End to end testing

End to end instructions about how to get Cucumber testing to work locally, which test steps are currently supported, a list with examples and per example a short explanation. This document is structured as followed:

- NodeJS installation
- Settting up your workspace
- Setup E2E environment
- Starting the webdriver
- Writing a test
- Running the tests
- Supported components
- QA-PAAS
- config.json configuration
- Custom steps
- Examples (supported steps)

NodeJS installation

NodeJS has to be installed. You can download it from https://nodejs.org/en/download/

For Windows, once NodeJS is installed 2 environment variables have to be added. The first variable points towards the folder where the global Node packages are installed.

The global NodeJS packages are normally installed in the %APPDATA%/Roaming/npm directory (e.g. C:/Users/username/AppData/Roaming/npm).

Press the Windows key and search for 'environment variables'. You should see an option called **Edit the system environment variables**. Select this option.

A window called **System Properties** should appear. Navigate to the **Advanced** tab and click on the **Environment Variables** option.

A window called **Environment Variables** should appear. Two environment variables should be added. These variables allows you to execute NodeJS and Protractor calls from any directory. In the **User variables for 'username'** list, select the **Path** variable and click Edit. If the Path variable does not exist, create it by clicking 'New'. Set the **Variable name** as 'Path' and the **Variable value** as the directory path.

This enables the execution of commands from the packages that are installed in your global package list.

The next variable that has to be added has to point towards the root installation folder of NodeJS. Without this, you cannot execute any NodeJS calls outside the NodeJS folder.

Underneath the **System Variables** list, select the **Path** variable and click Edit. Add a new variable and paste the root location of NodeJS (e.g. C:/Program Files/nodejs).

Once this is done, open a command prompt or any similar application and test if NodeJS works by executing the 'node --version' command. If this returns a version number, NodeJS is good to go.

Local installation guide:

Setting up your workspace

To get the tests to work locally, either the jenkins-repository has to be cloned or downloaded as a zip (https://source.servoy.com/projects/SC/repos/qapaas-e2e/browse).

Setup E2E environment

Open a bash or command prompt and navigate to the qapaas-e2e folder and execute the following commands: 'npm install'. This will install all node packages that are required for e2e testing (minus Protractor) After this is done, execute the following command: 'npm install -g protractor@5.4.2'. This will enable the usage of the webdriver (see next step) and allow you to start the tests from any directory. Make sure the -g is also part of the command!

From the same directory, execute the following commands:

- npm list

You should see the following packages installed:

- chai
- cucumber
- protractor-multiple-cucumber-html-reporter-plugin
- expect
- find
- fs-extra
- js-base64
- jsonfile
- lodash
- open
- protractor-cucumber-framework (this should give an *unmet peer dependency* warning)
- restler
- universal analytics (is disabled in E2E testing)
- wd
- wd-bridge

Execute the following command to validate that Protractor is installed globally:

- npm list -g protractor

You should see the Protractor package installed

Now that all the packages are installed, the webdriver can be updated and started.

Starting the webdriver

Start another command prompt or bash and execute the following commands:

- webdriver-manager update
- webdriver-manager start

The 'update' command gets the latest drivers for the most common browsers (Firefox and Chrome) The 'start' command starts the webdriver. The tests cannot run without the webdriver since the webdriver is a communication hatch between Protractor and the browser. Without it the browser cannot receieve commands.

Note: If both commands are not recognized, make sure that the environment variable that is directed towards the global package folder is setup correctly. Execute the following command: 'npm list -g'. This will give you the list of all packages and the installation directory. Make sure the directory matches the environment variable.

Writing a test

Feature files are the test files. Each feature file requires the following minimum basic syntax:

```
Feature: <here you describe the feature, for example "testing solution X">
  Scenario Outline: <here you describe what you want to test of solution X>

  <Here you write your test steps>
  Given I go to https://www.yoursolutionname.com
  When servoy calendar component I want to select <day> <month > <year>
  Then I want to sleep for 5 seconds
  @data_table_servoy
  Examples:
  | day     | month   | year  |
  | 30      | April   | 2016 |
  | 20      | December| 2020 |
```

This test gets repeated twice now due to having data_tables.

Advanced syntax. To make sure parts of the syntax do not have to be copied every time, the usage of the background syntax is recommended.

```
Feature: Testing solution X
Background: Login
    Given I go to https://www.yoursolutionname.com
    <do login steps>

Scenario Outline: Testing component A of solution X
    <Write test steps that test component A>
    @data_table_servoy
    Examples:
    |varA|
    |abc |
    |def |

Scenario Outline: Testing component B of solution X
        <Write test steps that test component B>
    @data_table_servoy
    Examples:
    ||
    ||


Scenario Outline: Testing component C of solution X
    <Write test steps that test component C>
    @data_table_servoy
    Examples:
    ||
    ||
```

The background gets run every time a ‘Scenario outline’ is executed. So the flow will be like this:

```
Background syntax gets executed. This will navigate to the solution and will
login.
Scenario Outline: Testing component A of solution X will get executed twice (due
to data_tables).
Background syntax gets executed.
Scenario Outline: Testing component B of solution X will get executed.
Background syntax gets executed.
Scenario Outline: Testing component C of solution X will get executed.
```

This saves having to use the login syntax three times. Do note, that if a loginSolution is used in your solution, you will be required to logout first. Otherwise the background syntax will get redirected to the main solution and it will fail.

Other examples:

Key press examples Skeleton syntax is ‘When I press {browserAction}’

Filled in examples:

```
When I press enter
When I press escape
When I press backspace
```

Running the tests

Once all these steps are finished, the test can start by navigating to the qapaas-e2e folder with a command prompt/bash. After this, start the test by executing the 'protractor baseConfig.config' command.

Protractor parameters The above command to start the tests can be expanded by adding parameters. Parameters added in the command line will overrule any property set in baseConfig.js. A few common examples:

Using a parameter to set which feature files are tested:

- protractor baseConfig.js --specs=%PROTRACTOR_FEATURES%/mySolution/testSetOne/*.feature In this case an environment variable is used which is set to a specific folder. Inside this folder/mySolution/testSetOne all tests will be executed.

The above example can also be modified like this:

- protractor baseConfig.js --specs=%PROTRACTOR_FEATURES%/mySolution/testSetOne/test_one.feature,%PROTRACTOR_FEATURES%/mySolution/testSetOne/test_two.feature

Using a parameter to set specific variables: Inside baseConfig.js is an object called 'params'. Inside this object, there are a few variables defined. One which is 'testDomainURL'. This variable is used in the 'Given I navigate to the test domain' test step. If you want to create a specific login step, this is a useful variable since you define it once and can re-use it in all the login steps.

To define this variable, the command has to be expanded like this: protractor baseConfig.js --params.testDomainURL=myURL

Jenkins Parameters For security reasons, once the tests are ready to be exported and ran during the automated build, it is highly advised to create 2 Jenkins Parameters. Protractor logs all steps in a log file and if one of the steps is: *When bootstrap data-bootstrapcomponents-textbox component with name myFormName.myPasswordField the text myNotSoHiddenPassword is inserted* It will display your password in the logs.

