|  |  |
| --- | --- |
| **category / JsonWireProtocol method** | **wd methods** |
| GET [/status](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/status) Query the server's current status. | status(cb) -> cb(err, status) |
| POST [/session](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session) Create a new session. | init(desired, cb) -> cb(err, sessionID, capabilities) Initialize the browser. capabilities return may be absent, depending on driver. |
| GET [/sessions](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/sessions) Returns a list of the currently active sessions. | sessions(cb) -> cb(err, sessions)  Alternate strategy to get session capabilities from server session list: altSessionCapabilities(cb) -> cb(err, capabilities) |
| GET [/session/:sessionId](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId) Retrieve the capabilities of the specified session. | sessionCapabilities(cb) -> cb(err, capabilities) |
| DELETE [/session/:sessionId](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId) Delete the session. | quit(cb) -> cb(err) Destroy the browser. |
| POST [/session/:sessionId/timeouts](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/timeouts) Configure the amount of time that a particular type of operation can execute for before they are aborted and a |Timeout| error is returned to the client. | setPageLoadTimeout(ms, cb) -> cb(err) (use setImplicitWaitTimeout and setAsyncScriptTimeout to set the other timeouts)  setCommandTimeout(ms, cb) -> cb(err) (this is for Appium only) |
| POST [/session/:sessionId/timeouts/async\_script](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/timeouts/async_script) Set the amount of time, in milliseconds, that asynchronous scripts executed by /session/:sessionId/execute\_async are permitted to run before they are aborted and a |Timeout| error is returned to the client. | setAsyncScriptTimeout(ms, cb) -> cb(err) |
| POST [/session/:sessionId/timeouts/implicit\_wait](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/timeouts/implicit_wait) Set the amount of time the driver should wait when searching for elements. | setImplicitWaitTimeout(ms, cb) -> cb(err) |
| GET [/session/:sessionId/window\_handle](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/window_handle) Retrieve the current window handle. | windowHandle(cb) -> cb(err, handle) |
| GET [/session/:sessionId/window\_handles](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/window_handles) Retrieve the list of all window handles available to the session. | windowHandles(cb) -> cb(err, arrayOfHandles) |
| GET [/session/:sessionId/url](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/url) Retrieve the URL of the current page. | url(cb) -> cb(err, url) |
| POST [/session/:sessionId/url](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/url) Navigate to a new URL. | get(url,cb) -> cb(err) Get a new url. |
| POST [/session/:sessionId/forward](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/forward) Navigate forwards in the browser history, if possible. | forward(cb) -> cb(err) |
| POST [/session/:sessionId/back](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/back) Navigate backwards in the browser history, if possible. | back(cb) -> cb(err) |
| POST [/session/:sessionId/refresh](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/refresh) Refresh the current page. | refresh(cb) -> cb(err) |
| POST [/session/:sessionId/execute](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/execute) Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. | execute(code, args, cb) -> cb(err, result) execute(code, cb) -> cb(err, result) args: script argument array (optional)  Safely execute script within an eval block, always returning: safeExecute(code, args, cb) -> cb(err, result) safeExecute(code, cb) -> cb(err, result) args: script argument array (optional)  Evaluate expression (using execute): eval(code, cb) -> cb(err, value)  Safely evaluate expression, always returning (using safeExecute): safeEval(code, cb) -> cb(err, value) |
| POST [/session/:sessionId/execute\_async](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/execute_async) Inject a snippet of JavaScript into the page for execution in the context of the currently selected frame. | executeAsync(code, args, cb) -> cb(err, result) executeAsync(code, cb) -> cb(err, result) args: script argument array (optional)  Safely execute async script within an eval block, always returning: safeExecuteAsync(code, args, cb) -> cb(err, result) safeExecuteAsync(code, cb) -> cb(err, result) args: script argument array (optional) |
| GET [/session/:sessionId/screenshot](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/screenshot) Take a screenshot of the current page. | takeScreenshot(cb) -> cb(err, screenshot) |
| GET [/session/:sessionId/ime/available\_engines](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/ime/available_engines) List all available engines on the machine. | availableIMEEngines(cb) -> cb(err, engines) |
