Componentized-WEAP RESTful Framework Installation and User Guide*

Version 1.1**

Mostafa D. Fard

Hessam S. Sarjoughian***

Arizona Center for Integrative Modeling and Simulation
School of Computing, Informatics, and Decision System Engineering
Arizona State University, Tempe, Arizona, USA
https://acims.asu.edu

July 2023

* Funding: United States National Science Foundation Grant #CNS-1639227, "INFEWS/T2: Flexible Model Compositions and Visual Representations for Planning and Policy Decisions at the Sub-regional level of food-energy-water nexus".

** Version 1.1 has changes for the latest versions of software tools and improved readability. There are no changes to the C-WEAP API specification and code base. This version replaces version 1.0 which released in July 2020.

*** Point-of-Contact

1. Componentized-WEAP Software Application

The Componentized-WEAP (C-WEAP) is a RESTful framework application [1] written in **NodeJS** for the Water Evaluation and Planning (WEAP) system [2, 3]. The executable version of this software is publicly available. It can be used as if it is running from source code. A short demo video is also available.

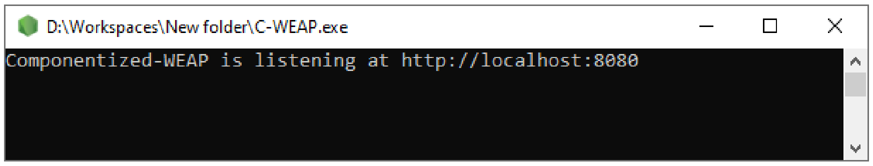
2. WEAP Software System

WEAP is a propriety system software that runs on the Windows 32 OS. Information on licensing this software is available at WEAP Licensing.

3. Executable Componentized-WEAP Software Application Installation

The executable version of the Componentized-WEAP (C-WEAP) framework is a standalone application to be run on the Windows OS. Download the executable version of the C-WEAP framework must from https://acims.asu.edu/software/c-weap/ and unzip the downloaded file in a directory. It contains the "workspace" folder (which manages the required flat files needed by the C-WEAP framework and WEAP system), the "config.json" file (to set the host and port number of the web-service), and the C-WEAP.exe file. All required libraries and frameworks (NodeJS, Express, etc.) are embedded in this executable file. The unzipped downloaded file must not be installed on any shared disk drive such as Dropbox.

To run the C-WEAP, double-click on the C-WEAP.exe file. Upon the successful execution of the C-WEAP application, the message "`Componentized-WEAP is listening at http://{hostname}:{port}`" appears in the first line of a Command Prompt console such as the one shown in Figure 1.



```
Componentized-WEAP is listening at http://localhost:8080
```

Figure 1. Windows console displaying the successful execution of the C-WEAP.

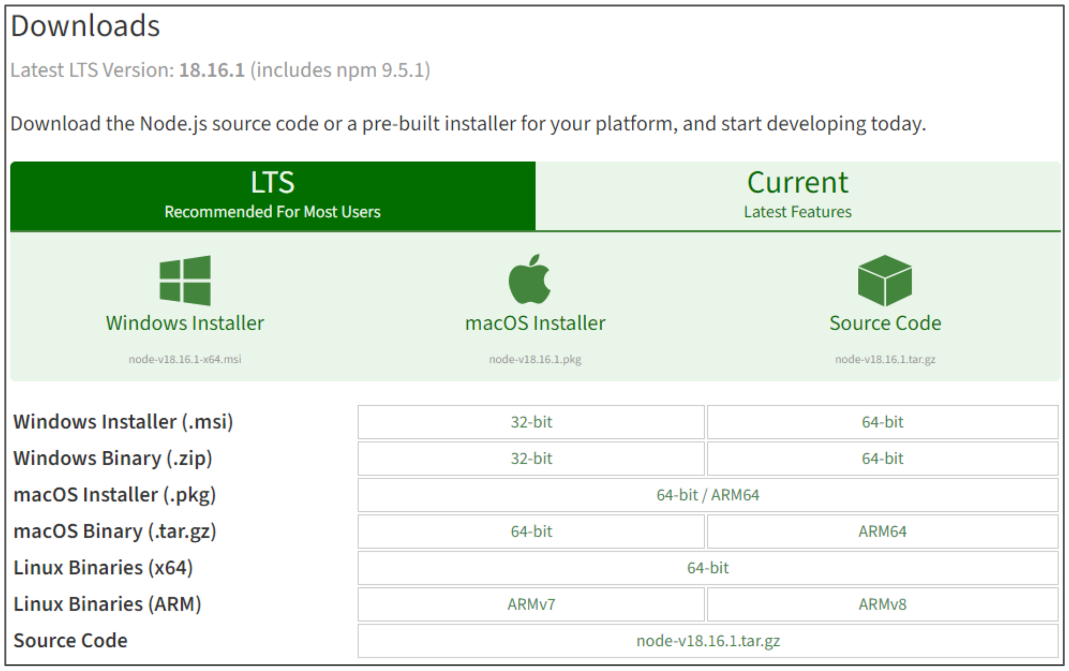
The C-WEAP can invoke a defined set of WEAP system APIs following the procedure described and exemplified in Section 6.