One way to prevent this is by adding a password variable in the config of the Jenkins job. Every Jenkins variable that is used in the E2E tests should have the following prefix: *E2E_VAR_*

A few complete example would be: *E2E_VAR_PASSWORD E2E_VAR_USERNAME E2E_VAR_TENANT*

Once these are defined in the build, these can be used to login the solution using a custom defined step. See the *custom_step_definition* section for an example on how to use the above variables in a complete step.

Supported components

List of all currently supported steps to test servoy components and to navigate to an URL. Each step will be given one or more examples. Each step starts with the word 'Given', 'Then' or 'When', followed by the name of the component and the data-svy-name (most of the times). After this part the step describes what it will do. There are several steps which do not test a component (like navigation, pressing a button (like ENTER or TAB)), thus not requiring a data-svy-name.

Note:

- Everything between brackets '{}' will be considered a variable. This variable can contain any text but must not be surrounded by quotes or double quotes
- Supported browsers: Chrome / Firefox (partially). Working on support for Firefox and IE
- Most step have a variable called 'elementName'. This variable is the data-svy-name of the component. The data-svy-name is only visible if the solution is deployed with test mode enabled and is always equal to the formname.elementName. Targetting an element by data-svy-name prevents targetting the wrong element since it is always unique.
- For now, certain tests are case sensitive, meaning, the word protractor is not equal to the word Protractor. Most tests do have partial text recognition meaning the word tractor will be found in the word protractor
- Cucumber tests are very gramar dependent. We are working on building an auto-complete functionality to prevent tests from failing due to the incorrectly spelling part of the test step.
- Including a seperate test (like a login test) into a test file is not supported
- This list is not final. Based upon new requirements, new tests will be created
- Custom components are currently not supported
- In certain situations, the data-svy-name of a component is not required. This happens (among others) when clicking on an element inside a combobox or typeahead component. If either of these components is selected, a container with a specific class is created. The tests use these containers to find the correct item. Once the test no longer has one of these components selected, the containers disappear.
- If a page is tested that does not contain angular, the test will crash unless the 'ignoreSynchronization' option is set to true. ignoreSynchronization is an option that allows protractor to wait for Angular promises such as timeouts or HTTP requests. To allow protractor to test a non-angular application, change the onPrepare function of the configuration file (config_chrome.js) to the following:

```
    browser.driver.executeScript(function () {
      return {
        width: window.screen.availWidth,
        height: window.screen.availHeight
      };
    }).then(function (result) {
      browser.driver.manage().window().setSize(result.width, result.height);
    }).then(function(){
      browser.ignoreSynchronization = true;
    });
```

QA-PAAS In order to run the tests during the automated build, the steps below have to be done. In the root of the repository of the main solution, a few folders have to be made:

- jenkins-custom

Inside jenkins-custom, two folders have to be made:

- e2e-test-scripts

Inside the **e2e-test-scripts**, a folder called **features** has to be added. In here, the following files required:

- A file called **config.json**
- At least 1 or more .feature files. They can of course be structured in sub-folders.

Once this is done, the structure should look like this:

Root of the Repository ----| Jenkins Custom --------| Features -------------| Config.json -------------| all your tests (these can be split into sub directories) -------------| custom_step_definitions -----------------| custom_step_definitions.js -------------| custom_scripts (optional) -----------------| custom JS scripts

Config.json specifies which tests are executed, which browsers are used and which properties are set to each browser.

config.json configuration

```
"configurations":{
  "capabilities":{

  },
  "multiCapabilities": [{
    "browserName": "firefox",
    "marionette": true,
    "webdriverClick":false
    }],

  "specs": [
    "./features/folderA/*.feature",
    "./features/folderB/example.feature"
  ],
   "cucumberOpts" : {
        "tags": [
            "~@skip", "@do-not-skip"
        ]
    }
}
```

Tags

Inside config.json you can define tags. Tags defined in here can be placed in a .feature file. In the above example, every feature file that starts with '@skip', will NOT be run. This can be used to structure your feature files to make sure Below of a feature that will always be skipped:

```
@do-not-skip
Feature: Testing my tags
    Scenario Outline: This test will not run

    Given I go to https://salesdemo-dev.sandbox.servoy-
cloud.eu/solutions/SalesDemo/index.html
    Then I want to sleep for 3 seconds



    @data_table_servoy
    Examples:
    ||
    ||
```

svyQAPAAS

To be able to test on the same dataset each time the E2E test starts, the module called svyQAPAAS has to be used. svyQAPAAS is a module that can be used to import/export data from/to the database. One of the crucial E2E test steps that eventually has to be used for proper E2E testing is called 'Given I navigate to the test domain'. Once the E2E test spins up, a new environment with an empty copy of the database will be created just for the E2E test. svyQAPAAS will fill this empty database with data resulting in a proper E2E test with a predictable result. One way to handle data importing is by adding a button on the landing page that fills the database with data. This step only has to be executed once!

Capabilities:

Uses 1 browser. If multicapabilities is used, this property is ignored. Can use the same syntax as multiCapabilities, just not with multiple browsers.

multiCapabilities:

Overrules capabilities. Property to use multiple browsers.

Specs:

Path to the tests that have to be executed. Root of the path is where baseConfig.js is located. As shown in the example, multiple specs are allowed if you want to run specific tests.

Multicapabilities examples:

```
"multiCapabilities": [{
  "browserName": "firefox",
  "marionette": true,
  "webdriverClick":false
}]
```

This will only test firefox.

```
"multiCapabilities": [{
  "browserName": "chrome"
}]
```

This will only test chrome

```
"multiCapabilities": [{
  "browserName": "firefox",
  "marionette": true,
  "webdriverClick":false
}, {
  “browserName”: “chrome”
}]
```

This will test both chrome and firefox.

Count. If required, multiple instances of the same browser can be started.

```
"multiCapabilities": [{
  "browserName": "chrome",
  “count”: 3
}]
```

This will start 3 instances of Chrome.

Headless testing Headless testing, meaning without a UI is excellent to use during the build. This will save a lot of loading time (since no UI has to be rendered)

```
"multiCapabilities": [{
  "browserName": "chrome",
  "chromeOptions": { "args": [ "--headless", "--disable-gpu"] }
  },{
  "browserName": "firefox",
  "moz:firefoxOptions": {
    "args": [ "--headless" ]
  },
  "marionette": true,
  "webdriverClick":false
  }]
```

This will start both chrome and firefox without a UI.

Custom steps Finally, in the features folder, a folder called custom_step_definitions has to be created. In here a file called custom_step_definitions.js has to be made. The file **servoy_step_definitions_chrome.js** has a lot of examples that can help creating custom steps.

