Most Complete Appium C# Cheat Sheet

Initialize

```
// NuGet: Appium.WebDriver
_appiumLocalService = new AppiumServiceBuilder().UsingAnyFreePort().Build();  // Creates local Appium server instance
_appiumLocalService.Start(); // Starts local Appium server instance
var appiumOptions = new AppiumOptions();
// Android capabilities
appiumOptions.AddAdditionalCapability(MobileCapabilityType.AutomationName, "UiAutomator2");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.DeviceName, "Android Virtual Device");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.PlatformName, "Android");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.PlatformVersion, "7.1");
// iOS capabilities
appiumOptions.AddAdditionalCapability(MobileCapabilityType.AutomationName, "XCUITest");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.DeviceName, "iPhone 11");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.PlatformName, "iOS");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.PlatformVersion, "13.2");
// Install and start an app on Android
appiumOptions.AddAdditionalCapability(AndroidMobileCapabilityType.AppPackage, "com.example.android.apis");
appiumOptions.AddAdditionalCapability(AndroidMobileCapabilityType.AppActivity, ".ApiDemos");
appiumOptions.AddAdditionalCapability(MobileCapabilityType.App, "path/to/TestApp.apk");
// Install and start an app on iOS
appiumOptions.AddAdditionalCapability(MobileCapabilityType.App, "path/to/TestApp.app.zip");
// Start mobile browser on Android
appiumOptions.AddAdditionalCapability(MobileCapabilityType.BrowserName, "Chrome");
// Start mobile browser on iOS
appiumOptions.AddAdditionalCapability(MobileCapabilityType.BrowserName, "safari");
// Set WebView Context for Hybrid Apps
((IContextAware)_driver).Context = ((IContextAware)_driver).Contexts.FirstOrDefault(c => c.Contains("WEBVIEW"));
// Use local server instance
_driver = new AndroidDriver<AndroidElement>(_appiumLocalService, appiumOptions); // initialize Android driver on local server instance
_driver = new IOSDriver<IOSElement>(_appiumLocalService, appiumOptions); // initialize iOS driver on local server instance
// Use remote Appium driver
_driver = new AndroidDriver<AndroidElement>(new Uri("127.0.0.1:4723/wd/hub"), appiumOptions); // initialize Android remote driver
_driver = new IOSDriver<IOSElement>(new Uri("127.0.0.1:4723/wd/hub"), appiumOptions); // initialize iOS remote driver
```

Locators

```
_driver.FindElementById("android:id/text1");
_driver.FindElementByClassName("android.widget.CheckBox");
_driver.FindElementByXPath("//*[@text='Views']");
_driver.FindElementByAccessibilityId("Views");
_driver.FindElementByImage(base64EncodedImageFile);
_driver.FindElementByAndroidUIAutomator("new
UiSelector().text(\"Views\");");
_driver.FindElementByIOSNsPredicate("type == 'XCUIElementBottn' AND
name == 'ComputeSumButton'");
// Find multiple elements
_driver.FindElementsByClassName("android.widget.CheckBox");
```

