**CYPRESS-AGILITY MULTICHANNEL**

**Cypress:**  
 Cypress is a next generation front end testing tool built for the modern web. We address the key pain point’s developers and QA engineers face when testing modern applications.

**Installation process:**

Step1: Before Cypress Install NodeJs

Visit [**https://nodejs.org/en/**](https://nodejs.org/en/)

Install NodeJS

Step2: Set Nodejs Path as Environment variable

Step3: Open command Prompt and run npm -v to check version and installed node js

Properly or not

**Installation process 2:**

* Create any Directory(Project Directory)
* Open Command Prompt with Project Directory Path
  + C:/ProjectDir>

Step1: Run command **npm init** for creating package.json fill the details as and save

{

"name": "agilitycypress",

"version": "1.0.0",

"description": "AgilityMultichannelTesting Cypress",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"cypress": "cypress open"

},

"author": "ravi",

"license": "ISC",

"dependencies": {

"cypress": "^3.1.0"

}

}

Step3: Visit <https://www.cypress.io/>

Run Command **npm install cypress**

Step4: After installation Cypress Successfully Run command

**/node\_modules/.bin/cypress**

Cypress installed successfully, and it will show Folder Structure and example Files

**After Installation (Recommended):**

Step1: Install Visual Studio Code

After Install Visual Studio Code Open Project in Visual Studio code and Run Cypress in Terminal

Step2: Create tsconfig file to load cypress command default in entire project.

{

"compilerOptions": {

"allowJs": true,

"baseUrl" : "../node\_modules",

"types" :[

"cypress"

]

},

"include" : [

"\*\*/\*.\*"

]

}

Save this file in Cypress folder

**Cypress.json**

Once Installation Process finished, Add Configuration as per requirement for cypress.json file

{

"baseUrl": “http://10.54.3.39/AMI”,

"video": false,

"defaultCommandTimeout": 20000,

"pageLoadTimeout": 900000,

"chromeWebSecurity": true,

"integrationFolder": "cypress/integration",

"pluginsFile": "cypress/plugins",

"fixturesFolder": "cypress/fixtures",

"supportFile": "cypress/support",

"screenshotsFolder": "cypress/screenshots",

"viewportHeight": 960,

"viewportWidth": 1400,

"waitForAnimations": true,

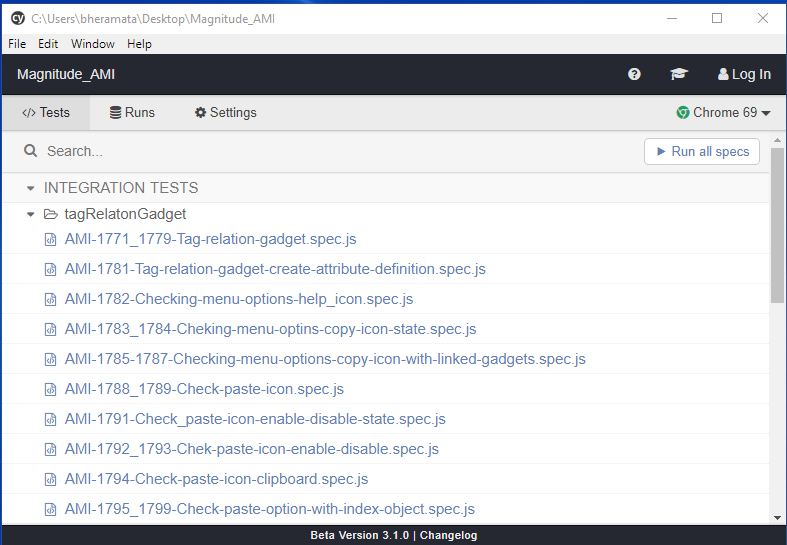
}

**Open Cypress Browser:**

After Configuration Open Cypress Browser in Visual Studio Terminal

**Command: npm run cypress**

The following window will open.