The step below is an example on how to add security to your login test. Knowledge about HTML locators is highly recommended in order to create custom steps

```
//To run this step, add the following syntax in your feature file:
//Given I want to login my solution
 Given('I want to login my solution', {timeout: 30 * 1000}, function(callback) {
    //define your elements by using locators in this case the UN/PW fields and the login button
    var un_field = element(by.css("input[data-svy-name='formName.elementName']"));
                var pw_field = element(by.css("data-servoydefault-password[data-svy-name='formName.elementName']")).$('input');
                var loggin_button = element(by.css("data-servoydefault-button[data-svy-name='formName.elementName']")).$('button');

    //Wait for a maximum of 15 seconds until one of the fields is present.
                browser.wait(EC.presenceOf(un_field), 15 * 1000, 'Username field not found!').then(function() {
      //sendKeys is a function that sends text to a field.
      //In this example, the E2E_VAR_USERNAME and E2E_VAR_PASSWORD variables are used to login. See the 'Jenkins Parameters' section for more information
                        sendKeys(un_field, browser.params.E2E_VAR_USERNAME, null);
      sendKeys(pw_field, browser.params.E2E_VAR_PASSWORD, null);
      //clickElement is a function that waits until the element is clickable and present.
                        clickElement(loggin_button).then(function() {
        //wrapup is a test that finalizes the test step
                                wrapUp(callback, null);
      });
       //.catch is a function that is required to have in every single custom step and will catch any errors in the block above. If it is not present, and this step fails, the test
       // will terminate.
                }).catch(function(error){
                        callback(new Error(error.message));
    });
  });
```

Another good custom step is to make sure the test always starts at the correct page. If the user is logged in and the test fails without login out, the login test will fail because it will try to find a US/PW field and a login button. To prevent this from happening, a custom step can be added to prevent this from happening

```
  //Before trying to login, this step can be used to make sure the test starts correctly.
  Then('I want to make sure I am on the landing page', {timeout: 30 * 1000}, function(callback) {
    //First, wait for a maximum of 15 seconds
    browser.wait(EC.urlContains('#login'), 15 * 1000).then(function(){
      wrapUp(callback, null);
    }).catch(function(error) {
      //If the above step fails, that means the test is not on the login page. In
this block the user should logout
      //in this case, the logout button is inside a data-bootstrapextracomponents-
navbar component.
      var nav = element(by.css(`data-bootstrapextracomponents-navbar[data-svy-
name='formName.elementName']`));
      //wait for max 15 seconds until the element appears
                        browser.wait(EC.presenceOf(nav), 15 * 1000, 'Navigation
menu not found!').then(function() {
        //define the logout element.
        var logoutBtn = nav.element(by.cssContainingText('*', 'Logout'));
        browser.wait(EC.presenceOf(logoutBtn), 15 * 1000, 'Logout item not
found!').then(function() {
          //Click on the logout button
          clickElement(logoutBtn).then(function() {
            //here you can do some extra validation like URL validaton in case you
want to be sure everything went correct
            wrapUp(callback, null);
          });
        });
                        }).catch(function() {
                                callback(new Error(error.message));
                        })
    });
  });
```

Once this is setup and the build starts with E2E_ENABLED, the tests defined in config.json will be executed.

Examples

(URL) navigation

Navigates to the given url.

```
Given I go to {URL}
```

Example:

```
Given I go to https://www.servoy.com
```

The same as navigation, but it uses a new tab for this

```
Then I want to navigate to {url} in a new tab
```

Example:

```
Then I want to navigate to https://www.google.com in a new tab
```

Given there are multiple browser tabs open, this test can switch to a specific tab

```
When I want to switch to browser tab {tabNumber}
```

Example:

```
When I want to switch to browser tab 1
```

During the deployment of the application, the URL of the application gets passed as a parameter. This parameter ensures that the test will always navigate to the URL of the deployed application.

```
Given I navigate to the test domain
```

This step compares the current URL with the given URL

```
Then I expect the url to be {browserUrl}
```

Example:

```
Then I expect the url to be https://www.servoy.com
```

This step presses the 'back' button. Browser navigates back to the previous page

```
Then I want to navigate back
```

This step presses 'F5' and refreshes the page

```
Then I want to refresh the page
```

END URL navigation

Sidenav component

This step clicks on a tab of the side navigation component (data-servoyextra-sidenav)

```
When servoy sidenav component with name {elementName} tab {tabName} is clicked
```

Example:

```
When servoy sidenav component with name sideNavForm.sideNavName tab Home is
clicked
```

Expects a tab with the given text to be present

```
Then servoy sidenav component with name {elementName} I expect the tab {tabText}
to be present
```

Example:

```
Then servoy sidenav component with name formname.elementName I expect the tab new
tab to be present
```

Note: This test is case sensitive

End Sidenav component

Generic test steps

This test pauses the test for x seconds

```
Then I want to {activity} for {second} second(s)
```

Examples:

```
Then I want to sleep for 5 seconds
Then I want to go for a coffee for 1 second
```

This presses a button on the keyboard (not case sensitive) As of 12th of july 2017 , the current keys are supported:

- Enter
- Tab

```
When I press {browserAction}
```

Examples:

```
When I press enter
When I press tab
```

This zooms the current browser out/in to a given percentage (only a number is required)

```
Then I want to zoom the page out to {percentage} percent
```

Examples:

```
Then I want to zoom the page out to 95 percent
Then I want to zoom the page out to 110 percent
```

Closes the community edition popup (the popup that is displayed if the user is using a free edition of Servoy)

```
Then I want to close the community edition popup
```

End Generic test steps

---

select2tokenizer component

Clicks the tokenizer component

```
```

```
When servoy select2tokenizer component with name {elementName} is clicked
```

Example:

```
When servoy select2tokenizer component with name orders.tokenizerName is clicked
```

The previous step clicks on the tokenizer. This will open a container with a specific class (select2-container--open among others) This container does not have a data-svy-name, nor is it nested in the select2tokenizer HTML. Meaning, no data-svy-name is required for this step. After that it will click on the given row.

```
When servoy select2tokenizer component record number {rowNumber} is clicked
```

Examples:

```
When servoy select2tokenizer component record number 1 is clicked
When servoy select2tokenizer component record number 9 is clicked
```

Looks inside the tokenizer container and gets the text from row X. The result gets compared with the given text

```
Then servoy select2tokenizer component record number {rowNumber} equals
{recordText}
```

Example:

```
Then servoy select2tokenizer component record number 5 equals recordFiveText
```

Inserts the given text inside the component.

```
When servoy select2tokenizer component with name {elementName} the text
{recordText} is inserted
```

Examples:

```
When servoy select2tokenizer component with name orders.tokenizerName the text
searchText is inserted
```

```
When servoy select2tokenizer component with name orders.tokenizerName the text 123
is inserted
When servoy select2tokenizer component with name orders.tokenizerName the text
abds12!@servoy.com is inserted
When servoy select2tokenizer component with name orders.tokenizerName the text
This is a very long string which can also be inserted is inserted
```

End select2tokenizer component

Typeahead component

Clicks on the typeahead component

```
When servoy default typeahead component with name {elementName} is clicked
```

Example:

```
When servoy default typeahead component with name companies.companyType is clicked
```

