

Setting up an Automation Test Report Server

Instructions for setting up an Automation Test Report Server.



The **Automation System** provides a way to run tests of various systems in **Unreal Engine**, but it has some limitations for providing visibility of data that can hamper its use for distributed testing. The Automation System can generate an HTML report instead of a log file, but over time these reports have become more complex, and if the HTML report is not self-contained locally, you cannot open it directly on a modern web browser.

As a solution to this limitation, you can use an **Automation Test Report Server** to output test results to a shared drive. You can then connect to the server to read the report.

Because there is no back-end to the Automation Test Report Server aside from serving files, it is a very basic, non-demanding web server that you can easily set up on most hardware. This means external third parties running tests remotely can set up a server on their own to share test results.

Prerequisites

To set up an Automation Test Report Server, you must have the following prerequisite libraries and programs:

- [Node.js](#)
- [Bower](#)
- [Git](#)

If you do not have these prerequisites installed, you will encounter errors.



You can perform this setup process on a local host, but because you cannot share a local host, you will need to set it up with a valid DNS and IP address for the server to be accessible remotely.

Setting up a web domain and address is beyond the scope of this documentation. Use whatever solution is most appropriate to meet your requirements.

Steps

To set up your Automation Test Report Server, follow the procedure below.

In the instructions below, replace the placeholder sections as follows:

- `(path to result)`: The path where the test results output will be stored. For example, `C:\http_server\local`.
- `(stream root)`: The path where the source control stream or the UE installation is located. For example, `C:\Epic\UE`.

Install the HTTP Server

1. Make sure you have installed all the needed [prerequisites](#).
2. Open a command prompt window.
3. Make sure `%USERPROFILE%\AppData\Roaming\npm` is in your command prompt path.
4. Run the following command:

```
1 npm install http-server bower -g
2
```

 Copy full snippet

This will install the Bower web server globally.

5. Run the following command:

```
1 xcopy (stream root)\Engine\Content\Automation (path to result) /E
2
```

 Copy full snippet

This will recursively copy over the images, config files, and templates for the HTTP server that will be referenced by the generated test reports.

6. Run the following command:

```
1 cd (path to result)& bower install
2
```

 Copy full snippet

This will tell Bower to install the javascript libraries that are referenced in the copied config files.

Run the HTTP Server

1. Run the following command:

```
1 cd (path to result)& http-server
2
```

 Copy full snippet


This changes to the working directory for the HTTP server, and starts it running.

2. Leave the command prompt open. Closing the command prompt will shut down the server.

Viewing Test Results

1. When you run your Automation tests, use the following command to specify the `(path to result)` directory for the output. This sets up your tests to output directly to the folder where you set up the web server and its resources.

```
-ReportExportPath=C:\http_server\local
```

 Copy full snippet

2. Open the IP address of your web server in a browser. You can use `http://localhost:8080/` internally, but you will need to set up a proper web address for distributed work, as noted previously.

If the output path is the same as the server path, you can use it directly. However, if you are using a subdirectory to store your files, you will need to add it to the path.

For example, if you output to something like: `C:\http_server\local\mypath`, then the url should be: `http://localhost:8080/mypath`.

3. The web server will provide a list of files. Any reports output to the correct directory will be visible immediately.
4. If you provide a [session name](#) and move `Index.html` and `index.json` (normally found in the target directory) along with the directory, you can access the results of a specific test session by name.
5. When you finish reviewing results, you can close the command prompt and shut down the server.

Setting Up Seperate Test Sessions

To better organize your test results and prevent overwriting, you can separate different test sessions into different directories.

1. Define a directory name of your choosing to make identifying particular tests easier. For this example, we will use **Session1**.
2. Output the result of your Automation tests to `(path to result)\Session1`.
3. When you run a different set of tests, change the name. You can either increment the session number, or use a completely different descriptive name, for every new set of tests. This will avoid overwriting previous results.
4. To access your results through the web browser, append the directory name for the appropriate test session to the path.

Result

If everything functions correctly, you will see something resembling the image below, showing a list of tests executed and their respective statuses.

Automation Results

Windows

Summary **1**

Warnings **0**

Errors **0**

Log

Warning

Error

Ran 1 of 1 tests in **00:00:00**



Algo

Topological Sort
Unique

Misc

Algos