| GET [/session/:sessionId/ime/active\_engine](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/ime/active_engine) Get the name of the active IME engine. | activeIMEEngine(cb) -> cb(err, activeEngine) |
| GET [/session/:sessionId/ime/activated](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/ime/activated) Indicates whether IME input is active at the moment (not if it's available). | activatedIMEEngine(cb) -> cb(err, active) |
| POST [/session/:sessionId/ime/deactivate](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/ime/deactivate) De-activates the currently-active IME engine. | deactivateIMEEngine(cb) -> cb(err) |
| POST [/session/:sessionId/ime/activate](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/ime/activate) Make an engines that is available (appears on the listreturned by getAvailableEngines) active. | activateIMEEngine(cb, engine) -> cb(err) |
| POST [/session/:sessionId/frame](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/frame) Change focus to another frame on the page. | frame(frameRef, cb) -> cb(err) |
| POST [/session/:sessionId/window](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/window) Change focus to another window. | window(name, cb) -> cb(err) |
| DELETE [/session/:sessionId/window](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId/window) Close the current window. | close(cb) -> cb(err) |
| POST [/session/:sessionId/window/:windowHandle/size](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/window/:windowHandle/size) Change the size of the specified window. | windowSize(handle, width, height, cb) -> cb(err)  setWindowSize(width, height, handle, cb) -> cb(err) setWindowSize(width, height, cb) -> cb(err) width: width in pixels to set size to height: height in pixels to set size to handle: window handle to set size for (optional, default: 'current') |
| GET [/session/:sessionId/window/:windowHandle/size](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/window/:windowHandle/size) Get the size of the specified window. | getWindowSize(handle, cb) -> cb(err, size) getWindowSize(cb) -> cb(err, size) handle: window handle to get size (optional, default: 'current') |
| POST [/session/:sessionId/window/:windowHandle/position](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/window/:windowHandle/position) Change the position of the specified window. | setWindowPosition(x, y, handle, cb) -> cb(err) setWindowPosition(x, y, cb) -> cb(err) x: the x-coordinate in pixels to set the window position y: the y-coordinate in pixels to set the window position handle: window handle to set position for (optional, default: 'current') |
| GET [/session/:sessionId/window/:windowHandle/position](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/window/:windowHandle/position) Get the position of the specified window. | getWindowPosition(handle, cb) -> cb(err, position) getWindowPosition(cb) -> cb(err, position) handle: window handle to get position (optional, default: 'current') |
| POST [/session/:sessionId/window/:windowHandle/maximize](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/window/:windowHandle/maximize) Maximize the specified window if not already maximized. | maximize(handle, cb) -> cb(err) |
| GET [/session/:sessionId/cookie](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/cookie) Retrieve all cookies visible to the current page. | allCookies() -> cb(err, cookies) |
| POST [/session/:sessionId/cookie](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/cookie) Set a cookie. | setCookie(cookie, cb) -> cb(err) cookie example: {name:'fruit', value:'apple'} Optional cookie fields: path, domain, secure, expiry |
| DELETE [/session/:sessionId/cookie](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId/cookie) Delete all cookies visible to the current page. | deleteAllCookies(cb) -> cb(err) |
| DELETE [/session/:sessionId/cookie/:name](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId/cookie/:name) Delete the cookie with the given name. | deleteCookie(name, cb) -> cb(err) |
| GET [/session/:sessionId/source](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/source) Get the current page source. | source(cb) -> cb(err, source) |
| GET [/session/:sessionId/title](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/title) Get the current page title. | title(cb) -> cb(err, title) |
| POST [/session/:sessionId/element](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element) Search for an element on the page, starting from the document root. | element(using, value, cb) -> cb(err, element)  elementByClassName(value, cb) -> cb(err, element) elementByCssSelector(value, cb) -> cb(err, element) elementById(value, cb) -> cb(err, element) elementByName(value, cb) -> cb(err, element) elementByLinkText(value, cb) -> cb(err, element) elementByPartialLinkText(value, cb) -> cb(err, element) elementByTagName(value, cb) -> cb(err, element) elementByXPath(value, cb) -> cb(err, element) elementByCss(value, cb) -> cb(err, element) elementByIosUIAutomation(value, cb) -> cb(err, element) elementByAndroidUIAutomator(value, cb) -> cb(err, element) elementByAccessibilityId(value, cb) -> cb(err, element) |