Inserts the given text in the typeahead component

```
When servoy default typeahead component with name {elementName} the text {text} is
inserted
```

Example:

```
When servoy default typeahead component with name books.typeaheadName the text
bookName is inserted
```

Looks inside the container of the typeahead, selects the row and compares the text with the given text **Note:** this test is NOT case sensitive

```
When servoy default typeahead component I want row {rowNumber} to equal {text}
```

Example:

```
When servoy default typeahead component I want row fruits.typeaheadName to equal
apple
```

Selects the given row inside the typeahead component

```
When servoy default typeahead component I want to select row number {rowNumber}
```

Example:

```
When servoy default typeahead component I want to select row number 10
```

Validates the value of the typeahead component

```
Then servoy default typeahead component with name {elementName} I want to validate
that the typeahead equals the text {text}
```

Example:

```
Then servoy default typeahead with name companies.companyType I want to validate
that the typeahead equals the text ISV
```

End typeahead component

Input fields

Text field component Inserts the given text in a basic text field component

```
When servoy default input component with name {elementName} the text {input} is
inserted
```

Example:

```
When servoy default input component with name fruits.textfieldName the text grape
is inserted
```

Clears the textfield of all input

```
When servoy default input component with name {elementName} I want to clear the
text field
```

Example:

```
When servoy default input component with name fruits.textfieldName I want to clear
the text field
```

Gets the text of the input field with the given name and compares the result with the given text
Note: this test is NOT case sensitive

```
Then servoy default input component with name {elementName} I want to validate
that the input field equals the text {text}
```

Example:

```
Then servoy default input component with name fruits.textfieldName I want to
validate that the input field equals the text grape
```

Sometimes the value of an input changes after loading data. This can mess up the test since protractor returns the value before it is changed. This test will wait for the value of the input field to equal the given text

```
Then servoy default input component with name {elementName} I want to wait until
the value equals {newValue}
```

Example:

```
Then servoy default input component with name orders.orderTotal I want to wait
until the value equals 500.000
```

End textfield component

Calendar component

Clicks the calendar component with the given name

```
When servoy calendar component with name {elementName} is clicked
```

Example:

```
When servoy calendar component with name contacts.dateCreated is clicked
```

Navigates through the calendar component and selects the given date **Note:** this test will not work without clicking on the calendar component.

```
When servoy calendar component I want to select {day} {month} {year}
```

Examples:

```
When servoy calendar component I want to select 15 june 2017
When servoy calendar component I want to select 15 August 2012
When servoy calendar component I want to select 31 july 2020
```

Clicks the given day. **Note:** this test will not work without clicking on the calendar component.

```
When servoy calendar component I want to select day {day}
```

Example: When servoy calendar component I want to select day 12

END Calendar component

Checkbox component Changes the state of the checkbox to be either checked or unchecked **Note:** {checkboxState} can only be replaced by the word 'checked' or 'unchecked'. Not case sensitive

```
When servoy data-servoydefault-check component with name {elementName} I want it
to be {checkboxOption}
```

Examples:

```
When servoy data-servoydefault-check component with name fruits.isAFruit I want it
to be checked
When servoy data-servoydefault-check component with name fruits.isAFruit I want it
to be unchecked
```

Checks whether the state of the checkbox equals the input Note: {checkboxState} can only be replaced by the word 'checked' or 'unchecked'. Not case sensitive

```
When servoy data-servoydefault-check component with name {elementName} I want to
validate that the checkbox is {checkBoxState}
```

Examples:

```
When servoy data-servoydefault-check component with name orders.isValidated I want
to validate that the checkbox is checked
When servoy data-servoydefault-check component with name orders.isValidated I want
to validate that the checkbox is unchecked
```

End checkbox component

Password component Inserts the given text into the password input component

```
When servoy data-servoydefault-password component with name {elementName} the text
{password} is inserted
```

Example:

```
When servoy data-servoydefault-password component with name login.passwordField
the text mySecretPassword is inserted
```

End password component

Combobox component Clicks on the combobox with the given name

```
When servoy combobox component with name {elementName} is clicked
```

Example:

```
When servoy combobox component with name fruits.fruitList is clicked
```

Looks inside the combobox container and selects item number x

```
Then servoy combobox component I want to select number {comboboxNumber} in the
combobox
```

Example:

```
Then servoy combobox component I want to select number 5 in the combobox
```

Selects an item in the combobox by text

```
Then servoy combobox component I want to select the combobox item with the text
{text}
```

Example:

```
Then servoy combobox component I want to select the combobox item with the text
grapes
```

Validates that the combobox with the given name has the item with the given text selected **Note:** this test is case sensitive

```
When servoy combobox component with name {elementName} I want to validate that the
combobox item with text {text} is selected
```

Example:

```
When servoy combobox component with name companies.companyCountry I want to
validate that the combobox item with text The Netherlands is selected
```

Important to know is that the combobox first has to be clicked. The input field only appears once the combobox is selected. When servoy combobox component the text banana is inserted **Note:** this step will change. Use the 2 example steps to achieve the same result:

```
When servoy combobox component the text {text} is inserted
```

Examples:

```
When servoy combobox component with name {elementName} is clicked
Then servoy combobox component I want to select the combobox item with the text
{text}
```

End combobox component

Button component Default button click

```
When servoy button component with name {elementName} is clicked
```

Example:

```
When servoy button component with name orders.confirmOrder is clicked
```

End button component

Textarea component Inserts the given text in the textarea component

```
When default textarea component with name {elementName} the text {text} is
inserted
```

Example:

```
When default textarea component with name products.productDescription the text
this is a product description is inserted
```

Validates that the value of the textarea component equals the given text

```
Then default textarea component with name {elementName} I want to validate that
the input field equals the text {text}
```

Example:

```
Then default textarea component with name products.productDescription I want to
validate that the input field equals the text this is a product description
```

Textarea component

End Input fields

Label fields

Clicks on a default label with the given name

```
When servoy data-servoydefault-label component with name {elementName} is clicked
```

Example:

```
When servoy data-servoydefault-label component with name orders.infoDialog is
clicked
```

Gets the text of the default label and compares it with the given text

```
Then servoy data-servoydefault-label component with name {elementName} I want to
validate that the label equals the text {text}
```

Examples:

```
Then servoy data-servoydefault-label component with name orders.totalPrice I want
to validate that the label equals the text 40.000
Then servoy data-servoydefault-label component with name orders.totalPrice I want
to validate that the label equals the text $500
```

End label fields

Listbox component

Validates that the listbox contains x amount of components

```
When servoy data-servoydefault-listbox with name {elementName} I want to validate
that there are {rowCount} rows
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
validate that there are 1 rows
When servoy data-servoydefault-listbox with name formName.elementName I want to
validate that there are 0 rows
When servoy data-servoydefault-listbox with name formName.elementName I want to
validate that there are 12 rows
```

Selects element number x inside the listbox

```
When servoy data-servoydefault-listbox with name {elementName} I want to select
row number {rowNumber}
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
select row number {rowNumber}
```