4. Componentized-WEAP Source Code Installation

The following software frameworks and tools need to be installed for executing the C-WEAP software application from the source code.

4.1. NodeJS

Download the NodeJS framework for the Windows 64-bit OS (MSI or ZIP) from https://nodejs.org/en/download/ (see Figure 2). At the time of preparing this user guide, the latest version of the NodeJS framework is *18.16.1*. It also includes *npm 9.5.1* (Node Package Management). After downloading, use the default choices to install NodeJS.


Downloads

Latest LTS Version: **18.16.1** (includes npm 9.5.1)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v18.16.1.tar.gz	

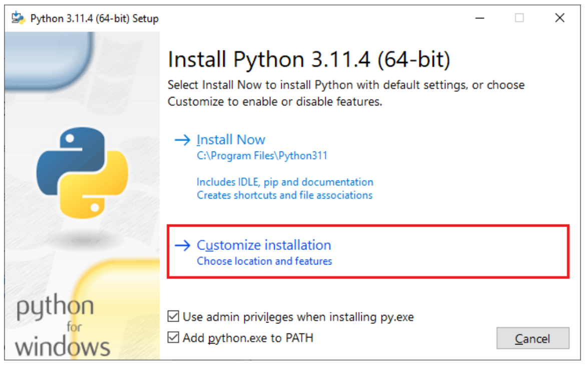
Figure 2. NodeJS download page (https://nodejs.org/en/download/).

4.2. Python

Installing Python requires multiple installations in the order provided below.

Download and install (run as administrator) **Python** from https://www.Python.org/downloads/. (current version is *3.11.4*)

As shown in Figure 3, select customize installation in the first window. Then, leave the default selections and click on the next button. Finally, make sure to check the "Add Python to environment variables" option, set a proper location and click on the install button.


Python 3.11.4 (64-bit) Setup

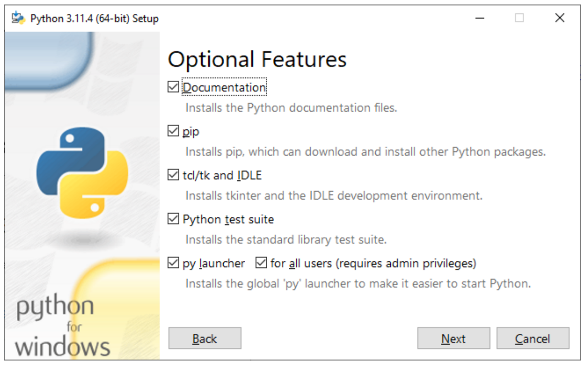
Install Python 3.11.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Program Files\Python311
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python for windows
Cancel

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

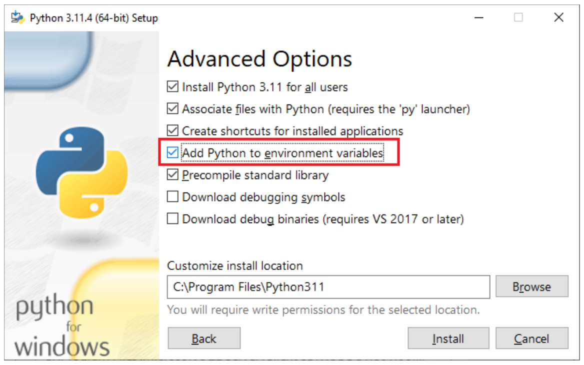
(a) select Customize installation



Optional Features

☑ Documentation
Installs the Python documentation files.
☑ pip
Installs pip, which can download and install other Python packages.
☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.
☑ Python test suite
Installs the standard library test suite.
☑ py launcher ☑ for all users (requires admin privileges)
Installs the global 'py' launcher to make it easier to start Python.

