Introduction to User Interface Automation using UIADriver Tool *aka* VedaTool ☺

# Introduction

The Windows Automation API consists of two technologies—Microsoft Active Accessibility and Microsoft UI Automation. Microsoft Active Accessibility is the legacy accessibility technology that was introduced as a platform add-in for Windows 95, while UI Automation is a newer, more capable technology that overcomes the limitations inherent in Microsoft Active Accessibility.

The UI Automation Specification provides flexible programmatic access to UI elements on the Windows desktop, enabling assistive technology products such as screen readers to provide information about the UI to end users and to manipulate the UI by means other than standard input.

UI Automation is broader in scope than just an interface definition. It provides:

* An object model and functions that make it easy for client applications to receive events, retrieve property values, and manipulate UI elements.
* A core infrastructure for finding and fetching across process boundaries.
* A set of interfaces for providers to express the tree structure, general properties, and functionality of UI elements.
* A "control type" property that allows clients and providers to clearly indicate the common properties, functionality, and structure of a UI object.

UI Automation improves on Microsoft Active Accessibility by:

* Enabling efficient out-of-process clients, while continuing to allow in-process access.
* Exposing more information about the UI in a way that allows clients to be out-of-process.
* Coexisting with and leveraging Microsoft Active Accessibility without inheriting its limitations.

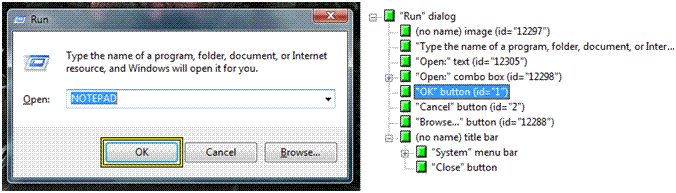
## UI Automation Elements

UI Automation exposes every piece of the UI to client applications as an automation element. Providers supply property values for each element. Elements are exposed as a tree structure, with the desktop as the root element.

Automation elements expose common properties of the UI elements they represent. One of these properties is the control type, which describes its basic appearance and functionality (for example, a button or a check box).

## UI Automation Tree

The UI Automation tree represents the entire UI: the root element is the current desktop, and child elements are application windows. Each of these child elements can contain elements representing menus, buttons, toolbars, and so on. These elements in turn can contain elements like list items, as the following illustration shows.



Be aware that the order of the siblings in the UI Automation tree is quite important. Objects that are next to each other visually should also be next to each other in the UI Automation tree.

UI Automation providers for a particular control support navigation among the child elements of that control. However, providers are not concerned with navigation between these control sub-trees. This is managed by the UI Automation core, using information from the default window providers.

## UI Automation Properties

The UI Automation Specification defines two kinds of properties: automation element properties and control pattern properties. Automation element properties apply to most controls, providing fundamental information about the element, such as its name. Control pattern properties apply to control patterns, which are described next.

## UI Automation Control Patterns

A control pattern describes a particular aspect of the functionality of an automation element. For example, a simple "click-able" control like a button or hyperlink should support the Invoke control pattern to represent the "click" action.

Each control pattern is a canonical representation of possible UI features and functions. The current implementation of UI Automation defines 22 control patterns. The Windows Automation API can also support custom control patterns. Unlike Microsoft Active Accessibility role or state properties, one automation element can support multiple UI Automation control patterns.

## UI Automation Control Types

A control type is an automation element property that specifies a well-known control that the element represents. Currently, UI Automation defines thirty-eight control types, including Button, CheckBox, ComboBox, DataGrid, Document, Hyperlink, Image, ToolTip, Tree, and Window.

Before you can assign a control type to an element, the element needs to meet certain conditions, including a particular automation tree structure, property values, control patterns, and events. However, you are not limited to these. You can extend a control with custom patterns and properties, as well as with the predefined ones.

## UI Automation Events

UI Automation events notify applications of changes to, and actions taken with automation elements. There are four different types of UI Automation events, and they do not necessarily mean that the visual state of the UI has changed. The UI Automation event model is independent of the [WinEvent](https://docs.microsoft.com/en-us/windows/win32/winauto/winevents-infrastructure) framework in Windows, although the Windows Automation API makes UI Automation events interoperable with the Microsoft Active Accessibility framework.

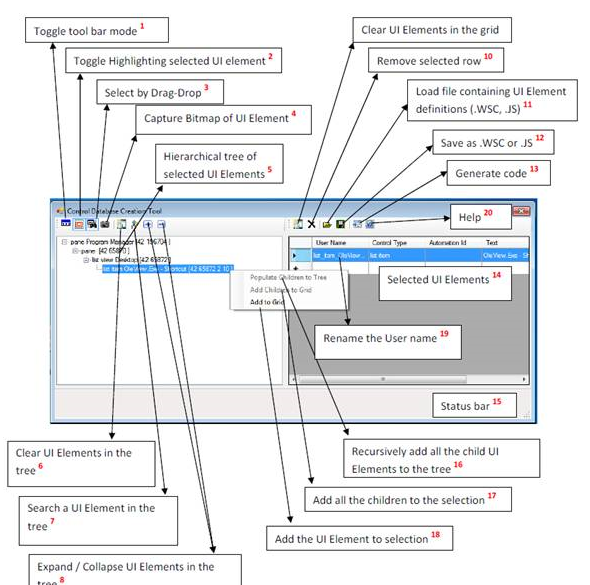
# UIADriver Tool

Consists of two components:

### ControlDB Tool

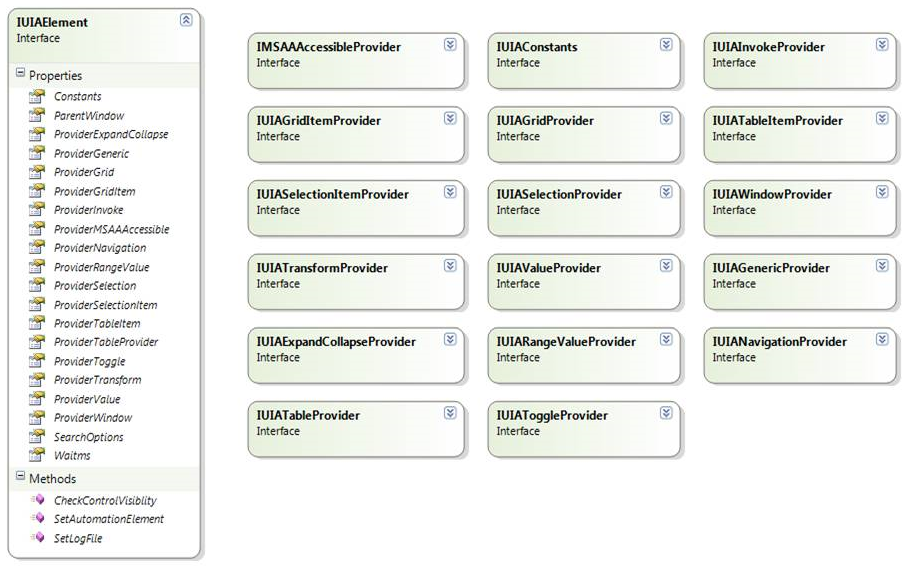
A GUI application to capture UI elements that needs to be automated. These UI elements can be saved as a C# class later to be used with UI control patterns to create user interface automation.

It also provides a powerful “record and play” option to record user actions and automatically generate code for user interface automation



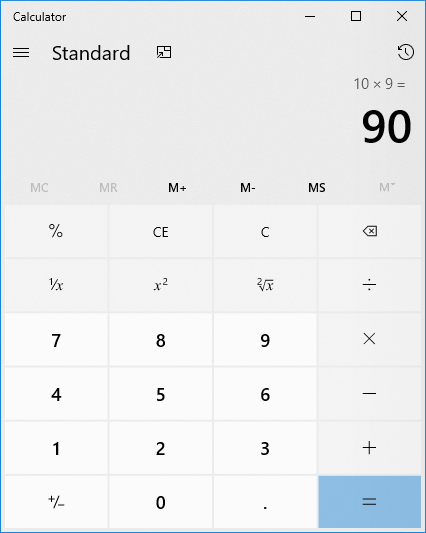
### UIADriver class Library

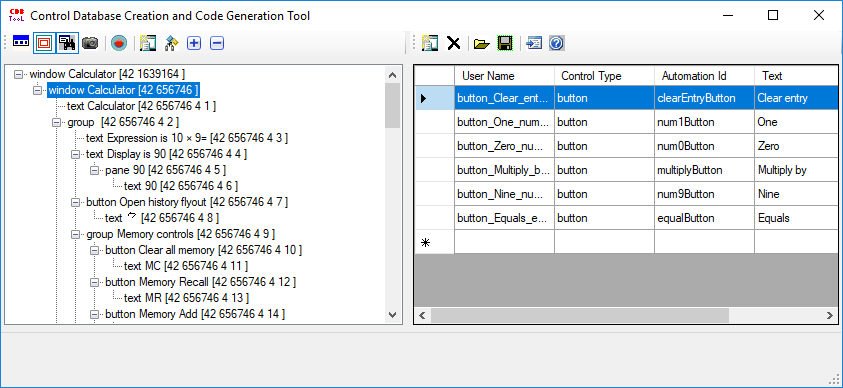
Provides a class library of control patterns that can be applied on the captured ui elements above to provide user automation. Both MSAA and UIA technologies are supported.



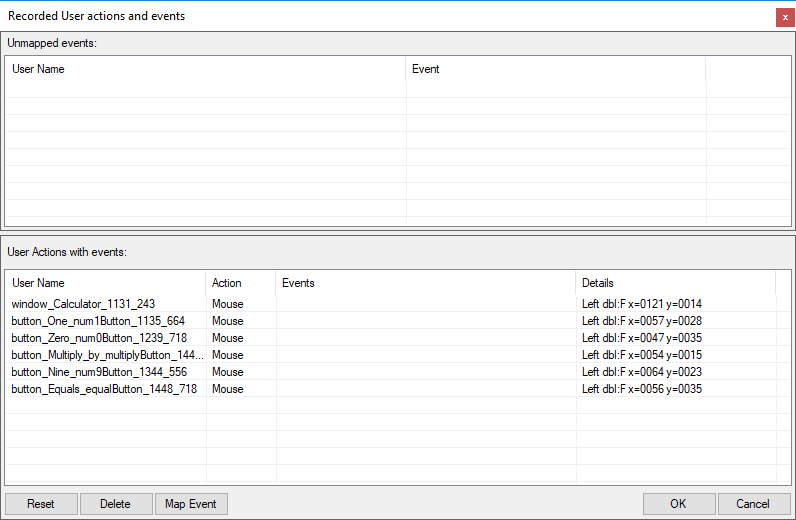
## Example

Using ControlDBtool a simple calculation is automated on the calc.exe.



Using record and play feature, the operation 10 x 9 is recorded.

Using code generator, the entire recorded sequence is presented to the user to view and update any unmapped UI Event with user action.



The playback code for the recorded sequence is automatically generated for the selected UI controls.