| POST [/session/:sessionId/elements](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/elements) Search for multiple elements on the page, starting from the document root. | elements(using, value, cb) -> cb(err, elements)  elementsByClassName(value, cb) -> cb(err, elements) elementsByCssSelector(value, cb) -> cb(err, elements) elementsById(value, cb) -> cb(err, elements) elementsByName(value, cb) -> cb(err, elements) elementsByLinkText(value, cb) -> cb(err, elements) elementsByPartialLinkText(value, cb) -> cb(err, elements) elementsByTagName(value, cb) -> cb(err, elements) elementsByXPath(value, cb) -> cb(err, elements) elementsByCss(value, cb) -> cb(err, elements) elementsByIosUIAutomation(value, cb) -> cb(err, elements) elementsByAndroidUIAutomator(value, cb) -> cb(err, elements) elementsByAccessibilityId(value, cb) -> cb(err, elements)  Retrieve an element avoiding not found exception and returning null instead: elementOrNull(using, value, cb) -> cb(err, element)  elementByClassNameOrNull(value, cb) -> cb(err, element) elementByCssSelectorOrNull(value, cb) -> cb(err, element) elementByIdOrNull(value, cb) -> cb(err, element) elementByNameOrNull(value, cb) -> cb(err, element) elementByLinkTextOrNull(value, cb) -> cb(err, element) elementByPartialLinkTextOrNull(value, cb) -> cb(err, element) elementByTagNameOrNull(value, cb) -> cb(err, element) elementByXPathOrNull(value, cb) -> cb(err, element) elementByCssOrNull(value, cb) -> cb(err, element) elementByIosUIAutomationOrNull(value, cb) -> cb(err, element) elementByAndroidUIAutomatorOrNull(value, cb) -> cb(err, element) elementByAccessibilityIdOrNull(value, cb) -> cb(err, element)  Retrieve an element avoiding not found exception and returning undefined instead: elementIfExists(using, value, cb) -> cb(err, element)  elementByClassNameIfExists(value, cb) -> cb(err, element) elementByCssSelectorIfExists(value, cb) -> cb(err, element) elementByIdIfExists(value, cb) -> cb(err, element) elementByNameIfExists(value, cb) -> cb(err, element) elementByLinkTextIfExists(value, cb) -> cb(err, element) elementByPartialLinkTextIfExists(value, cb) -> cb(err, element) elementByTagNameIfExists(value, cb) -> cb(err, element) elementByXPathIfExists(value, cb) -> cb(err, element) elementByCssIfExists(value, cb) -> cb(err, element) elementByIosUIAutomationIfExists(value, cb) -> cb(err, element) elementByAndroidUIAutomatorIfExists(value, cb) -> cb(err, element) elementByAccessibilityIdIfExists(value, cb) -> cb(err, element)  Check if element exists: hasElement(using, value, cb) -> cb(err, boolean)  hasElementByClassName(value, cb) -> cb(err, boolean) hasElementByCssSelector(value, cb) -> cb(err, boolean) hasElementById(value, cb) -> cb(err, boolean) hasElementByName(value, cb) -> cb(err, boolean) hasElementByLinkText(value, cb) -> cb(err, boolean) hasElementByPartialLinkText(value, cb) -> cb(err, boolean) hasElementByTagName(value, cb) -> cb(err, boolean) hasElementByXPath(value, cb) -> cb(err, boolean) hasElementByCss(value, cb) -> cb(err, boolean) hasElementByIosUIAutomation(value, cb) -> cb(err, boolean) hasElementByAndroidUIAutomator(value, cb) -> cb(err, boolean) hasElementByAccessibilityId(value, cb) -> cb(err, boolean) |
| POST [/session/:sessionId/element/active](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/active) Get the element on the page that currently has focus. | active(cb) -> cb(err, element) |
| POST [/session/:sessionId/element/:id/element](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/element) Search for an element on the page, starting from the identified element. | element.element(using, value, cb) -> cb(err, element)  element.elementByClassName(value, cb) -> cb(err, element) element.elementByCssSelector(value, cb) -> cb(err, element) element.elementById(value, cb) -> cb(err, element) element.elementByName(value, cb) -> cb(err, element) element.elementByLinkText(value, cb) -> cb(err, element) element.elementByPartialLinkText(value, cb) -> cb(err, element) element.elementByTagName(value, cb) -> cb(err, element) element.elementByXPath(value, cb) -> cb(err, element) element.elementByCss(value, cb) -> cb(err, element) element.elementByIosUIAutomation(value, cb) -> cb(err, element) element.elementByAndroidUIAutomator(value, cb) -> cb(err, element) element.elementByAccessibilityId(value, cb) -> cb(err, element) |
| POST [/session/:sessionId/element/:id/elements](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/elements) Search for multiple elements on the page, starting from the identified element. | element.elements(using, value, cb) -> cb(err, elements)  element.elementsByClassName(value, cb) -> cb(err, elements) element.elementsByCssSelector(value, cb) -> cb(err, elements) element.elementsById(value, cb) -> cb(err, elements) element.elementsByName(value, cb) -> cb(err, elements) element.elementsByLinkText(value, cb) -> cb(err, elements) element.elementsByPartialLinkText(value, cb) -> cb(err, elements) element.elementsByTagName(value, cb) -> cb(err, elements) element.elementsByXPath(value, cb) -> cb(err, elements) element.elementsByCss(value, cb) -> cb(err, elements) element.elementsByIosUIAUtomation(value, cb) -> cb(err, elements) element.elementsByAndroidUIAutomator(value, cb) -> cb(err, elements) element.elementsByAccessibilityId(value, cb) -> cb(err, elements) |
