## List of Desired Capabilities Methods

1. **getBrowserName()**

public java.lang.String getBrowserName()

1. **setBrowserName()**

public void setBrowserName(java.lang.String browserName)

1. **getVersion()**

public java.lang.String getVersion()

1. **setVersion()**

public void setVersion(java.lang.String version)

1. **getPlatform()**

public Platform getPlatform()

1. **setPlatform()**

public Platform getPlatform()

1. **getCapability Method**

The getCapability method of the DesiredCapabilities class can be used to get the capability that is in use currently in the system.

public java.lang.Object getCapability(java.lang.String capabilityName)

1. **setCapabilityMethod**

The setCapability() method of the Desired Capabilities class can be used to set the device name, platform version, platform name, absolute path of the app under test (the .apk file of the app(Android) under test), app Activity (in Android) and appPackage(java).

"setCapability method" in[Java](https://www.guru99.com/java-tutorial.html)has the below declarations:

setCapability : public void setCapability(java.lang.String capabilityName,boolean value)

setCapability :public void setCapability(java.lang.String capabilityName,java.lang.String value)

setCapability :public void setCapability(java.lang.String capabilityName,Platform value)

setCapability :public void setCapability(java.lang.String key,java.lang.Object value)

**Example for set capability method**

Let us consider an example where we want to run our[Test Case](https://www.guru99.com/test-case.html)on Internet explorer browser to open www.gmail.com website using Selenium Webdriver.

Following is the code.

importorg.openqa.selenium.WebDriver;

importorg.openqa.selenium.ie.InternetExplorerDriver;

public class IEtestforDesiredCapabilities {

public static void main(String[] args) {

WebDriver IEdriver = new InternetExplorerDriver();

driver.manage().window().maximize();

driver.get("http://gmail.com");

driver.quit();

}

}

|  |  |  |
| --- | --- | --- |
| **Used by the selenium server for browser selection** | | |
| **Key** | **Type** | **Description** |
| browserName | string | The name of the browser being used; should be one of {android, chrome, firefox, htmlunit, internet explorer, iPhone, iPad, opera, safari}. |
| version | string | The browser version, or the empty string if unknown. |
| platform | string | A key specifying which platform the browser should be running on. This value should be one of {WINDOWS, XP, VISTA, MAC, LINUX, UNIX, ANDROID}. When requesting a new session, the client may specify ANY to indicate any available platform may be used. For more information see [Grid-Platforms] |

|  |  |  |
| --- | --- | --- |
| **Read-only capabilities** | | |
| **Key** | **Type** | **Description** |
| handlesAlerts | boolean | Whether the session can interact with modal popups, such as window.alert and window.confirm. |
| cssSelectorsEnabled | boolean | Whether the session supports CSS selectors when searching for elements. |

|  |  |  |
| --- | --- | --- |
| **Read-write capabilities** | | |
| **Key** | **Type** | **Description** |
| javascriptEnabled | boolean | Whether the session supports executing user supplied JavaScript in the context of the current page (only on HTMLUnitDriver). |
| databaseEnabled | boolean | Whether the session can interact with database storage. |
| locationContextEnabled | boolean | Whether the session can set and query the browser's location context. |
| applicationCacheEnabled | boolean | Whether the session can interact with the application cache. |
| browserConnectionEnabled | boolean | Whether the session can query for the browser's connectivity and disable it if desired. |
| webStorageEnabled | boolean | Whether the session supports interactions with storage objects. |
| acceptSslCerts | boolean | Whether the session should accept all SSL certs by default. |
| rotatable | boolean | Whether the session can rotate the current page's current layout between portrait and landscape orientations (only applies to mobile platforms). |
| nativeEvents | boolean | Whether the session is capable of generating native events when simulating user input. |
| proxy | proxy object | Details of any proxy to use. If no proxy is specified, whatever the system's current or default state is used. The format is specified under Proxy JSON Object. |
| unexpectedAlertBehaviour | string | What the browser should do with an unhandled alert before throwing out the UnhandledAlertException. Possible values are "accept", "dismiss" and "ignore" |
| elementScrollBehavior | integer | Allows the user to specify whether elements are scrolled into the viewport for interaction to align with the top (0) or bottom (1) of the viewport. The default value is to align with the top of the viewport. Supported in IE and Firefox (since 2.36) |

|  |  |  |  |
| --- | --- | --- | --- |
| **RemoteWebDriver specific** | | | |
| webdriver.remote.sessionid | | string | WebDriver session ID for the session. Readonly and only returned if the server implements a server-side webdriver-backed selenium. |
| webdriver.remote.quietExceptions | | boolean | Disable automatic screnshot capture on exceptions. This is False by default. |
|  | |  |  |
| **Grid-specific** | | | |
| path | string | | ??? Path to route request to, or maybe listen on. |
| seleniumProtocol | string | | Which protocol to use. Accepted values: WebDriver, Selenium. |
| maxInstances | integer | | Maximum number of instances to allow to connect to grid, |
| environment | string | | ??? Possible duplicate of browserName? See RegistrationRequest. |

**Browser Specific Capabilities**

**Chrome specific:**

Capabilities are options that you can use to customize and configure a ChromeDriver session. This page documents all ChromeDriver supported capabilities and how to use them.