Selects the element with a partial text match

```
When servoy data-servoydefault-listbox with name {elementName} I want to select
the row with the partial text {text}
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
select the row with the partial text {text}
```

Selects the element with an exact text match

```
When servoy data-servoydefault-listbox with name {elementName} I want to select
the row with the exact text {text}
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
select the row with the exact text {text}
```

Validates that an element with a partial text match exists

```
When servoy data-servoydefault-listbox with name {elementName} I want to validate
that a row with the partial text {text} exists
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
validate that a row with the partial text {text} exists
```

Validates that an element with the exact text exists

```
When servoy data-servoydefault-listbox with name {elementName} I want to validate
that a row with the exact text {text} exists
```

Example:

```
When servoy data-servoydefault-listbox with name formName.elementName I want to
validate that a row with the exact text {text} exists
```

End Listbox component

Clicks on the font awesome icon

```
When servoy data-servoyextra-fontawesome component with name {elementName} is
clicked
```

Example:

```
When servoy data-servoyextra-fontawesome component with name formName.elementName
is clicked
```

Bootstrap components

Bootstrap textbox component Inserts the given text inside the bootstrap textbox component with the given name

```
When bootstrap data-bootstrapcomponents-textbox component with name {elementName}
the text {text} is inserted
```

Example:

```
When bootstrap data-bootstrapcomponents-textbox component with name
contacts.firstName the text Protractor is inserted
```

Validates that the text equals the given text

```
Then bootstrap data-bootstrapcomponents-textbox component with name {elementName}
I want to validate that the input field equals the text {text}
```

Example

```
Then bootstrap data-bootstrapcomponents-textbox component with name
formName.elementName I want to validate that the input field equals the text Exact
text
```

End bootstrap textbox component

Bootstrap button component Clicks the bootstrap button component with the given name

```
When bootstrap data-bootstrapcomponents-button component with name {elementName}
is clicked
```

Example:

```
When bootstrap data-bootstrapcomponents-button component with name
orders.confirmOrder is clicked
```

End bootstrap button component

Bootstrap navbar component Selects the main tab of the nav bar

```
When bootstrap data-bootstrapextracomponents-navbar component with name
{elementName} the tab {tabText} is clicked
```

Example:

```
When bootstrap data-bootstrapextracomponents-navbar component with name
formName.elementName the tab Tab text is clicked
```

Note: this test is case sensitive **End bootstrap navbar component**

Bootstrap button group component Selects button number x in the group

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
{elementName} I want to select button number {buttonNumber}
```

Example:

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
formname.elementName I want to select button number 1
When bootstrap data-bootstrapextracomponents-buttons-group component with name
formname.elementName I want to select button number 4
```

Selects the button in the button group with an exact text match

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
{elementName} I want to select the button with the exact text {text}
```

Example:

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
formname.elementName I want to select the button with the exact text Button Text
```

Note: this text is case sensitive

Selects the button in the button group with a partial text match

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
{elementName} I want to select the button with the partial text {text}
```

Example

```
When bootstrap data-bootstrapextracomponents-buttons-group component with name
{elementName} I want to select the button with the partial text Button Text
When bootstrap data-bootstrapextracomponents-buttons-group component with name
{elementName} I want to select the button with the partial text utton Te
```

End bootstrap button group component

Bootstrap select component Clicks the bootstrap select component with the given name

```
When bootstrap data-bootstrapcomponents-select component with name {elementName}
is clicked
```

Example:

```
When bootstrap data-bootstrapcomponents-select component with name
addresses.countries is clicked
```

Selects the row with the given text in the select component with the given name

```
When bootstrap data-bootstrapcomponents-select component with name {elementName} I
want to select the row with {text} as text
```

Example:

```
When bootstrap data-bootstrapcomponents-select component with name
countries.countries I want to select the row with Canada as text
```

Selects the given row in the select component with the given name

```
When bootstrap data-bootstrapcomponents-select component with name {elementName} I
want to select row number {rowNumber}
```

Example:

```
When bootstrap data-bootstrapcomponents-select component with name
countries.countries I want to select row number 5
```

Inserts the given text into the bootstrap select component with the given name

```
When bootstrap data-bootstrapcomponents-select component with name {elementName} I
want to insert {text}
```

Example:

```
When bootstrap data-bootstrapcomponents-select component with name
orders.orderType I want to insert Financial
```

End bootstrap select component

Bootstrap textarea component Inserts the given text into the bootstrap textarea component with the given name

```
When bootstrap data-bootstrapcomponents-textarea component with name {elementName}
the text {text} is inserted
```

Example:

```
When bootstrap data-bootstrapcomponents-textarea component with name
orders.description the text order description is inserted
```

End bootstrap textarea component

Bootstrap checkbox component Changes the state of the checkbox to be either checked or unchecked **Note:** {checkboxState} can only be replaced by the word 'checked' or 'unchecked'. Not case sensitive

```
When bootstrap data-bootstrapcomponents-checkbox component with name {elementName}
I want it to be {checkboxState}
```

Example:

```
When bootstrap data-bootstrapcomponents-checkbox component with name
fruits.isAFruit I want it to be unchecked
```

Validates whether the checkbox is checked or unchecked

```
Then bootstrap data-bootstrapcomponents-checkbox component with name {elementName}
I want to validate that the checkbox is {checkBoxState}
```

Examples:

```
Then bootstrap data-bootstrapcomponents-checkbox component with name
formName.elementName I want to validate that the checkbox is checked
Then bootstrap data-bootstrapcomponents-checkbox component with name
formName.elementName I want to validate that the checkbox is unchecked
```

Validates that the checkbox its text has an exact match with the given text

```
Then bootstrap data-bootstrapcomponents-checkbox component with name {elementName}
I want to validate that the checkbox label equals the text {text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-checkbox component with name
formName.elementName I want to validate that the checkbox label equals the text
Exact Text
```

Note: this test is case sensitive

Validates that the checkbox its text has a partial match with the given text

```
Then bootstrap data-bootstrapcomponents-checkbox component with name {elementName}
I want to validate that the checkbox label partially equals the text {text}
```

Examples:

```
Then bootstrap data-bootstrapcomponents-checkbox component with name
formName.elementName I want to validate that the checkbox label partially equals
the text Partial Text
Then bootstrap data-bootstrapcomponents-checkbox component with name
formName.elementName I want to validate that the checkbox label partially equals
the text artial Tex
```

Note: this test is case sensitive

End bootstrap checkbox component

Bootstrap badge component Clicks on the bootstrap data-bootstrapextracomponents-badge component with the given name

```
When bootstrap data-bootstrapextracomponents-badge component with name
{elementName} is clicked
```

Example:

```
When bootstrap data-bootstrapextracomponents-badge component with name
orders.badgeName is clicked
```

End bootstrap badge component

Bootstrap label component Clicks the label

```
When bootstrap data-bootstrapcomponents-label component with name {elementName} is
clicked
```

Example:

```
When bootstrap data-bootstrapcomponents-label component with name
formName.elementName is clicked
```

Validates that the label has no text