| POST [/session/:sessionId/element/:id/click](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/click) Click on an element. | clickElement(element, cb) -> cb(err)  element.click(cb) -> cb(err) |
| POST [/session/:sessionId/element/:id/submit](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/submit) Submit a FORM element. | submit(element, cb) -> cb(err) Submit a `FORM` element.  element.submit(cb) -> cb(err) |
| GET [/session/:sessionId/element/:id/text](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/text) Returns the visible text for the element. | text(element, cb) -> cb(err, text) element: specific element, 'body', or undefined  element.text(cb) -> cb(err, text)  Check if text is present: textPresent(searchText, element, cb) -> cb(err, boolean) element: specific element, 'body', or undefined  element.textPresent(searchText, cb) -> cb(err, boolean) |
| POST [/session/:sessionId/element/:id/value](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/value) Send a sequence of key strokes to an element. | type(element, keys, cb) -> cb(err) Type keys (all keys are up at the end of command). special key map: wd.SPECIAL\_KEYS (see lib/special-keys.js)  element.type(keys, cb) -> cb(err)  element.keys(keys, cb) -> cb(err) |
| POST [/session/:sessionId/keys](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/keys) Send a sequence of key strokes to the active element. | keys(keys, cb) -> cb(err) Press keys (keys may still be down at the end of command). special key map: wd.SPECIAL\_KEYS (see lib/special-keys.js) |
| GET [/session/:sessionId/element/:id/name](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/name) Query for an element's tag name. | getTagName(element, cb) -> cb(err, name)  element.getTagName(cb) -> cb(err, name) |
| POST [/session/:sessionId/element/:id/clear](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/element/:id/clear) Clear a TEXTAREA or text INPUT element's value. | clear(element, cb) -> cb(err)  element.clear(cb) -> cb(err) |
| GET [/session/:sessionId/element/:id/selected](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/selected) Determine if an OPTION element, or an INPUT element of type checkbox or radiobutton is currently selected. | isSelected(element, cb) -> cb(err, selected)  element.isSelected(cb) -> cb(err, selected) |
| GET [/session/:sessionId/element/:id/enabled](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/enabled) Determine if an element is currently enabled. | isEnabled(element, cb) -> cb(err, enabled)  element.isEnabled(cb) -> cb(err, enabled) |
| GET [/session/:sessionId/element/:id/attribute/:name](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/attribute/:name) Get the value of an element's attribute. | getAttribute(element, attrName, cb) -> cb(err, value)  element.getAttribute(attrName, cb) -> cb(err, value)  Get element value (in value attribute): getValue(element, cb) -> cb(err, value)  element.getValue(cb) -> cb(err, value) |
| GET [/session/:sessionId/element/:id/equals/:other](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/equals/:other) Test if two element IDs refer to the same DOM element. | element.equals(other, cb) -> cb(err, value)  equalsElement(element, other , cb) -> cb(err, value) |
| GET [/session/:sessionId/element/:id/displayed](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/displayed) Determine if an element is currently displayed. | isDisplayed(element, cb) -> cb(err, displayed)  element.isDisplayed(cb) -> cb(err, displayed) |
| GET [/session/:sessionId/element/:id/location](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/location) Determine an element's location on the page. | getLocation(element, cb) -> cb(err, location)  element.getLocation(cb) -> cb(err, location)  element.getLocationInView(cb) -> cb(err, location) |
| GET [/session/:sessionId/element/:id/location\_in\_view](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/location_in_view) Determine an element's location on the screen once it has been scrolled into view. | getLocationInView(element, cb) -> cb(err, location) |
| GET [/session/:sessionId/element/:id/size](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/size) Determine an element's size in pixels. | getSize(element, cb) -> cb(err, size)  element.getSize(cb) -> cb(err, size) |
| GET [/session/:sessionId/element/:id/css/:propertyName](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/element/:id/css/:propertyName) Query the value of an element's computed CSS property. | getComputedCss(element, cssProperty , cb) -> cb(err, value)  element.getComputedCss(cssProperty , cb) -> cb(err, value) |
| GET [/session/:sessionId/orientation](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/orientation) Get the current browser orientation. | getOrientation(cb) -> cb(err, orientation) |
| POST [/session/:sessionId/orientation](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/orientation) Set the browser orientation. | setOrientation(orientation) -> cb(err) |