(b) Leave the default selections


Python 3.11.4 (64-bit) Setup

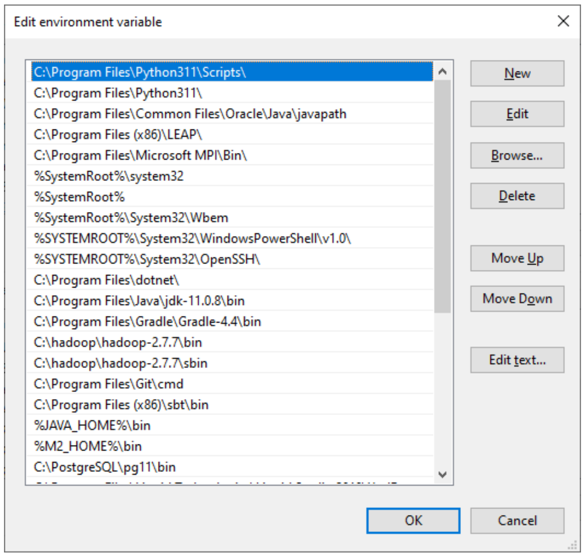
Advanced Options

☑ Install Python 3.11 for all users
☑ Associate files with Python (requires the 'py' launcher)
☑ Create shortcuts for installed applications
☑ Add Python to environment variables
☑ Precompile standard library
☐ Download debugging symbols
☐ Download debug binaries (requires VS 2017 or later)

Customize install location
C:\Program Files\Python311
You will require write permissions for the selected location.

(c) check "Add Python to environment variables"
Figure 3. Install Python.

Note: Check the Python installation by running "Python --version" in the command prompt. The version of the installed Python must be printed. If it does not show anything, the Python path must be added to the system environment variables (as shown in Figure 4).


Edit environment variable

```
C:\Program Files\Python311\Scripts\
C:\Program Files\Python311\
C:\Program Files\Common Files\Oracle\Java\javapath
C:\Program Files (x86)\LEAP\
C:\Program Files\Microsoft MPI\Bin\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\dotnet\
C:\Program Files\Java\jdk-11.0.8\bin
C:\Program Files\Gradle\Gradle-4.4\bin
C:\hadoop\hadoop-2.7.7\bin
C:\hadoop\hadoop-2.7.7\sbin
C:\Program Files\Git\cmd
C:\Program Files (x86)\sbt\bin
%JAVA_HOME%\bin
%M2_HOME%\bin
C:\PostgreSQL\pg11\bin
```

Figure 4. Python path in the system variables.

4.3. TypeScript

Run the following commands using the Windows Command Prompt (cmd) as follows:

Step 1: *npm install typescript --global*

Step 2: *npm install node-gyp --global*

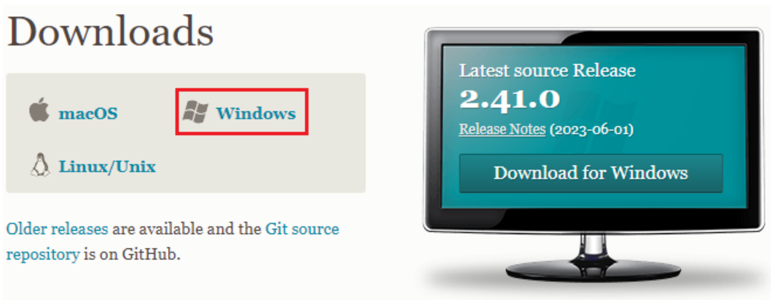
Run the following command using Windows PowerShell (run as administrator)

Step 1: *npm install --global --production windows-build-tools*

Note: Download & Install Visual C++ from here if there is an error in executing the previous step. Also, this step may take a long time (e.g., ~15-30 minutes) and further require multiple runs.

4.4. Git

Download and install the **Git** version control from https://git-scm.com/downloads. At the time of preparing this user guide, the latest version is *2.41.0*. Use the default choices in the installation steps.



Older releases are available and the Git source repository is on GitHub.

Figure 5. Git download page.

4.5. VS-Code

Different IDEs can be used for code development. We recommend the VS-Code editor, and it is used in the rest of this User Guide. Download the Windows version of the VS-Code from https://code.visualstudio.com/Download. Use the default choices in the installation steps.