The WebDriver language APIs provides ways to pass capabilities to ChromeDriver. The exact mechanism differs by the language, but most languages use one or both of the following mechanisms:

1. Use the ChromeOptions class. This is supported by Java, Python, etc.
2. Use the DesiredCapabilities class. This is supported by Python, Ruby, etc. While it is also available in Java, its usage in Java is deprecated.

**Using the ChromeOptions class**

You can create an instance of ChromeOptions, which has convenient methods for setting ChromeDriver-specific capabilities. we can then pass the ChromeOptions object into the ChromeDriver constructor

Since Selenium version 3.6.0, the ChromeOptions class in Java also implements the Capabilities interface, allowing you to specify other WebDriver capabilities not specific to ChromeDriver.

ChromeOptions options = new ChromeOptions();

// Add the WebDriver proxy capability.

Proxy proxy = new Proxy();

proxy.setHttpProxy("myhttpproxy:3337");

options.setCapability("proxy", proxy);

// Add a ChromeDriver-specific capability.

options.addExtensions(new File("/path/to/extension.crx"));

ChromeDriver driver = new ChromeDriver(options);

**Using DesiredCapabilities**

To use DesiredCapabilities, you need to know the name of the capability and the type of value it takes. See the full list further below.

**Common use cases**

**Use custom profile (also called user data directory)**

By default, ChromeDriver will create a new temporary profile for each session. At times you may want to set special preferences or just use a custom profile altogether. If the former, you can use the 'chrome.prefs' capability (described later below) to specify preferences that will be applied after Chrome starts. If the latter, you can use the user-data-dir Chrome command-line switch to tell Chrome which profile to use:

ChromeOptions options = new ChromeOptions();

options.addArguments("user-data-dir=/path/to/your/custom/profile");

You can create your own custom profile by just running Chrome (on the command-line or through ChromeDriver) with the user-data-dir switch set to some new directory. If the path doesn't exist, Chrome will create a new profile in the specified location. You can then modify the profile settings as desired, and ChromeDriver can use the profile in the future. Open chrome://version in the browser to see what profile Chrome is using.  
**Start Chrome maximized**

ChromeOptions options = new ChromeOptions();

options.addArguments("start-maximized");

**Using a Chrome executable in a non-standard location**

ChromeOptions options = new ChromeOptions();

options.setBinary("/path/to/other/chrome/binary");

**Set a Chrome preference**

ChromeOptions options = new ChromeOptions();

Map<String, Object> prefs = new HashMap<String, Object>();

prefs.put("profile.default\_content\_settings.popups", 0);

options.setExperimentalOption("prefs", prefs);

**List of recognized capabilities**

This is a list of all the WebDriver-standard capabilities that ChromeDriver supports:

**chromeOptions object:** This is a list of all the Chrome-specific desired capabilities, which all are under the chromeOptions dictionary.

If possible, use the ChromeOptions class instead of specifying these directly.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| args | list of strings | List of command-line arguments to use when starting Chrome. Arguments with an associated value should be separated by a '=' sign (e.g., ['start-maximized', 'user-data-dir=/tmp/temp\_profile']). See here for a list of Chrome arguments. |
| binary | string | Path to the Chrome executable to use (on Mac OS X, this should be the actual binary, not just the app. e.g., '/Applications/Google Chrome.app/Contents/MacOS/Google Chrome') |
| extensions | list of strings | A list of Chrome extensions to install on startup. Each item in the list should be a base-64 encoded packed Chrome extension (.crx) |
| localState | dictionary | A dictionary with each entry consisting of the name of the preference and its value. These preferences are applied to the Local State file in the user data folder. |
| prefs | dictionary | A dictionary with each entry consisting of the name of the preference and its value. These preferences are only applied to the user profile in use. See the 'Preferences' file in Chrome's user data directory for examples. |
| detach | Boolean (Default is false) | If false, Chrome will be quit when ChromeDriver is killed, regardless of whether the session is quit. If true, Chrome will only be quit if the session is quit (or closed). Note, if true, and the session is not quit, ChromeDriver cannot clean up the temporary user data directory that the running Chrome instance is using. |
| debuggerAddress | string | An address of a Chrome debugger server to connect to, in the form of <hostname/ip:port>, e.g. '127.0.0.1:38947' |
| excludeSwitches | list of strings | List of Chrome command line switches to exclude that ChromeDriver by default passes when starting Chrome.  Do not prefix switches with --. |
| minidumpPath | string | Directory to store Chrome minidumps . (Supported only on Linux.) |
| mobileEmulation | dictionary | A dictionary with either a value for “deviceName,” or values for “deviceMetrics” and “userAgent.” Refer to Mobile Emulation for more information. |
| perfLoggingPrefs | dictionary | An optional dictionary that specifies performance logging preferences. See below for more information. |
| windowTypes | list of strings | A list of window types that will appear in the list of window handles. For access to <webview> elements, include "webview" in this list. |