| GET [/session/:sessionId/alert\_text](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/alert_text) Gets the text of the currently displayed JavaScript alert(), confirm(), or prompt() dialog. | alertText(cb) -> cb(err, text) |
| POST [/session/:sessionId/alert\_text](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/alert_text) Sends keystrokes to a JavaScript prompt() dialog. | alertKeys(keys, cb) -> cb(err) |
| POST [/session/:sessionId/accept\_alert](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/accept_alert) Accepts the currently displayed alert dialog. | acceptAlert(cb) -> cb(err) |
| POST [/session/:sessionId/dismiss\_alert](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/dismiss_alert) Dismisses the currently displayed alert dialog. | dismissAlert(cb) -> cb(err) |
| POST [/session/:sessionId/moveto](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/moveto) Move the mouse by an offset of the specificed element. | moveTo(element, xoffset, yoffset, cb) -> cb(err) Move to element, element may be null, xoffset and y offset are optional.  element.moveTo(xoffset, yoffset, cb) -> cb(err) xoffset and y offset are optional. |
| POST [/session/:sessionId/click](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/click) Click any mouse button (at the coordinates set by the last moveto command). | click(button, cb) -> cb(err) Click on current element. Buttons: {left: 0, middle: 1 , right: 2} |
| POST [/session/:sessionId/buttondown](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/buttondown) Click and hold the left mouse button (at the coordinates set by the last moveto command). | buttonDown(button ,cb) -> cb(err) button is optional. {LEFT = 0, MIDDLE = 1 , RIGHT = 2}. LEFT if not specified. |
| POST [/session/:sessionId/buttonup](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/buttonup) Releases the mouse button previously held (where the mouse is currently at). | buttonUp(button, cb) -> cb(err) button is optional. {LEFT = 0, MIDDLE = 1 , RIGHT = 2}. LEFT if not specified. |
| POST [/session/:sessionId/doubleclick](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/doubleclick) Double-clicks at the current mouse coordinates (set by moveto). | doubleclick(cb) -> cb(err)  element.doubleClick(cb) -> cb(err) |
| POST [/session/:sessionId/touch/click](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/touch/click) Single tap on the touch enabled device. | tap(element) -> cb(err) Taps element  element.tap(cb) -> cb(err) |
| POST [/session/:sessionId/touch/flick](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/touch/flick) Flick on the touch screen using finger motion events. | flick(xSpeed, ySpeed, swipe, cb) -> cb(err) Flicks, starting anywhere on the screen. flick(element, xoffset, yoffset, speed, cb) -> cb(err) Flicks, starting at element center.  element.flick(xoffset, yoffset, speed, cb) -> cb(err) |
| GET [/session/:sessionId/location](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/location) Get the current geo location. | getGeoLocation(cb) -> cb(err, geoLocationObj) |
| POST [/session/:sessionId/location](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/location) Set the current geo location. | setGeoLocation(lat, lon, alt, cb) -> cb(err) |
| POST [/session/:sessionId/local\_storage](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/local_storage) Set the storage item for the given key. | setLocalStorageKey(key, value, cb) -> cb(err) # uses safeExecute() due to localStorage bug in Selenium |
| DELETE [/session/:sessionId/local\_storage](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId/local_storage) Clear the storage. | clearLocalStorage(cb) -> cb(err) # uses safeExecute() due to localStorage bug in Selenium |
| GET [/session/:sessionId/local\_storage/key/:key](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/local_storage/key/:key) Get the storage item for the given key. | getLocalStorageKey(key, cb) -> cb(err) # uses safeEval() due to localStorage bug in Selenium |
| DELETE [/session/:sessionId/local\_storage/key/:key](http://code.google.com/p/selenium/wiki/JsonWireProtocol#DELETE_/session/:sessionId/local_storage/key/:key) Remove the storage item for the given key. | removeLocalStorageKey(key, cb) -> cb(err) # uses safeExecute() due to localStorage bug in Selenium |
| POST [/session/:sessionId/log](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/log) Get the log for a given log type. | scroll(xOffset, yOffset, cb) -> cb(err)  log(logType, cb) -> cb(err, arrayOfLogs) |
| GET [/session/:sessionId/log/types](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/log/types) Get available log types. | logTypes(cb) -> cb(err, arrayOfLogTypes) |
| GET [/session/:sessionId/context](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/context) Get the current context (mjsonWire). | currentContext(cb) -> cb(err) |
| POST [/session/:sessionId/context](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/context) Set the current context (mjsonWire). | context(contextRef, cb) -> cb(err, context) |
