# WorldSpace Attest Quick Start Guide

## Contents

# What is WorldSpace Attest?

WorldSpace Attest is a bundle of functionality that allows for easy automated accessibility testing for Web-based applications.

## What Comprises WorldSpace Attest?

WorldSpace Attest consists of:

1. The FireEyes II extension with custom rule upload functionality
2. A JavaScript API for loading custom rules and performing accessibility tests on Web pages
3. A JavaScript API for logging test results
4. An application to generate reports from the test result logs in HTML, Excel and JUnit XML formats
5. An online repository for downloading the components
6. An online repository for managing and distributing custom rule sets
7. Documentation and help
8. Support

# What do I need to get started?

## Prerequisites

Before you get started, you must have the following pieces of information.

1. The company-id for your license of WorldSpace Attest
2. The username and password for your login to the centralized repository system

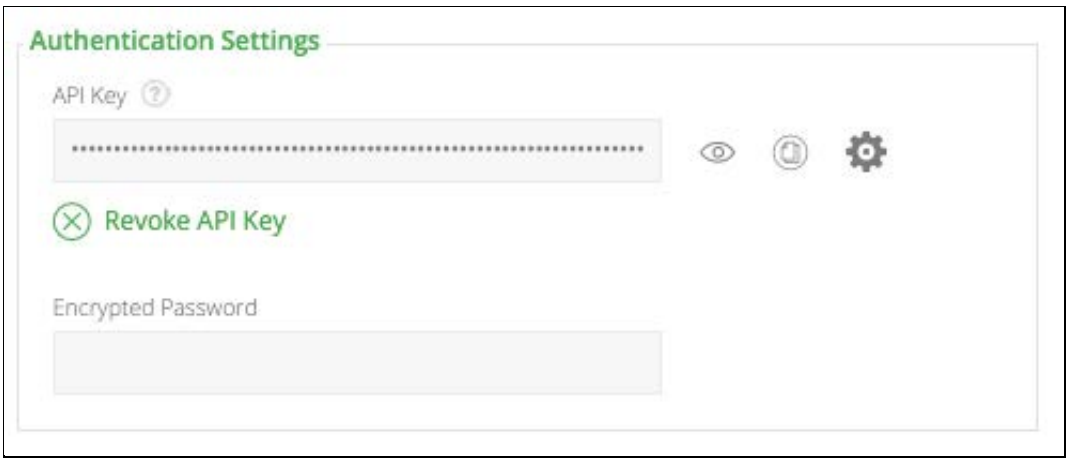
  

## Generating your personal API key

In order to access the WorldSpace Attest components, you need to generate a personal API key for your account. This key is private to yourself and only needs to be generated once and can be re-used. It can be revoked and regenerated at any time as required.

**To generate an API key**:

1. Log in to the Deque repository at https://agora.dequecloud.com using your Username and Password credentials.
2. Go to your profile page, by clicking the 'Welcome' message on the home page, or by going to the following URL https://agora.dequecloud.com/artifactory/webapp/#/profile
3. In the Authentication settings, click the 'cog' icon to generate the key.
4. Click the 'copy key to clipboard' button. Paste the key into a local text file on your PC for later use.




You can at any time go back to your profile page to get or regenerate the API key.

## Downloading the Components

### *Node.js and WebdriverJS*

WorldSpace Attest uses the npm system for managing the JavaScript libraries. There is 1 registry that you need access to:

1. The **attest** registry

This registry allows **npm** to be used to download modules using npm's **--registry** option

## Attest URL

https://agora.dequecloud.com/artifactory/api/npm/attest

This registry contains multiple packages:

1. `attest-node'
2. `attest'
3. `attest-webdriverjs'
4. `axe-core'

The `attest-node' package is for use with Node.js environments such as the Node.js version of Selenium (selenium-webdriver) and is generally used in conjunction with the `attest-webdriverjs' package.

The `attest' package is for use within a browser environment such as for use in unit tests like QUnit, Jasmine and Mocha.

The `axe-core' package is the raw rules engine which is embedded into the other packages. You should generally not have to use this library directly.

## Custom Rules

In addition, if you want to use your company's custom rules setup(s), there is a registry with the following URL:

https://agora.dequecloud.com/artifactory/api/npm/**company-id**

Where **company-id** is an identifier that uniquely identifies your company within the Attest repository structure.

## Setting up access to each registry

To register each registry with your local npm and allow npm to download from that registry, you must configure your **.npmrc** file with authentication information. This is done by sending a **curl** command to the registry from the command line and putting the response into your .npmrc file.