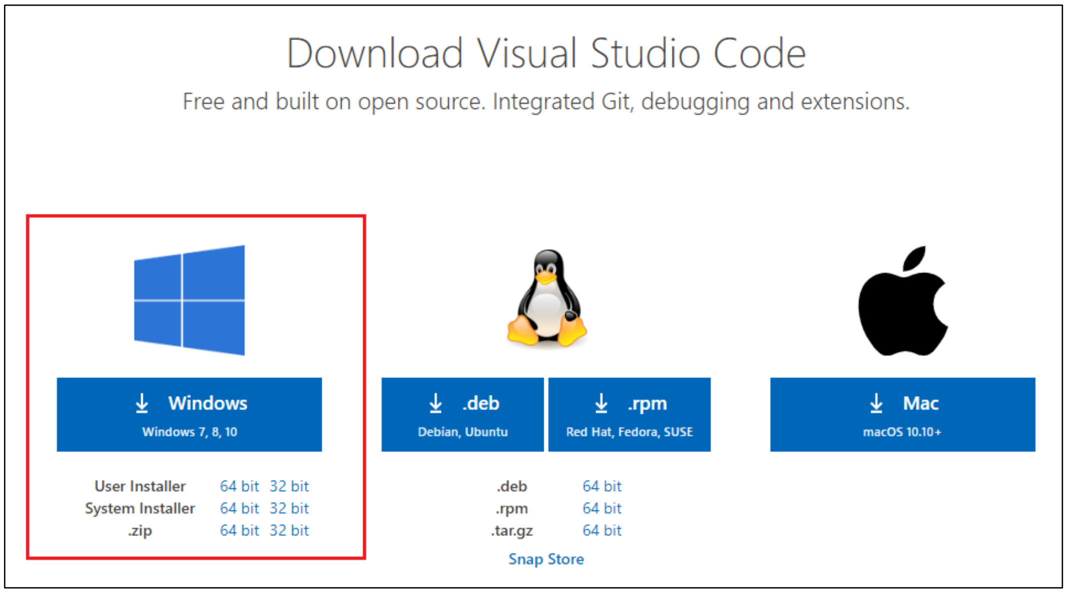
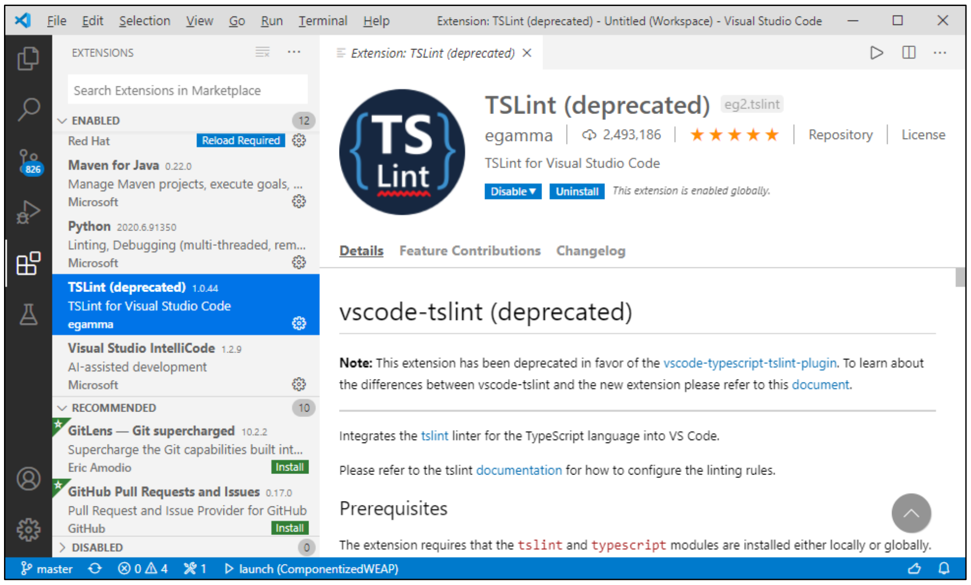

Download Visual Studio Code
Free and built on open source. Integrated Git, debugging and extensions.
Windows
Windows 7, 8, 10
.deb
Debian, Ubuntu
.rpm
Red Hat, Fedora, SUSE
Mac
macOS 10.10+
User Installer 64 bit 32 bit
System Installer 64 bit 32 bit
.zip 64 bit 32 bit
.deb 64 bit
.rpm 64 bit
.tar.gz 64 bit
Snap Store

Figure 6. VS-Code Editor download page.

After installing the VS-Code, some extensions must be installed (e.g., *TSLint*), and some are recommended to be installed (e.g., *Code Runner*). As shown in Figure 7, open the VS-Code editor, go to the extension page, type **TSLint**, and click on the install button. The same can be done for the **Code Runner** extension.


File Edit Selection View Go Run Terminal Help
Extension: TSLint (deprecated) - Untitled (Workspace) - Visual Studio Code
EXTENSIONS
Search Extensions in Marketplace
ENABLED 12
Red Hat Reload Required
Maven for Java 0.22.0
Manage Maven projects, execute goals, ...
Microsoft
Python 2020.6.91350
Linting, Debugging (multi-threaded, rem...
Microsoft
TSLint (deprecated) 1.0.44
TSLint for Visual Studio Code
egamma
Visual Studio IntelliCode 1.2.9
AI-assisted development
Microsoft
RECOMMENDED 10
GitLens — Git supercharged 10.2.2
Supercharge the Git capabilities built int...
Eric Amodio Install
GitHub Pull Requests and Issues 0.17.0
Pull Request and Issue Provider for GitHub
GitHub Install
DISABLED 0
Extension: TSLint (deprecated)
TSLint (deprecated) eg2.tslint
egamma 2,493,186 ★★★★★ Repository License
TSLint for Visual Studio Code
Disable Uninstall This extension is enabled globally.
Details Feature Contributions Changelog
master ⊗ 0 ⚠ 4 1 launch (ComponentizedWEAP)

vscode-tslint (deprecated)

Note: This extension has been deprecated in favor of the vscode-typescript-tslint-plugin. To learn about the differences between vscode-tslint and the new extension please refer to this document.