| GET [/session/:sessionId/contexts](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/contexts) Get a list of the available contexts (mjsonWire). | contexts(cb) -> cb(err, handle) |
| POST [/session/:sessionId/touch/perform](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/touch/perform) Perform touch action (mjsonWire). | performTouchAction(touchAction) -> cb(err) |
| POST [/session/:sessionId/touch/multi/perform](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/touch/multi/perform) Perform multitouch action (mjsonWire). | performMultiAction(element, multiAction) -> cb(err, touchStateObjects) performMultiAction(multiAction) -> cb(err, touchStateObjects)  element.performMultiAction(actions) -> cb(err, touchStateObjects) |
| POST [/session/:sessionId/appium/device/shake](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/shake) Shake device (mjsonWire). | shakeDevice(cb) -> cb(err)  shake(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/lock](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/lock) Lock device (mjsonWire). | lockDevice(seconds, cb) -> cb(err)  lock(seconds, cb) -> cb(err)  unlockDevice(cb) -> cb(err)  unlock(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/keyevent](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/keyevent) Send key event to device (DEPRECATED) (mjsonWire). | deviceKeyEvent(keycode, metastate, cb) -> cb(err) metastate is optional. DEPRECATED: use pressKeycode instead.  pressDeviceKey(keycode, metastate, cb) -> cb(err) metastate is optional. DEPRECATED: use pressKeycode instead. |
| POST [/session/:sessionId/appium/device/press\_keycode](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/press_keycode) Send key event to device (mjsonWire). | pressKeycode(keycode, metastate, cb) -> cb(err) metastate is optional. |
| POST [/session/:sessionId/appium/device/rotate](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/rotate) Rotate device (mjsonWire). | rotateDevice(element, opts, cb) -> cb(err) rotateDevice(opts, cb) -> cb(err) opts is like the following: {x: 114, y: 198, duration: 5, radius: 3, rotation: 220, touchCount: 2}  rotate(element, opts, cb) -> cb(err) rotate(opts, cb) -> cb(err) opts is like the following: {x: 114, y: 198, duration: 5, radius: 3, rotation: 220, touchCount: 2}  element.rotate(opts, cb) -> cb(err) opts is like the following: {x: 114, y: 198, duration: 5, radius: 3, rotation: 220, touchCount: 2} |
| GET [/session/:sessionId/appium/device/current\_activity](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/appium/device/current_activity) Get current activity (mjsonWire). | getCurrentDeviceActivity(cb) -> cb(err)  getCurrentActivity(cb) -> cb(err) |
| GET [/session/:sessionId/appium/device/current\_package](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/appium/device/current_activity) Get current package (mjsonWire). | getCurrentPackage(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/install\_app](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/install_app) Install app (mjsonWire). | installAppOnDevice(appPath, cb) -> cb(err)  installApp(appPath, cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/remove\_app](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/remove_app) Remove app (mjsonWire). | removeAppFromDevice(appId, cb) -> cb(err)  removeApp(appId, cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/app\_installed](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/app_installed) Check if the app is installed (mjsonWire). | isAppInstalledOnDevice(bundleId, cb) -> cb(isInstalled, err)  isAppInstalled(bundleId, cb) -> cb(isInstalled, err) |
| POST [/session/:sessionId/appium/device/push\_file](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/push_file) Push file to device (mjsonWire). | pushFileToDevice(pathOnDevice, base64Data, cb) -> cb(err)  pushFile(pathOnDevice, base64Data, cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/pull\_file](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/pull_file) Pull file from device (mjsonWire). | pullFileFromDevice(pathOnDevice, cb) -> cb(base64EncodedData, err)  pullFile(pathOnDevice, cb) -> cb(base64EncodedData, err) |
| POST [/session/:sessionId/appium/device/pull\_folder](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/pull_folder) Pull folder from device (mjsonWire). | pullFolderFromDevice(pathOnDevice, cb) -> cb(base64EncodedData, err)  pullFolder(pathOnDevice, cb) -> cb(base64EncodedData, err) |