The format of the curl command is

```
curl -uusername:api-key "REPO-URL/auth/REPO-SHORTNAME"
```

Where the bolded components are replaced as follows:

- **username** - is replaced by your Attest server user name
- **api-key** - is replaced by your Attest server username (API key)
- **REPO-URL** is replaced by one of the two registry URLs listed above
- **REPO-SHORTNAME** is replaced by any string that is a unique identifier for that registry (you can choose this)

So the command for the **attest** registry for a user "dylan" with password "pwd" and REPO-SHORTNAME would be:

```
curl -udylan:api-key
"https://agora.dequecloud.com/artifactory/api/npm/attest/auth/dequeattest"
```

(Note: for more information on curl for Windows, see cURL for Windows.)

The response from the server should be a series of lines that looks like:

```
@dequeattest:registry=https://agora.dequecloud.com/artifactory/api/npm/attest/
//agora.dequecloud.com/artifactory/api/npm/attest/:_password=GOBBELDYGOOK
//agora.dequecloud.com/artifactory/api/npm/attest/:username=dylan
//agora.dequecloud.com/artifactory/api/npm/attest/:email=dylan.barrell@deque.com
//agora.dequecloud.com/artifactory/api/npm/attest/:always-auth=true
```

Copy these lines verbatim and append them to your ~/.npmrc file.

Repeat this process for each registry with a new, unique REPO-SHORTNAME for each registry.

### *Downloading from a registry*

To download from a registry, you use the **npm install** command using the **--registry** option

*Attest-node module*

```
npm install --save-dev attest-node --registry
https://agora.dequecloud.com/artifactory/api/npm/attest/
```

*Attest Browser module*

```
npm install --save-dev attest --registry
https://agora.dequecloud.com/artifactory/api/npm/attest/
```

*Custom rules*

```
npm install --save-dev attest-rules --registry
https://agora.dequecloud.com/artifactory/api/npm/company-id/
```

## Example: with webdriverJS

Create a directory for the application and initialize your npm application:

```
mkdir use-attest
cd use-attest
npm init
```

Install the WebdriverJS and other test harness components (note if you are integrating Attest into an existing project, you should have some of these components, or their equivalents, already installed)

```
npm install --save-dev selenium-webdriver chai mocha
npm install -g mocha-cli
```

Install the attest-node and attest-webdriverjs modules:

```
npm install --save-dev attest-node attest-webdriverjs --registry
https://agora.dequecloud.com/artifactory/api/npm/attest/
```

You now have **three** alternatives in terms of using the custom rules.

### Global Custom Rules Config

To store and use the rules from a global location, install the custom rules module globally:

```
npm install -g attest-rules --registry
https://agora.dequecloud.com/artifactory/api/npm/company-id/
```

When you do this, npm will print out the location of the global module like this:

```
attest-rules@1.0.0
/Users/dylanbarrell/.nvm/v4.2.2/lib/node_modules/attest-rules
```

Set up the ATTEST_PATH environment variable to use the attest.json file from this module.

```
export
ATTEST_PATH=/Users/dylanbarrell/.nvm/v4.2.2/lib/node_modules/attest-
rules/attest.json
```

## Project-Specific Custom Rules Config

To store and use the rules within the project

```
npm install --save-dev attest-rules --registry
https://agora.dequecloud.com/artifactory/api/npm/company-id/
```

Now copy the attest.json file to the root of your project

```
cp node_modules/attest-rules/attest.json
```

## No Custom Rules

If you are not using custom rules, create an attest.json file in the project root with the contents of

```
{}
```

(this is the equivalent of an empty JSON object)

## Creating the WebdriverJS test

Install the WebdriverJS and other test harness components (note if you are integrating Attest into an existing project, you should have some of these components, or their equivalents, already installed)

```
npm install --save-dev selenium-webdriver chai mocha
npm install -g mocha-cli
```