```
Then bootstrap data-bootstrapcomponents-label component with name {elementName} I
want to validate that the label has no text
```

Example:

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the label has no text
```

Validates that the label its button its text has an exact match with the given text. Once a bootstrap label has an onAction attached, this test can be used for it

```
Then bootstrap data-bootstrapcomponents-label component with name {elementName} I
want to validate that the button its text equals the exact text {text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the button its text equals the exact
text Exact Text
```

Note: this test is case sensitive

Validates that the label its button its text has a partial match with the given text. Once a bootstrap label has an onAction attached, this test can be used for it.

```
Then bootstrap data-bootstrapcomponents-label component with name {elementName} I
want to validate that the button its text partially equals the text {text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the button its text partially equals
the text artial Tex
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the button its text partially equals
the text Partial Text
```

Validates that the label its text has an exact match with the given text

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the label equals the exact text
{text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the label equals the exact text Exact
Text
```

Validates that the label its text has a partial match with the given text

```
Then bootstrap data-bootstrapcomponents-label component with name {elementName} I
want to validate that the label equals the partial text {text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the label equals the partial text
Partial Text
Then bootstrap data-bootstrapcomponents-label component with name
formName.elementName I want to validate that the label equals the partial text
artial Tex
```

End bootstrap label component

Bootstrap datalabel Clicks the datalabel

```
When bootstrap data-bootstrapcomponents-datalabel component with name
{elementName} is clicked
```

Example:

```
When bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName is clicked
```

Validates that the datalabel its text has an exact match with the given text

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
{elementName} I want to validate that the label equals the exact text {text}
```

Example:

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the exact text Exact
Text
```

Validates that the datalabel its text has a partial match with the given text

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
{elementName} I want to validate that the label equals the partial text {text}
```

Examples:

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the partial text
Partial Text
Then bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the partial text
artial Tex
```

Validates that the datalabel has no text

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
{elementName} I want to validate that the label has no text
```

Example:

```
Then bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label has no text
```

End bootstrap datalabel

Bootstrap input group Validates that the field in fieldnumber x equals the given value

```
Then bootstrap data-bootstrapextracomponents-input-group component with name
{elementName} I want to validate that the text in field number {fieldNumber}
equals the text {text}
```

Example:

```
Then bootstrap data-bootstrapextracomponents-input-group component with name
formName.elementName I want to validate that the text in field number 1 equals the
text Exact Match
Then bootstrap data-bootstrapextracomponents-input-group component with name
formName.elementName I want to validate that the text in field number 12 equals
the text Exact Match
```

Inserts the text in textfield number x

```
When bootstrap data-bootstrapextracomponents-input-group component with name
{elementName} I want to insert the text {text} in field number {fieldNumber}
```

Examples:

```
When bootstrap data-bootstrapextracomponents-input-group component with name
formName.elementName I want to insert the text new text in field number 1
When bootstrap data-bootstrapextracomponents-input-group component with name
formName.elementName I want to insert the text 123 in field number 12
```

End bootstrap input group

End Bootstrap components

Toast component

The toast tests are NOT case sensitive Validates that there is a toast with the given type present {toastType} should be replaced by one of the following in the tests:

- info
- success
- warning
- error

```
Then default toast component I want to validate that there is a(n) {toastType}
toast present
```

Examples:

```
Then default toast component I want to validate that there is an info  toast
present
Then default toast component I want to validate that there is a warning toast
present
Then default toast component I want to validate that there is an info toast
present
Then default toast component I want to validate that there is an error toast
present
```

Validates that the toast component with the given type its text equals the given text

```
Then default toast component I want to validate that the text of the {toastType}
toast equals {toastMessage}
```

Example:

```
Then default toast component I want to validate that the text of the info toast
equals This is an info message
```

End toast component

Modal-dialog component

Presses the modal-dialog button with the given text. This can be used in any dialog (info/warning/error, etc.) **Note:** this test is case sensitive

```
When default modal-dialog component the button with the text {text} is pressed
```

Examples:

```
When default modal-dialog component the button with the text Yes is pressed
When default modal-dialog component the button with the text Cancel is pressed
```

Inserts the given text into the input field. Only works in a 'showInfoDialog'.

```
When default input-dialog I want to insert the text {text}
```

Example:

```
When default input-dialog I want to insert the text Example input dialog text
```

Clicks the input field and selects the option field with the given text. Only works on a 'showSelectDialog'

```
When default select-dialog I want to select the combobox item with the text {text}
```

Example:

```
When default select-dialog I want to select the combobox item with the text {text}
```

Certain dialogs cover the rest of the application with a gray overlay. If the test tries to interact with any component while this is active, an error will be given. To prevent such errors from happening, this test will wait until the overlay is gone. Example:

```
When default modal-dialog I want to wait untill the modal-dialog view is gone
```

Validation that the modal-dialog has an element with the given text. **Note:** no exact text is required. Meaning the text 'delete' will match the text 'delete this order'

```
Then default modal-dialog component I want to validate that the text {dialogText}
is present
```

Example:

```
Then default modal-dialog component I want to validate that the text Do you want
to delete this order? is present
```

If the text of the dialog popup is dynamic, this is a good replacement. Browser expects that a dialog appears. Example:

```
Then I expect a monal-dialog popup to appear
```

End modal-dialog component

Window component

This involves the 'application.createWindow(...) component Waits until the window component has disappeared. Example:

```
When servoy window component I want to wait untill the window disappears
```

End window component

Tabpanel component

Clicks the tab with the given text in the tabpanel component **Note:** this text is case sensitive. Partial match works

```
When servoy data-servoydefault-tabpanel component with name {elementName} the tab
with the text {text} is clicked
```

Example:

```
When servoy data-servoydefault-tabpanel component with name tabpanels.tabpanelName
the tab with the text Orders is clicked
```

Clicks the tab with the exact text in the tabpanel component **Note:** this text is case sensitive. This test expects an exact match with the tab its text

```
When servoy data-servoydefault-tabpanel component with name {elementName} the tab
with the exact text {text} is clicked
```

Example:

```
When servoy data-servoydefault-tabpanel component with name tabpanels.tabpanelName
the tab with the text Orders is clicked
```

End tabpanel component

Form Components

The following test steps only work for components that are inside form components. These steps require (most of the times) the data-svy-name of the 'formComponent' component and the data-svy-name of the component that requires interaction. Why are these steps required? Formcomponents can be copy pasted. This means that the component inside the form component can be on the same form twice. The data-svy-name of that component will no longer be unique. To counter this, the name of the form component AND the name of the component is required.