| POST [/session/:sessionId/appium/device/toggle\_airplane\_mode](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/toggle_airplane_mode) Toggle airplane mode (mjsonWire). | toggleAirplaneModeOnDevice(cb) -> cb(err)  toggleAirplaneMode(cb) -> cb(err)  toggleFlightMode(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/toggle\_wifi](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/toggle_wifi) Toggle wifi (mjsonWire). | toggleWiFiOnDevice(cb) -> cb(err)  toggleWiFi(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/toggle\_location\_services](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/toggle_location_services) Toggle location services (mjsonWire). | toggleLocationServicesOnDevice(cb) -> cb(err)  toggleLocationServices(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/toggle\_data](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/toggle_data) Toggle data (mjsonWire). | toggleDataOnDevice(cb) -> cb(err)  toggleData(cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/start\_activity](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/start_activity) Start an Android activity (mjsonWire). | startActivity(options, cb) -> cb(err) Start an arbitrary Android activity during a session. The 'options' parameter should implement the interface {appPackage, appActivity, [appWaitPackage], [appWaitActivity]}. |
| POST [/session/:sessionId/appium/app/launch](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/launch) Launch app (mjsonWire). | launchApp(cb) -> cb(err) |
| POST [/session/:sessionId/appium/app/close](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/close) Close app (mjsonWire). | closeApp(cb) -> cb(err) |
| POST [/session/:sessionId/appium/app/reset](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/reset) Reset app (mjsonWire). | resetApp(cb) -> cb(err) |
| POST [/session/:sessionId/appium/app/background](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/background) Background app (mjsonWire). | backgroundApp(seconds, cb) -> cb(err) |
| POST [/session/:sessionId/appium/app/end\_test\_coverage](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/end_test_coverage) End test coverage (mjsonWire). | endTestCoverageForApp(intentToBroadcast, pathOnDevice) -> cb(base64Data,err)  endTestCoverage(intentToBroadcast, pathOnDevice) -> cb(base64Data,err)  endCoverage(intentToBroadcast, pathOnDevice) -> cb(base64Data,err) |
| POST [/session/:sessionId/appium/app/complex\_find](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/complex_find) Find within app (mjsonWire). | complexFindInApp(selector) -> cb(element(s)) Return a single element or an elements array depending on selector  complexFind(selector) -> cb(element(s)) Return a single element or an elements array depending on selector |
| POST [/session/:sessionId/appium/app/strings](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/app/strings) Retrieve app strings (mjsonWire). | getAppStrings(cb) -> cb(err) |
| POST [/session/:sessionId/appium/element/:elementId?/value](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/element/:elementId?/value) Set element immediate value (mjsonWire). | setImmediateValueInApp(element, value, cb) -> cb(err)  setImmediateValue(element, value, cb) -> cb(err)  element.setImmediateValueInApp(value, cb) -> cb(err)  element.setImmediateValue(value, cb) -> cb(err) |
| GET [/session/:sessionId/network\_connection](http://code.google.com/p/selenium/wiki/JsonWireProtocol#GET_/session/:sessionId/network_connection) Get appium selendroid network connection type (mjsonWire). | getNetworkConnection(cb) -> cb(err, networkConnectionInfo) |
| POST [/session/:sessionId/network\_connection](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/network_connection) Set appium selendroid network connection type (mjsonWire). | setNetworkConnection(type, cb) -> cb(err) |
| POST [/session/:sessionId/appium/device/hide\_keyboard](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/hide_keyboard) Hide keyboard (mjsonWire). | hideKeyboard() -> cb(err) hideKeyboard(keyName, cb) -> cb(err) hideKeyboard({strategy: 'pressKey', key:'<key>'}) -> cb(err) hideKeyboard({strategy: 'tapOutside'}) -> cb(err) |
| POST [/session/:sessionId/appium/device/open\_notifications](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/open_notifications) Open Notifications (mjsonWire). | openNotifications(cb) -> cb(err) |
| POST [/session/:sessionId/appium/simulator/touch\_id](http://code.google.com/p/selenium/wiki/JsonWireProtocol#POST_/session/:sessionId/appium/device/simulator/touch_id) TouchID (mjsonWire). | touchId(cb) -> cb(err) |
| extra | attach(sessionID, cb) -> cb(err) Connect to an already-active session. |
| extra | detach(cb) -> cb(err) Detach from the current session. |
| extra |  |
| extra | Retrieves the current session id. getSessionId(cb) -> cb(err, sessionId) getSessionId() |
| extra | Opens a new window (using Javascript window.open): newWindow(url, name, cb) -> cb(err) newWindow(url, cb) -> cb(err) name: optional window name Window can later be accessed by name with the window method, or by getting the last handle returned by the windowHandles method. |
| extra | windowName(cb) -> cb(err, name) |
| extra | configureHttp(opts) opts example: {timeout:60000, retries: 3, 'retryDelay': 15, baseUrl='http://example.com/'} more info in README. |
| extra | waitFor(asserter, timeout, pollFreq, cb) -> cb(err, return\_value) timeout and pollFreq are optional (default 1000ms/200ms) waitFor(opts, cb) -> cb(err) opts with the following fields: timeout, pollFreq, asserter. asserter like: function(browser , cb) -> cb(err, satisfied, return\_value) |