**perfLoggingPrefs object:** The perfLoggingPrefs dictionary has the following format (all keys are optional):

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Default** | **Description** |
| enableNetwork | boolean | TRUE | Whether or not to collect events from Network domain. |
| enablePage | boolean | TRUE | Whether or not to collect events from Page domain. |
| traceCategories | string | (empty) | A comma-separated string of Chrome tracing categories for which trace events should be collected. An unspecified or empty string disables tracing. |
| bufferUsageReportingInterval | positive integer | 1000 | The requested number of milliseconds between DevTools trace buffer usage events. For example, if 1000, then once per second, DevTools will report how full the trace buffer is. If a report indicates the buffer usage is 100%, a warning will be issued. |

**Returned Capabilities :** This is a list of all the Chrome-specific returned capabilities. (i.e., what ChromeDriver returns when you create a new session)

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| chrome.chromedriverVersion | string | version of ChromeDriver |
| userDataDir | string | path to user data directory that Chrome is using; note, this is inside a 'chrome' dictionary |

**Object structures**

**Proxy JSON Object** : A JSON object describing a Proxy configuration.

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Description** |
| proxyType | string | (Required) The type of proxy being used. Possible values are: direct - A direct connection - no proxy in use, manual - Manual proxy settings configured, e.g. setting a proxy for HTTP, a proxy for FTP, etc, pac - Proxy autoconfiguration from a URL, autodetect - Proxy autodetection, probably with WPAD, system - Use system settings |
| proxyAutoconfigUrl | string | (Required if proxyType == pac, Ignored otherwise) Specifies the URL to be used for proxy autoconfiguration. Expected format example: http://hostname.com:1234/pacfile |
| ftpProxy, httpProxy, sslProxy, socksProxy | string | (Optional, Ignored if proxyType != manual) Specifies the proxies to be used for FTP, HTTP, HTTPS and SOCKS requests respectively. Behaviour is undefined if a request is made, where the proxy for the particular protocol is undefined, if proxyType is manual. Expected format example: hostname.com:1234 |
| socksUsername | string | (Optional, Ignored if proxyType != manual and socksProxy is not set) Specifies SOCKS proxy username. |
| socksPassword | string | (Optional, Ignored if proxyType != manual and socksProxy is not set) Specifies SOCKS proxy password. |
| noProxy | string | (Optional, Ignored if proxyType != manual) Specifies proxy bypass addresses. Format is driver specific. |

**LoggingPreferences JSON object**

A JSON object describing the logging level of different components in the browser, the driver, or any intermediary WebDriver servers.

Available values for most loggers are "OFF", "SEVERE", "WARNING", "INFO", "CONFIG", "FINE", "FINER", "FINEST", "ALL".

This produces a JSON object looking something like: {"loggingPrefs": {"driver": "INFO", "server": "OFF", "browser": "FINE"}}.

See the documentation of each driver for what browser specific logging components are available.

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Description** |
| component | string | How verbose the logging should be. |

**FirefoxProfile settings**

Preferences accepted by the FirefoxProfile with special meaning, in the WebDriver API:

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Description** |
| webdriver\_accept\_untrusted\_certs | boolean | Whether to trust all SSL certificates. TODO: Maybe in some way different to the acceptSslCerts or trustAllSSLCertificates capabilities. |
| webdriver\_assume\_untrusted\_issuer | boolean | Whether to trust all SSL certificate issuers. TODO: Maybe in some way different to the acceptSslCerts or trustAllSSLCertificates capabilities. |
| webdriver.log.driver | string | Level at which to log FirefoxDriver logging statements to a temporary file, so that they can be retrieved by a getLogs command. Available options; DEBUG, INFO, WARNING, ERROR, OFF. Defaults to OFF. |
| webdriver.log.file | string | Path to file to which to copy firefoxdriver logging output. Defaults to no file (like /dev/null). |
| webdriver.load.strategy | string | Experimental API. Defines different strategies for how long to wait until a page is loaded. Values: unstable, conservative. Defaults to conservative. |
| webdriver\_firefox\_port | integer | Port to listen on for WebDriver commands. Defaults to 7055. |