Actions

```
// Alert handling
_driver.SwitchTo().Alert().Accept()
_driver.SwitchTo().Alert().Dismiss()
// Basic actions:
element.Click();
element.SendKeys("textToType");
element.Clear();
// Change orientation
var rotatable = (IRotatable)_driver;
rotatable.Orientation = ScreenOrientation.Landscape; // Portrait
// Clipboard
_driver.SetClipboardText("1234", "plaintext");
string clipboard = _driver.GetClipboardText();
// Mobile gestures
TouchAction touchAction = new TouchAction(_driver);
touchAction.Tap(x, y, count);
touchAction.Press(x, y);
touchAction.Wait(200);
touchAction.MoveTo(x, y);
touchAction.Release();
touchAction.LongPress(x, y);
touchAction.Perform();
// Simulate phone call (Emulator only)
_driver.MakeGsmCall("5551237890", GsmCallActions.Accept);
_driver.MakeGsmCall("5551237890", GsmCallActions.Call);
_driver.MakeGsmCall("5551237890", GsmCallActions.Cancel);
_driver.MakeGsmCall("5551237890", GsmCallActions.Hold);
// Set GSM voice state (Emulator only)
_driver.SetGsmVoice(GsmVoiceState.Denied);
_driver.SetGsmVoice(GsmVoiceState.Home);
_driver.SetGsmVoice(GsmVoiceState.Off);
_driver.SetGsmVoice(GsmVoiceState.On);
_driver.SetGsmVoice(GsmVoiceState.Roaming);
_driver.SetGsmVoice(GsmVoiceState.Searching);
_driver.SetGsmVoice(GsmVoiceState.Unregistered);
// Set GSM signal strength (Emulator only)
_driver.SetGsmSignalStrength(GsmSignalStrength.Good);
_driver.SetGsmSignalStrength(GsmSignalStrength.Great);
_driver.SetGsmSignalStrength(GsmSignalStrength.Moderate);
_driver.SetGsmSignalStrength(GsmSignalStrength.NoneOrUnknown);
_driver.SetGsmSignalStrength(GsmSignalStrength.Poor);
// Simulate receiving SMS message (Emulator only)
_driver.SendSms("555-555-5555", "Your code is 123456");
// Toggle services
_driver.ToggleAirplaneMode();
_driver.ToggleData();
_driver.ToggleLocationServices();
_driver.ToggleWifi();
// Soft keyboard actions
_driver.IsKeyboardShown(); // returns bool
_driver.HideKeyboard();
// Lock device
_driver.IsLocked(); // returns bool
_driver.Lock();
_driver.Unlock();
// Notifications
_driver.OpenNotifications();
// Geolocation actions
_driver.Location; // returns Location {Latitude, Longitude, Altitude}
_driver.Location.Altitude = 94.23;
_driver.Location.Latitude = 121.21;
_driver.Location.Longitude = 11.56;
// Get system time
_driver.DeviceTime; // returns string
// Get display density
_driver.GetDisplayDensity(); // returns float
// File actions
_driver.PushFile("/data/local/tmp/file", new FileInfo("path/to/file"))
_driver.PullFile("/path/to/device/file"); // returns byte[]
_driver.PullFolder("/path/to/device"); // returns byte[]
```

Application Management – Android

```
// Install app
_driver.InstallApp("path/to/app.apk");
// Remove app
_driver.RemoveApp("com.example.AppName");
// Verify app is installed
_driver.IsAppInstalled("com.example.android.apis"); // returns bool
// Launch app
_driver.LaunchApp();
// Start activity
_driver.StartActivity("com.example.android.apis", ".ApiDemos");
// Start activity with intent
_driver.StartActivityWithIntent("com.example.android.apis",
".ApiDemos", "intentAction");
// Get current activity
_driver.CurrentActivity; // string property
// Get current package
_driver.CurrentPackage; // string property
// Reset app
_driver.ResetApp();
// Close app
_driver.CloseApp();
// Run current app in background
_driver.BackgroundApp(10); // 10 seconds
// Terminate app
_driver.TerminateApp("com.example.android.apis"); // returns bool
// Check app's current state
_driver.GetAppState("com.example.android.apis"); // returns AppState
// Get app strings
_driver.GetAppStringDictionary("en", "/path/to/file"); // returns
Dictionary<string, object>
```