| extra | waitForElement(using, value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElement(using, value, timeout, pollFreq, cb) -> cb(err, el) timeout and pollFreq are optional (default 1000ms/200ms) waitForElement(using, value, opts, cb) -> cb(err, el) opts with the following fields: timeout, pollFreq, asserter. asserter like: function(element , cb) -> cb(err, satisfied, el) |
| extra | waitForElements(using, value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElements(using, value, timeout, pollFreq, cb) -> cb(err, els) timeout and pollFreq are optional (default 1000ms/200ms) waitForElements(using, value, opts, cb) -> cb(err, els) opts with the following fields: timeout, pollFreq, asserter. asserter like: function(element , cb) -> cb(err, satisfied, el) |
| extra | saveScreenshot(path, cb) -> cb(err, filePath) path maybe a full file path, a directory path (finishing with /), the screenshot name, or left blank (will create a file in the system temp dir). |
| extra | waitForElementByClassName(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByCssSelector(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementById(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByName(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByLinkText(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByPartialLinkText(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByTagName(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByXPath(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByCss(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByIosUIAutomation(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByAndroidUIAutomator(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementByAccessibilityId(value, asserter, timeout, pollFreq, cb) -> cb(err, el) asserter, timeout, pollFreq are optional, opts may be passed instead, as in waitForElement. |
| extra | waitForElementsByClassName(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByCssSelector(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsById(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByName(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByLinkText(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByPartialLinkText(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByTagName(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByXPath(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByCss(value, asserter, timeout, pollFreq, cb) -> cb(err, els) waitForElementsByIosUIAutomation(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementsByAndroidUIAutomator(value, asserter, timeout, pollFreq, cb) -> cb(err, el) waitForElementsByAccessibilityId(value, asserter, timeout, pollFreq, cb) -> cb(err, el) asserter, timeout, pollFreq are optional, opts may be passed instead, as in waitForElements. |
| extra | Retrieves the pageIndex element (added for Appium): getPageIndex(element, cb) -> cb(err, pageIndex) |
| extra | Uploads a local file using undocumented POST /session/:sessionId/file uploadFile(filepath, cb) -> cb(err, filepath) |
| extra | Waits for JavaScript condition to be true (async script polling within browser): waitForConditionInBrowser(conditionExpr, timeout, pollFreq, cb) -> cb(err, boolean) conditionExpr: condition expression, should return a boolean timeout and pollFreq are optional, default: 1000/100. return true if condition satisfied, error otherwise. |
| extra | sauceJobUpdate(jsonData, cb) -> cb(err) |
| extra | sauceJobStatus(hasPassed, cb) -> cb(err) |
| extra | sleep(ms, cb) -> cb(err) |
| extra | noop(cb) -> cb(err) |
| extra | Equivalent to the python sendKeys binding. Upload file if a local file is detected, otherwise behaves like type. element.sendKeys(keys, cb) -> cb(err) |
| extra | Equivalent to the python sendKeys binding, but replaces texts instead of keeping original. Upload file if a local file is detected, otherwise behaves like type. element.setText(keys, cb) -> cb(err) |
| extra | isVisible(cb) -> cb(err, boolean) |
| extra | element.sleep(ms, cb) -> cb(err) |
| extra | element.noop(cb) -> cb(err) |
| asserter | asserters.nonEmptyText |
| asserter | asserters.textInclude(content) -> Asserter |
| asserter | asserters.isVisible |
| asserter | asserters.isHidden |
| asserter | asserters.jsCondition(jsConditionExpr) -> Asserter jsConditionExpr: js script expression, should evaluate as boolean. |
| wd | wd.configureHttp(opts) opts example: {timeout:60000, retries: 3, 'retryDelay': 15, baseUrl='http://example.com/'} more info in README. |
| wd | wd.showHideDeprecation(boolean) |
| wd | wd.addAsyncMethod(name, func) |
| wd | wd.addElementAsyncMethod(name, func) |
| wd | wd.addPromiseMethod(name, func) |
| wd | wd.addElementPromiseMethod(name, func) |
| wd | wd.addPromiseChainMethod(name, func) |
| wd | wd.addElementPromiseChainMethod(name, func) |
| wd | wd.removeMethod(name, func) |
| MISSING: POST /session/:sessionId/appium/device/is\_locked | isLocked(cb) -> cb(err) |
| MISSING: GET /session/:sessionId/appium/settings | settings(cb) -> cb(err, settingsObject) |
| MISSING: POST /session/:sessionId/appium/settings | updateSettings(settingsObject, cb) -> cb(err) |