Integrates the tslint linter for the TypeScript language into VS Code.

Please refer to the tslint documentation for how to configure the linting rules.

Prerequisites

The extension requires that the `tslint` and `typescript` modules are installed either locally or globally.

Figure 7. VS-Code editor, extensions page.

Note: The execution policy in the Windows OS Client must be changed to RemoteSigned to be able to run the script. For more information, see About Execution Policy and Set Execution Policy pages. So, open Windows PowerShell (run as administrator), and run:

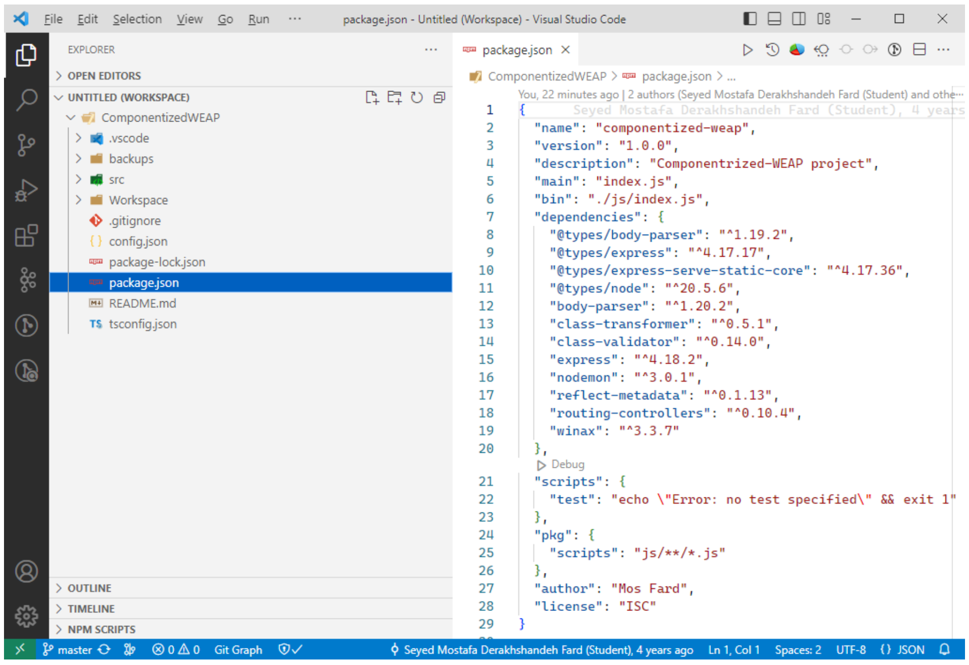
- `Set-ExecutionPolicy RemoteSigned`

4.6. Download Source Code

To download the C-WEAP framework from GitHub in the VS-Code editor, follow the following steps:

1) Open VS-Code
2) Press CTRL+SHIFT+P (View/Command Palette...) and type "*`git:Clone`*"
3) Enter the C-WEAP git URL (https://github.com/comses/ComponentizedWEAP.git); contact hss@asu.edu for access.
4) Select a folder for the project to be uploaded

After downloading the source code, you should see the folders similar to Figure 8 in the Explorer window of the VS-Code editor.


File Edit Selection View Go Run ···
package.json - Untitled (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
UNTITLED (WORKSPACE)
ComponentizedWEAP
.vscode
backups
src
Workspace
.gitignore
config.json
package-lock.json
package.json
README.md
tsconfig.json
OUTLINE
TIMELINE
NPM SCRIPTS
ComponentizedWEAP > package.json > ...
You, 22 minutes ago | 2 authors (Seyed Mostafa Derakhshandeh Fard (Student) and othe···
Debug
master
Git Graph
Seyed Mostafa Derakhshandeh Fard (Student), 4 years ago
Ln 1, Col 1
Spaces: 2
UTF-8
{} JSON