Bootstrap labels Clicks on the bootstrap label with the given name.

```
When formcomponent with the name {formComponentName} with a data-
bootstrapcomponents-label component with name {elementName} is clicked
```

Example:

```
When formcomponent with the name formName.formcomponentElementName with a data-
bootstrapcomponents-label component with name formName.elementName is clicked
```

Validates that the text of the label equals the given text

```
Then formcomponent with the name {formComponentName} with a data-
bootstrapcomponents-label component with name {elementName} I want to validate
that the label equals the exact text {text}
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a data-
bootstrapcomponents-label component with name formName.elementName I want to
validate that the label equals the exact text exact label text
```

Validates that the text of the label partially equals the given text

```
Then formcomponent with the name {formComponentName} with a data-
bootstrapcomponents-label component with name {elementName} I want to validate
that the label equals the partial text {text}
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a data-
bootstrapcomponents-label component with name formName.elementName I want to
validate that the label equals the partial text this is a partial text matc
Then formcomponent with the name formName.formcomponentElementName with a data-
bootstrapcomponents-label component with name formName.elementName I want to
validate that the label equals the partial text his is a partial text match
```

Validates that the label has no text

```
Then formcomponent with the name {elementName} with a bootstrap data-
bootstrapcomponents-label component with name {elementName} I want to validate
that the label has no text
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-label component with name formName.elementName
I want to validate that the label has no text
```

End bootstrap labels

Bootstrap datalabels Clicks the datalabel

```
When formcomponent with the name {formComponentName} with a bootstrap data-
bootstrapcomponents-datalabel component with name {elementName} is clicked
```

Example:

```
When formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName is clicked
```

Validates that the datalabel its text has an exact match with the given text

```
Then formcomponent with the name {formComponentName} with a bootstrap data-
bootstrapcomponents-datalabel component with name {elementName} I want to validate
that the label equals the exact text {text}
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the exact text Exact
Text
```

Validates that the datalabel its text has aa partial match with the given text

```
Then formcomponent with the name {formComponentName} with a bootstrap data-
bootstrapcomponents-datalabel component with name {elementName} I want to validate
that the label equals the partial text {text}
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the partial text
Partial Text
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label equals the partial text
artial Tex
```

Validates that the datalabel has no text

```
Then formcomponent with the name {formComponentName} with a bootstrap data-
bootstrapcomponents-datalabel component with name {elementName} I want to validate
that the label has no text
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-datalabel component with name
formName.elementName I want to validate that the label has no text
```

End bootstrap datalabels

Bootstrap select component Checks the combobox component and checks if a row with the given text does not exist.

```
When formcomponent with the name {formComponentName} with a data-bootstrapcomponents-select component with name {elementName} I want to validate that a row with the text {text} does not exist
```

Example:

```
When formcomponent with the name formName.formcomponentElementName with a data-bootstrapcomponents-select component with name formName.elementName I want to validate that a row with the text rowText does not exist
```

Selects a row inside the combobox with the exact given text

```
When formcomponent with the name {formComponentName} with a data-bootstrapcomponents-select component with name {elementName} I want to select the combobox item with the exact text {text}
```

Example:

```
When formcomponent with the name formName.formcomponentName with a data-bootstrapcomponents-select component with name formName.elementName I want to select the combobox item with the exact text Red
```

Clicks the given select component

```
When formcomponent with the name {formComponentName} a bootstrap data-bootstrapcomponents-select component with name {elementName} is clicked
```

Example:

```
When formcomponent with the name formName.componentName a bootstrap data-bootstrapcomponents-select component with name formName.elementName is clicked
```

End bootstrap select component

Bootstrap textfield component Inserts the text into the given component

```
When formcomponent with the name {formComponentName} a bootstrap data-
bootstrapcomponents-textbox component with name {elementName} the text {text} is
inserted
```

Example:

```
When formcomponent with the name formName.formcomponentElementName a bootstrap
data-bootstrapcomponents-textbox component with name formName.elementName the text
my text is inserted
```

Validates that the bootstrap textfield is empty

```
Then formcomponent with the name {formComponentName} with a data-
bootstrapcomponents-textbox component with name {elementName} I want to validate
that text text is blank
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a data-
bootstrapcomponents-textbox component with name formName.elementName I want to
validate that text text is blank
```

End bootstrap textfield component

Bootstrap button component Clicks the bootstrap button

```
When formcomponent with the name {formComponentName} a bootstrap data-
bootstrapcomponents-button component with name {elementName} is clicked
```

Example:

```
When formcomponent with the name formName.formcomponentElementName a bootstrap
data-bootstrapcomponents-button component with name formName.elementName is
clicked
```

Validates that the given button is disabled or enabled

```
Then formcomponent with the name {formComponentName} with a bootstrap data-
bootstrapcomponents-button component with name {elementName} I want to validate
that the button is {enabled|disabled}
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-button component with name formName.elementName
I want to validate that the button is enabled
Then formcomponent with the name formName.formcomponentElementName with a
bootstrap data-bootstrapcomponents-button component with name formName.elementName
I want to validate that the button is disabled
```

End bootstrap button component

Presense of an element Expects an element to be present

```
Then formcomponent with the name {formComponentName} I expect an element with the
name {elementName} to be present
```

Example:

```
Then formcomponent with the name formName.formcomponentElementName I expect an
element with the name formName.elementName to be present
```

End presense of an element

End Form Components

HTML view component

Validates that the given text partialy (or exact) matches the value of the HTML view component

```
Then servoy htmlview component with name {elementName} I want to validate that the
htmlview contains the text {text}
```

Example:

```
Then servoy htmlview component with name orders.htmlView I want to validate that
the htmlview contains the text 5 licenses
```

End HTML view component

Tables

Basic Servoy Table

Protractor works similar to humans. It can only interact with elements that are loaded in the viewport. This function scrolls the table until it has found a record with the give text. This will always be the first record that matches the String. **Note:** this test is case sensitive

```
When servoy table component with name {elementName} I scroll to the record with {text} as text
```

Example:

```
When servoy table component with name customers.customerTable I scroll to the record with Protractor test as text
```

This is a very specific test. This test can find an element that has a specific attribute. For example an attribute with 'disabled: disabled' or 'title:elementTitle'.

```
Then servoy table component with name {elementName} I want to validate that an element contains a(n) {attribute} with the value {value} on the row with the text {text} exists
```

Examples:

```
Then servoy table component with name formName.elementName I want to validate that an element contains a title with the value titleValue on the row with the text rowtext exists
Then servoy table component with name formName.elementName I want to validate that an element contains a(n) disabled with the value disabled on the row with the text rowtext exists
```

This step does the same as the previous scroll step, minus the scrolling. **Note:** this test is case sensitive

```
When servoy table component with name {elementName} I want to validate that a record with the text {text} exists
```

Example:

```
When servoy table component with name customers.customerTable I want to validate
that a record with the text Protractor test exists
```

This step requires 2x a data-svy-name. This is because the basic Servoy table uses components. This test clicks on the component with the given name inside the table Components inside the table component have a longer name than other components.

```
When servoy table component with name {tableName} I want to select element number
{number} with name {elementName}
```

Example:

```
When servoy table component with name customers.customerTable I want to select
element number 5 with name customers.svy_lvp_customers.customerName
```

Sometimes a table is too wide to be fully viewed in the viewport. This test scrolls horizontally until the header can be seen.

```
When servoy table component with name {elementName} I want to scroll the table to
the right to an element with the name {headerElementName}
```