Advanced Actions – Common

```
// Multitouch action
TouchAction actionOne = new TouchAction(_driver);
actionOne.Press(10, 10);
actionOne.MoveTo(10, 100);
actionOne.Release();
TouchAction actionTwo = new TouchAction(_driver);
actionTwo.Press(20, 20);
actionTwo.MoveTo(20, 200);
actionTwo.Release();
MultiAction action = new MultiAction(_driver);
action.Add(actionOne);
action.Add(actionTwo);
action.Perform();
// Swipe using touch action
TouchAction touchAction = new TouchAction(_driver);
var element = _driver.FindElementById("android:id/text1");
Point point = element.Coordinates.LocationInDom;
Size size = element.Size;
 touchAction
  .Press(point.X + 5, point.Y + 5).Wait(200)
  .MoveTo(point.X + size.Width - 5, point.Y + size.Height - 5)
  .Release().Perform()
// Scroll using IJavaScriptExecutor
var js = (IJavaScriptExecutor)_driver;
var swipe = new Dictionary<string, string>();
swipe.Add("direction", "down"); // "up", "right", "left"
swipe.Add("element", element.Id);
js.ExecuteScript("mobile:swipe", swipe);
// Scroll element into veiw by Android UI automator
_driver.FindElementByAndroidUIAutomator(
"new UiScrollable(new UiSelector()).scrollIntoView(new
UiSelector().text(\"Views\"));");
// Update Device Settings
_driver.SetSetting("sample", "value");
_driver.Settings // Dictionary<string, object> property
```

Advanced Actions – Android

```
// Add photos
driver.PushFile("/mnt/sdcard/Pictures/img.jpg", new
FileInfo("path/to/img.jpg"))
// Simulate hardware key
_driver.PressKeyCode(new KeyEvent(AndroidKeyCode.Home));
_driver.LongPressKeyCode(new KeyEvent(AndroidKeyCode.Keycode_POWER));
// Get current battery percentage
_driver.ExecuteScript("mobile: shell", "{\"command\": \"dumpsys\",
\"args\": [\"battery\"]}").ToString();
// Set battery percentage
_driver.ExecuteScript("mobile: shell", "{\"command\": \"dumpsys\",
\"args\": [\"battery\", \"set\", \"level\", \"100\"]}")
.ToString();
// Reset battery percentage
_driver.ExecuteScript("mobile: shell", "{\"command\": \"dumpsys\",
\"args\": [\"battery\", \"reset\"]}")
.ToString();
// Take screenshot
_driver.GetScreenshot(); // returns Screenshot (base64 encoded PNG)
// Screen record
_driver.StartRecordingScreen(
 AndroidStartScreenRecordingOptions
 .GetAndroidStartScreenRecordingOptions()
  .WithTimeLimit(TimeSpan.FromSeconds(10))
  .WithBitRate(500000)
  .WithVideoSize("720x1280"));
_driver.StopRecordingScreen();
```

Advanced Actions – iOS

```
// Add photos
driver.PushFile("img.jpg", new FileInfo("path/to/img.jpg"))
// Simulate hardware key
var args = new Dictionary<string, string>();
args.Add("name", "home"); // volumeup; volumedown
_driver.ExecuteScript("mobile: pressButton", args);
// Sending voice commands to Siri
var args = new Dictionary<string, string>();
args.Add("text", "Hey Siri, what's happening?");
_driver.ExecuteScript("mobile: siriCommand", args);
// Perform Touch ID in iOS Simulator
_driver.PerformTouchID(false); // Simulates a failed touch ID
_driver.PerformTouchID(true); // Simulates a passing touch ID
// Shake device
_driver.ShakeDevice();
```

Application Management - iOS

```
// Install app
var args = new Dictionary<string, string>();
args.Add("app", "path/to/app.ipa");
_driver.ExecuteScript("mobile: installApp", args);
// Remove app
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.myapp");
_driver.ExecuteScript("mobile: removeApp", args);
// Verify app is installed
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.myapp");
(bool)_driver.ExecuteScript("mobile: isAppInstalled", args);
// Launch app
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.apple.calculator");
_driver.ExecuteScript("mobile: launchApp", args);
// Switch app to foreground
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.myapp");
_driver.ExecuteScript("mobile: activateApp", args);
// Terminate app
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.myapp");
// will return false if app is not running, otherwise true
(bool)_driver.ExecuteScript("mobile: terminateApp", args);
// Check app's current state
var args = new Dictionary<string, string>();
args.Add("bundleId", "com.myapp");
(AppState)_driver.ExecuteScript("mobile: queryAppState", args);
```