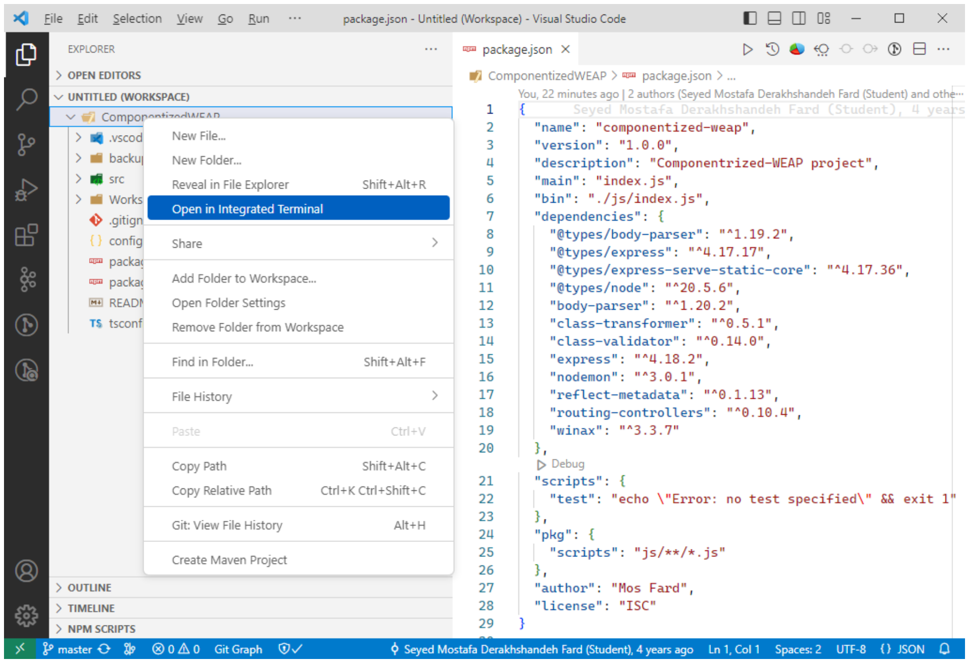
```
{
  "name": "componentized-weap",
  "version": "1.0.0",
  "description": "Componentrized-WEAP project",
  "main": "index.js",
  "bin": "./js/index.js",
  "dependencies": {
    "@types/body-parser": "^1.19.2",
    "@types/express": "^4.17.17",
    "@types/express-serve-static-core": "^4.17.36",
    "@types/node": "^20.5.6",
    "body-parser": "^1.20.2",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "express": "^4.18.2",
    "nodemon": "^3.0.1",
    "reflect-metadata": "^0.1.13",
    "routing-controllers": "^0.10.4",
    "winax": "^3.3.7"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "pkg": {
    "scripts": "js/**/*.js"
  },
  "author": "Mos Fard",
  "license": "ISC"
}
```

Figure 8. The C-WEAP project directory in the VS-Code editor.

4.7. Update the C-WEAP Packages

After downloading the C-WEAP framework for the first time, or after updating any third-party packages (the *dependencies* section of the "**./package.json**" file, see Figure 8), the following steps must be applied to generate required packages and files.

Step 1: Right-click on the project folder in the Explorer window (or right clock in the blank area of the project in the Explorer window), and select **Open in Terminal** (as shown in Figure 9). It will open the terminal windows of the VS-Code for the C-WEAP project.

