML](http://en.wikipedia.org/wiki/XML). The parser is responsible for processing, interpreting and creating objects at runtime. The minimal unit of work in an Automation Script is a Step.

It’s a message dispatcher as well, because the step contains information about the Windows’ object that will receive the message and the message itself with its associated arguments.

The way it works can be described by the following example:

*Engine engineObj = Engine.GetInstance(hWnd);*

*engineObj.OnAutomationExecuted += engineObj \_OnAutomationExecuted;*

*engineObj.OnAutomationFailed += engineObj \_OnAutomationFailed;*

*engineObj.OnMessageSentCompleted += engineObj \_OnMessageSentCompleted;*

*engineObj.OnScriptStopped += engineObj \_OnScriptStopped;*

*engineObj.RunAutomationScript(script, IntPtr.Zero, AutomationEndCode);*

An *Engine* object is created from a [hWnd](http://en.wikipedia.org/wiki/Hwnd), the developer subscribes to the events he’s interested in. The method *RunAutomationScript* is executed, taking as parameters:

1. A XDocument object containing the Automation Script
2. An optional [hWnd](http://en.wikipedia.org/wiki/Hwnd) in case we didn’t create an instance from an existing [hWnd](http://en.wikipedia.org/wiki/Hwnd)
3. A [callback](http://en.wikipedia.org/wiki/Callback_(computer_science)) which points to custom code that’s executed at the end of the script

A sample Automation script would be like

<autoscript version="1.0">

<!-- Report Header -->

<step Msg="770" wParam="0" lParam="0" targetClass="Edit" clipBoardData="This file has

been automatic generated @ {0}"/>

<step targetClass="Edit" stepAction="KeyStroke" virtualKeyCode="VK\_ENTER"/>

<step Msg="770" wParam="0" lParam="0" targetClass="Edit"

clipBoardData="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"/>

<step targetClass="Edit" >

<preRequisites>

<for stepCount="2" stepAction="KeyStroke" virtualKeyCode="VK\_ENTER"/>

</preRequisites>

</step>

<!-- Save file -->

<step Msg="273" wParam="65539" delay="1000" lParam="0"/>

<step Msg="770" wParam="0" lParam="0" delay="1000" pathOrTitle="Save As"

clipBoardData="{2}" exitAutomationOnFailure="true">

<preRequisites>

<condition type="Dialog" behavior="Show" retries="3" timeInterval="1500" />

</preRequisites>

</step>

</autoscript>

***Note: There’s a sample application with this document and the Automation Script source code.***

## Benefits

We can mention the following benefits:

1. Provide a starting point for automating existing Windows applications without being dependent on [CCF](http://en.wikipedia.org/wiki/Customer_Care_Framework)
2. Provide developers with a [framework](http://en.wikipedia.org/wiki/Software_framework) that can be extended and it’s based in standards
3. Centralized repository for Automation Scripts (SQL Server)
4. Developers don’t need to learn a new set of tools or languages, the only required tool comes with Visual Studio .NET (SPY++)
5. Automation Script relies on the Windows message dispatching infrastructure, so it’s maintained between versions and every application uses it regardless of the programming language it was written on (e.g.: VB, C++, Delphi, PowerBuilder among others)