Cypress Browser Opened Successfully, with Example Files (Delete Example Files Folder Recommended).

**Cypress Folder Structure:**

Open Cypress Folder

Folder 1: fixtures --- Supporting all Data Files (.Json format)

Folder 2: integrations --- Supporting Test Scripts (.spec.js or .js Extensions)

Folder 3: plugins --- Supporting Plugins

Index.js --- (Used to upload plugins)

Folder 4: support --- Supporting User Defined Commands

Commands.js --- (Create Custom and overwrite existing Commands)

Index.js--- (Global Configuration support, load automatically before Test Files)

**Cypress Agility Multi Channel:**

1. Created Two Json Files in fixture Folder Named as
2. amiDataValue.json
3. amiDomElements.json
4. All Test Scripts Files Created In Integration subfolders with .spec.js Extensions
   1. integration/tagRelationGadget
      1. Created all AMI Test Scripts Related Tag Relation Gadget
      2. All files named as Test case id
   2. integration/workingListGadget
      1. Created all AMI Test Scripts Related Working List Gadget
      2. All files named as Test case id
   3. plugins/index.js
      1. Created logReport plugin
   4. support/commands.js
      1. Created 4 Custom commands
   5. utils/ravi\_utils
      1. Reusable util files/Functions
   6. tsconfig.json
      1. Type Scripting for cypress
   7. cypress.json
      1. Configuration setup
   8. package-lock.json
   9. package.json

**Log Reports**

**Manually logging:**

We have created 4 commands in supports/commands.js file called start and finish. These 4 functions can be called like this

1. cy. logHeader(‘test case description’)

* logHeader method will checks whether logReport.txt is present or not, if not logReport.txt will be created.
* logHeader will write the heading of the log.
* Ex: cy.logHeader("Work list gadget with list of objects");
* o/p: Test Scenario: Work list gadget with list of objects

1. cy.start(‘test case description’):

* start method used to write the start of the test case.
* Ex: cy.start('AMI-2021:28').
* o/p: 10/18/2018, 7:29:47 PM: AMI-2027:34 Test Execution is Started.

1. cy.finish(‘test case description’);

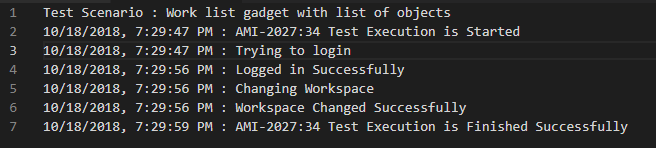
* finish method used to write the finish of the test case.
* Ex: cy.finish('AMI-2021:28').
* o/p: 10/18/2018, 7:29:59 PM: AMI-2027:34 Test Execution is Finished Successfully

1. cy.addLog (‘test suit description’);

* addLog method used to write the intermediate steps completed inside the test case.
* Ex: cy.addLog("Trying to login");
* o/p: 10/18/2018, 7:29:47 PM: Trying to login

These functions will log into the logReport.txt file in the project.

Example of sample log Report:



**Automatic logging of failed testcases:**

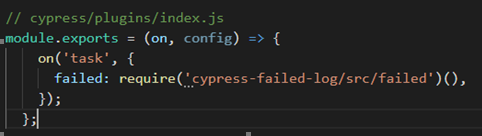
1. This is achieved by “cypress-failed-log” third party package.
2. It can be installed with following command

npm install --save-dev cypress cypress-failed-log

1. Then include this code block in your cypress/support/index.js file



1. Then include this code block in your cypress/plugins/index.js



1. This will automatically start logging the failed test scripts and also thaks its snap shots.
   1. Failed Logs will be logged in cypress/Logs folder as a json file.
   2. Snap shots will be present in the same folder cypress/Logs.
2. We have a problem here one of the xhr requests is failing post/agility/… So our automatic logging is failing. So not implemented but we have Refernce

<https://github.com/bahmutov/cypress-failed-log>