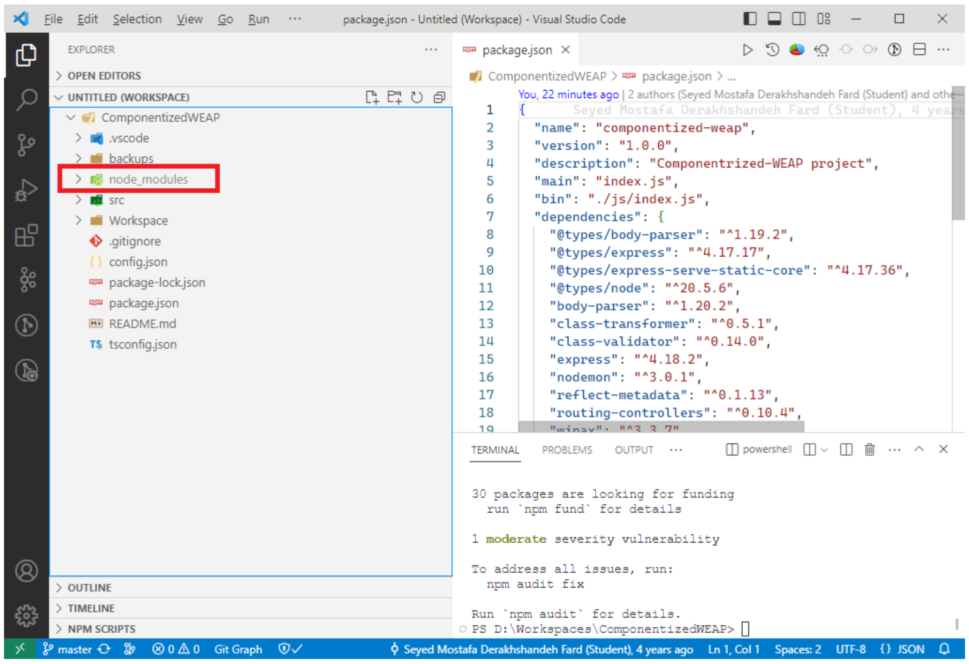

File Edit Selection View Go Run ···
package.json - Untitled (Workspace) - Visual Studio Code
EXPLORER
OPEN EDITORS
UNTITLED (WORKSPACE)
ComponentizedWEAP
New File...
New Folder...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Share
Add Folder to Workspace...
Open Folder Settings
Remove Folder from Workspace
Find in Folder... Shift+Alt+F
File History
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Git: View File History Alt+H
Create Maven Project
OUTLINE
TIMELINE
NPM SCRIPTS
master
Git Graph
Ln 1, Col 1
Spaces: 2
UTF-8
{} JSON

```
{
  "name": "componentized-weap",
  "version": "1.0.0",
  "description": "Componentrized-WEAP project",
  "main": "index.js",
  "bin": "./js/index.js",
  "dependencies": {
    "@types/body-parser": "^1.19.2",
    "@types/express": "^4.17.17",
    "@types/express-serve-static-core": "^4.17.36",
    "@types/node": "^20.5.6",
    "body-parser": "^1.20.2",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "express": "^4.18.2",
    "nodemon": "^3.0.1",
    "reflect-metadata": "^0.1.13",
    "routing-controllers": "^0.10.4",
    "winax": "^3.3.7"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "pkg": {
    "scripts": "js/**/*.js"
  },
  "author": "Mos Fard",
  "license": "ISC"
}
```

Figure 9. Open Terminal for a project in the VS-Code editor.

Step 2: Run the following command in the Terminal. As shown in Figure 10, it will generate the “node_modules” package in the root directory. The “node_modules” package includes all third-party packages defined in the dependency section of the “package.json” file.

- *`npm install`*

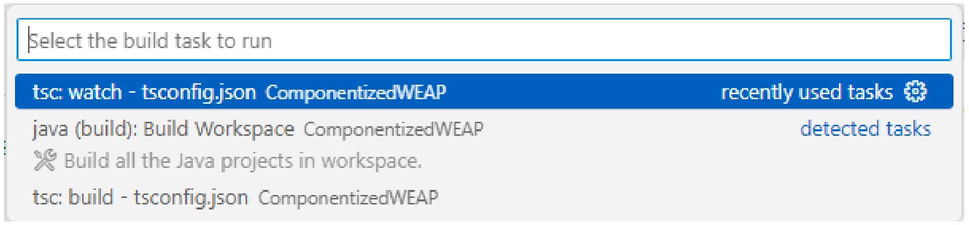


```json
{
  "name": "componentized-weap",
  "version": "1.0.0",
  "description": "Componentrized-WEAP project",
  "main": "index.js",
  "bin": "./js/index.js",
  "dependencies": {
    "@types/body-parser": "^1.19.2",
    "@types/express": "^4.17.17",
    "@types/express-serve-static-core": "^4.17.36",
    "@types/node": "^20.5.6",
    "body-parser": "^1.20.2",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "express": "^4.18.2",
    "nodemon": "^3.0.1",
    "reflect-metadata": "^0.1.13",
    "routing-controllers": "^0.10.4",
    "winax": "^3.3.7"
```

```
30 packages are looking for funding
  run `npm fund` for details

1 moderate severity vulnerability

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
PS D:\Workspaces\ComponentizedWEAP>
```

Figure 10. Generating “node_modules” package after each update in the dependencies.

- **Step 3:** Press [CTRL+SHIFT+B] in VS-Code editor to open the window shown in Figure 11 and select “`tsc: build - tsconfig.json ComponentizedWEAP`” or “`tsc: watch - tsconfig.json ComponentizedWEAP`”.