Example:

```
When servoy table component with name customers.customerTable I want to scroll the
table to the right to an element with the name customers.dateCreation
```

Validates that the table has x amount of rows

```
Then servoy table component with name {elementName} I want to validate that there
is/are {rowCount} row(s) currently visible
```

Examples:

```
Then servoy table component with name formName.elementName I want to validate that
there are 5 rows currently visible
Then servoy table component with name formName.elementName I want to validate that
there is 1 row currently visible
```

```
Then servoy table component with name formName.elementName I want to validate that
there are 0 rows currently visible
```

End Basic Servoy Table

Servoy extra Table

Scrolls to the first element with the given text inside the table **Note:** this test is case sensitive

```
When servoy extra table component with name {elementName} I scroll to the record
with {string} as text
```

Examples:

```
When servoy extra table component with name companies.companyTable I scroll to the
record with Pete as text
When servoy extra table component with name companies.companyTable I scroll to the
record with this is a sentence as text
```

Selects the given row inside the table that is currently visible. Trying to target a row outside of the viewport will trigger an out of bounce error

```
When servoy extra table component with name {elementName} I want to select row
number {rowNumber}
```

Example:

```
When servoy extra table component with name companies.companyTable I want to
select row number 5
```

Validates that a record with the given that is currently in the viewport of the extra table exists

```
When servoy extra table component with name {elementName} I want to validate that
a record with the text {text} exists
```

Example:

```
When servoy extra table component with name companies.companyTable I want to
validate that a record with the text testText exists
```

Counts the rows of the extra table that are currently visible in the viewport

```
When servoy extra table component with name {elementName} I want to validate that
there are {rowCount} row(s)
```

Examples:

```
When servoy extra table component with name companies.companyTable I want to
validate that there are 2 rows
When servoy extra table component with name companies.companyTable I want to
validate that there is 2 row
```

Scrolls to a record with the given text within the table **Note:** this test is case sensitive

```
When servoy extra table component with name {elementName} I want to scroll and
select the row with text {rowText}
```

Example:

```
When servoy extra table component with name companies.companyTable I want to
scroll and select the row with text Servoy BV
```

Since it's possible to add style class dataproviders to table columns, this test has been made to be able to look for a column with the given text and select the column with the given class on the same row. This is especially useful in tables which have a button that goes to the detail page. **Note:** only 1 class can be passed in this test

```
When servoy extra table component with name {elementName} I want to click on the
icon with the class {className} on the row with the text {text}
```

Example

```
When servoy extra table component with name companies.companyTable I want to click
on the icon with the class fa-arrow-right on the row with the text Servoy BV
```

Clicks on one of the headers of the extra table with the given text **Note:** this test is case sensitive

```
When servoy extra table component with name {elementName} I want to sort the table
by {tableHeader}
```

Example:

```
When servoy extra table component with name companies.companyTable I want to sort
the table by ID
```

End Servoy extra Table

Aggrid table component

Important note: scrolling through an aggrid table does not work while the table is grouped

Scrolls the table to a record with the given text

```
When servoy data-aggrid-groupingtable component with name {elementName} I scroll
to the record with {string} as text
```

Examples:

```
When servoy data-aggrid-groupingtable component with name companies.companyTable I
scroll to the record with Servoy as text
When servoy data-aggrid-groupingtable component with name companies.companyTable I
scroll to the record with Servoy BV as text
```

Since the grouping table can be grouping, this test searches for a row in the table that matches the given text. After that it will either expand or collapse the row. Row levels are required to expand/collapse the correct row level (1 indexed) **Note:** the table first has to be grouped before this test works. rowOption has to be replaced by either 'collapse' or 'expand'

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
{rowOption} row level {rowLevel} with {rowText} as text
```

Examples:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to expand row level 1 with ISV as text
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to collapse row level 3 with ISV as text
```

Sorts the grouping grid by a header with the given text **Note:** this test is NOT case sensitive

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
sort the table by {sortBy}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to sort the table by Company Name
```

Groups the grouping table by the given text **Note:** this test is case sensitive

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
group the table by {tableHeaderText}
```

Example:

```
When servoy data-aggrid-groupingtable component with name customers.groupingTable
I want to group the table by manager
```

Ungroups the grouping table by the given text **Note:** this test is case sensitive

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
ungroup the table by {tableHeaderText}
```

Example:

```
When servoy data-aggrid-groupingtable component with name customers.groupingTable
I want to ungroup the table by manager
```

Changes the order in which way the grouping table is grouped. This drags the given grouped header with the given and moves it as the main grouping header **Note:** on the currently released version (12th of july 2018), this step does not work yet

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
drag the grouping item with {groupingText} as text to the start
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to drag the grouping item with State as text to the start
```

Scrolls the grouping table back to the top

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
scroll to the top
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to scroll to the top
```

Selects the nth row of the table currently visible in the viewport

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
select row number {rowNumber}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to select row number 5
```

Selects the row with the given text in the table currently visible in the viewport **Note:** this test is case sensitive

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
select the record with the text {text}
```

Example:

```
When servoy data-aggrid-groupingtable component with name fruits.groupingTable I
want to select the record with the text apple
```

Validates that a record with the given text exists within the viewport, within the table. **Note:** this test is case sensitive

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
validate that a record with the text {text} exists
```

Example:

```
When servoy data-aggrid-groupingtable component with name contacts.groupingTable I
want to validate that a record with the text Pete exists
```

Finds a record in the table that matches the given text and clicks it

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
scroll and select the row with the text {rowText}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to scroll and select the row with the text Servoy BV
```

Finds a record in the grouping table that matches the given text

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
scroll to the row with text {rowText}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to scroll to the row with text Servoy BV
```

Finds a record in the grouping table that matches the given text and clicks on the class with the given class name which is located on the same row

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
scroll and select the row with text {rowText} and click the element which contains
the class {className}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to scroll and select the row with text Servoy BV and click the element
which contains the class fa-times
```

Finds a record in the grouping table that matches the given text and clicks on the class with the given class name which is located on the same row. This will not scroll the grid!

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
click on the element which contains the class {className} on the row with the text
{text}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to click on the element which contains the class fa-times on the row with
the text Servoy
```

Clicks on the given class of the given row in the grouping table

```
When servoy data-aggrid-groupingtable component with name {elementName} I want to
click on the element which contains the class {className} in row number
{rowNumber}
```

Example:

```
When servoy data-aggrid-groupingtable component with name companies.groupingTable
I want to click on the element which contains the class fa-arrow-right in row
number 12
```

Validates that the grouping table consists of x rows. Do note that it only counts the rows currently visible in the viewport.

```
Then servoy data-aggrid-groupingtable component with name {elementName} I want to
validate that there are/is {rowNumber} row(s)
```

Examples:

```
Then servoy data-aggrid-groupingtable component with name {elementName} I want to
validate that there are 3 rows
Then servoy data-aggrid-groupingtable component with name {elementName} I want to
validate that there is 1 row
```

End Aggrid table component

End tables