PowerToys Tests

The PowerToys tests are implemented using Appium and use the Windows Application Driver as an Appium compatible server for Windows applications.

Prerequisites

- Install the latest stable version of Windows Application Driver in the test machine: v1.1 Release
- Install the ".Net desktop development" components in Visual Studio 2019. It should have support for `C#` and `.Net Framework 4.7.2`.
- Install PowerToys
- In Windows 10 Settings, turn on `Developer Mode` (open the Windows 10 Settings and search "Developer settings").

If you have `PowerToys` installed, it can be launched automatically. Otherwise, if you are testing a local build, you should start `PowerToys` before running the tests.

Preparing the test machine

- Start `PowerToys` if necessary (see the Prerequisites).
- Run `WinAppDriver.exe` in Administrator mode, on the test machine. By default it's installed in `C:\Program Files (x86)\Windows Application Driver\`
- **Note:** notifications or other application windows, that are shown above the PowerToys settings window or tray icon, can disrupt the testing process.

When testing on a remote machine, a Firewall exceptions must be added and the IP and port must be passed when starting "Windows Application Driver". Here's how to do it from the Windows Application Driver FAQ:

Running on a Remote Machine Windows Application Driver can run remotely on any Windows 10 machine with `WinAppDriver.exe` installed and running. This *test machine* can then serve any JSON wire protocol commands coming from the *test runner* remotely through the network. Below are the steps to the one-time setup for the *test machine* to receive inbound requests:

1. On the *test machine* you want to run the test application on, open up **Windows Firewall with Advanced Security**
    - Select **Inbound Rules** -> **New Rule...**
    - **Rule Type** -> **Port**
    - Select **TCP**
    - Choose specific local port (4723 is WinAppDriver standard)
    - **Action** -> **Allow the connection**
    - **Profile** -> select all
    - **Name** -> optional, choose name for rule (e.g. WinAppDriver remote).

Below command when run in admin command prompt gives same result

```
netsh advfirewall firewall add rule name="WinAppDriver remote" dir=in action=allow prot
```

2. Run `ipconfig.exe` to determine your machine's local IP address > **Note**: Setting `*` as the IP address command line option will cause it to bind to all bound IP addresses on the machine
3. Run `WinAppDriver.exe 10.X.X.10 4723/wd/hub` as **administrator** with command line arguments as seen above specifying local IP and port
4. On the *test runner* machine where the runner and scripts are, update the test script to point to the IP of the remote *test machine*

Starting the tests on the Development Machine

- Open `PowerToys.sln` in Visual Studio.
- Build the `PowerToysTests` project.
- Select `Test > Windows > Test Explorer`.
- Select `Test > Run > All` tests in the menu bar.

Once the project is successfully built, you can use the **TestExplorer** to pick and choose the test scenario(s) to run

If Visual Studio fail to discover and run the test scenarios: 1. Select **Tools** > **Options...** > **Test** 2. Under *Active Solution*, uncheck *For improved performance, only use test adapters in test assembly folder or as specified in runsettings file*

If a remote test machine is being used, the IP of the test machine must be used to replace the `WindowsApplicationDriverUrl` value in PowerToysSession.cs.

Extra tools and information

For tests creation you will need a tool that enables you select any UI element and view the element's accessibility data. For this purpose you could use AccessibilityInsights or Inspect.

- `inspect.exe` you can find installed at `C:\Program Files (x86)\Windows Kits\10\bin\<version>\<platform>\inspect.exe`
- `AccessibilityInsights` you can download here

How to use Inspect Open Inspect, find element you need to investigate (by clicking on element or finding it in a tree) and in the right part of inspector window you will see info about this element.

Examples for searching elements with values of `Name`, `AutomationId` and `ControlType`:

```
//use FindElementByAccessibilityId with AutomationId value
session.FindElementByAccessibilityId("40001");
```

```
session.FindElementByAccessibilityId("decrementZones");

session.FindElementByName("PowerToys Settings");

//with XPath you can search elements with more specific information
session.FindElementByXPath("//Pane[@Name=\"PowerToys Settings\"]");
session.FindElementByXPath("//Edit[contains(@Name, \"hotkey\")]");
session.FindElementByXPath("//Pane[@Name=\"PowerToys Settings\"]/*[@LocalizedControlType=\"t
```

One more thing to notice: close helper tools while running tests.
Overlapping windows can affect test results.