Create a test file called test.js and copy the following code into it:

```
var selenium = require('selenium-webdriver');
var AttestBuilder = require('attest-webdriverjs');

var assert = require('chai').assert;

describe('Attest Quick Start', function() {
   var driver, browser;

   // Load the browser
   before(function() {
       driver = new selenium.Builder().forBrowser('firefox');
       browser = driver.build();
       browser.manage().timeouts().setScriptTimeout(10000);
   });

   // Close the browser at the end
   after(function(done) {
       browser.quit().then(done);
   });
```

```
   describe('dequeuniversity.com', function (done) {

       it('has 0 accessibility violations on root', function
(done) {
           browser.get('http://www.dequeuniversity.com/')
           .then(includeAttest)
           .then(function (attest) {
               attest.analyze(function(results) {
                   assert(results.violations.length === 0);
                   done();
               });
           });
       });
   });

   /**
    * Returns a promise indicating when attest is ready to test
the page
    */
   function includeAttest() {
       return browser.executeAsyncScript(function(callback) {
           var script = document.createElement('script');
           script.innerHTML =
'document.documentElement.classList.add("deque-attest-is-
ready");';
           document.documentElement.appendChild(script);
           callback();

       }).then(function () {
           return
browser.wait(selenium.until.elementsLocated(selenium.By.css('.de
que-attest-is-ready')));

       }).then(function () {
           return new AttestBuilder(browser, 'wcag2');
       });
   }

});
```

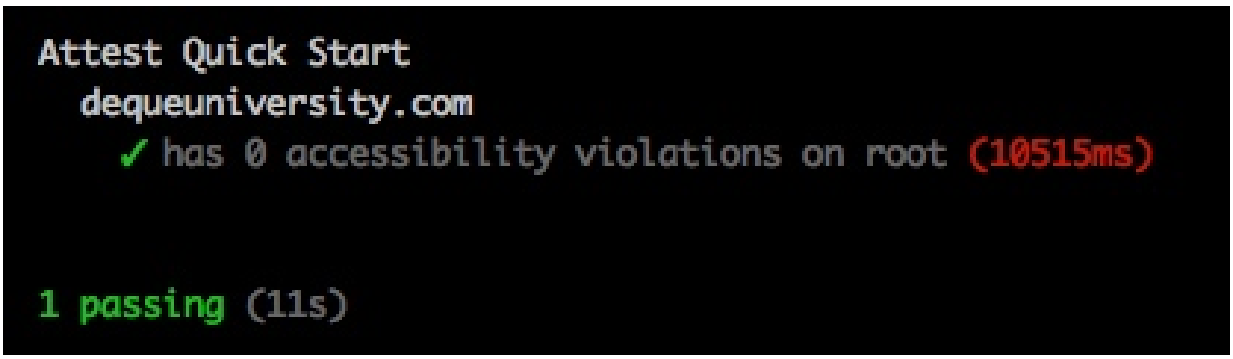
Now rule the mocha command from the command line:

```
mocha test --timeout 20000
```

You should see a browser spawned that tests the home page of Deque University and then closes. After which the mocha command will print something along the lines of:

```
Attest Quick Start
  dequeuniversity.com
    ✓ has 0 accessibility violations on root (NNNNNms)

1 passing (Ns)
```



## Example in Browser-based Unit Tests like Mocha, Jasmine and QUnit

Install the attest module (this assumes you are using npm, if this is not possible, please contact Deque on how to get access to the attest.js file).

```
npm install --save-dev attest --registry
https://agora.dequecloud.com/artifactory/api/npm/attest/
```

Include the attest.js file into your test fixture. The attest.js file is in the base directory of the attest Node module (./node_modules/attest/attest.js). Including this can be done in many ways and is dependent on your test system. If you have a static test harness, you would insert a <script> tag like:

```
<script src="path_to/attest.js"></script>
```

Within your tests, make a single call to initialize the library. This can be done by calling attest.init() or by calling attest.configure(). Attest.init is used if you want to use one of the built-in rule sets such as the WCAG 2 A and AA set.

Attest.configure is used if you want to use custom rules.

```
Initializing using attest.init()
beforeEach(function (done) {
      attest.init('wcag2', function () {
            done();
      })
});
```

Call the checker wherever you want inside your tests. It is good practice to call it whenever a new piece of UI or UI state is exposed.

```
describe('some test', function () {
      beforeEach(function (done) {
            attest.init('wcag2', function () {
                  done();
            })
      });
      it('should do something that creates some UI', function (done) {
            var fixtureScope = document.querySelector('some selector');

            // do stuff with the fixture

            attest.a11yCheck(fixtureScope, function (results) {
                  assert(results.violations.length, 0);
                  done();
            });
      });
});
```

# Appendices

## cURL for Windows

cURL is a simple command line http client. cURL is available for Windows and is commonly found in Windows GIT command-line interfaces like GitBash.

You can download cURL http://curl.haxx.se/download.html. Learn how to use and configure cURL http://curl.haxx.se/docs/manpage.html.

Any http or REST client that allows basic authentication username and passwords can be used as an alternative to cURL.