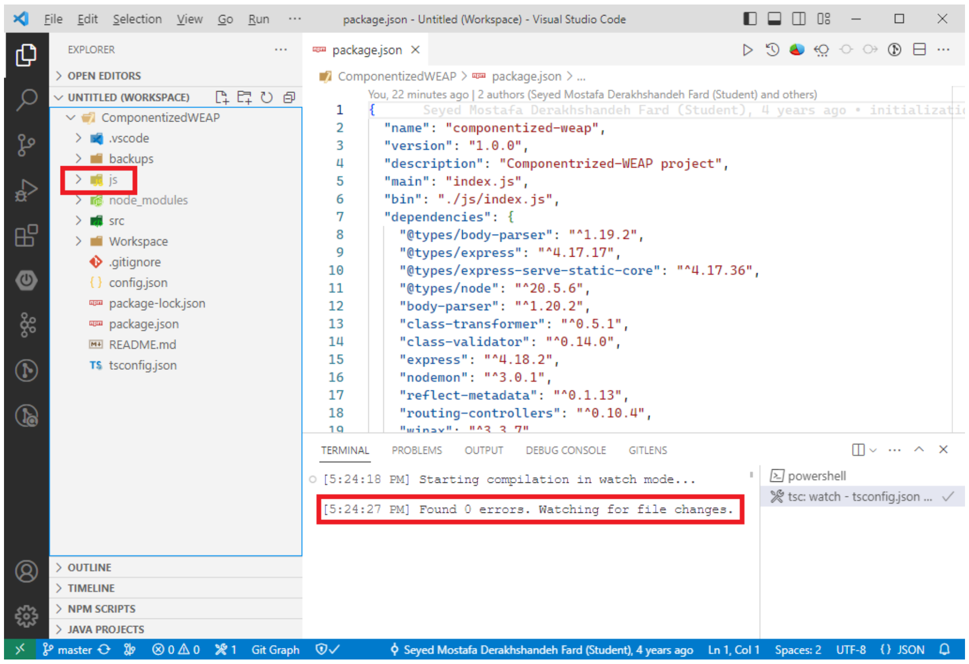


Select the build task to run
tsc: watch - tsconfig.json ComponentizedWEAP
recently used tasks
java (build): Build Workspace ComponentizedWEAP
detected tasks
Build all the Java projects in workspace.
tsc: build - tsconfig.json ComponentizedWEAP

Figure 11. Generating js package using typescript engine in VS-Code.

As shown in Figure 12, selecting this option will run a command in the terminal to generate the “js” package in the root directory. The “`Found 0 errors. Watching for ile changes.`” message must be printed in the terminal, and “js” package must be generated successfully.

Note: The C-WEAP framework is written in the TypeScript framework to use facilities which are not available in JavaScript. Finally, the TypeScript files must be converted to JavaScript files to be able to run on the server using the NodeJS framework. TypeScript does the conversion automatically using the “**./tsconfig.json**” file.



```
{
  "name": "componentized-weap",
  "version": "1.0.0",
  "description": "Componentrized-WEAP project",
  "main": "index.js",
  "bin": "./js/index.js",
  "dependencies": {
    "@types/body-parser": "^1.19.2",
    "@types/express": "^4.17.17",
    "@types/express-serve-static-core": "^4.17.36",
    "@types/node": "^20.5.6",
    "body-parser": "^1.20.2",
    "class-transformer": "^0.5.1",
    "class-validator": "^0.14.0",
    "express": "^4.18.2",
    "nodemon": "^3.0.1",
    "reflect-metadata": "^0.1.13",
    "routing-controllers": "^0.10.4",
```

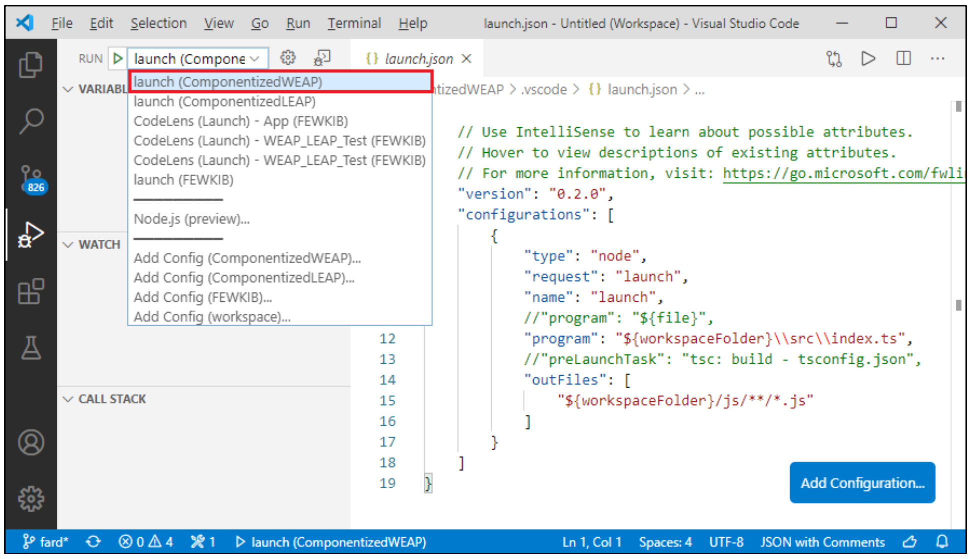
```
[5:24:18 PM] Starting compilation in watch mode...
[5:24:27 PM] Found 0 errors. Watching for file changes.
```

Figure 12. Final Componentized WEAP folder structure in VS-Code.

Note: In the C-WEAP RESTful framework, all the TypeScript files are organized under the "**./src**" folder, and all the generated JavaScript files are under the "**./js**" folder (based on the configuration in the *tsconfig.json* file). Also, the conversion is from TypeScript to ES6.

4.8. Running the C-WEAP RESTful Framework

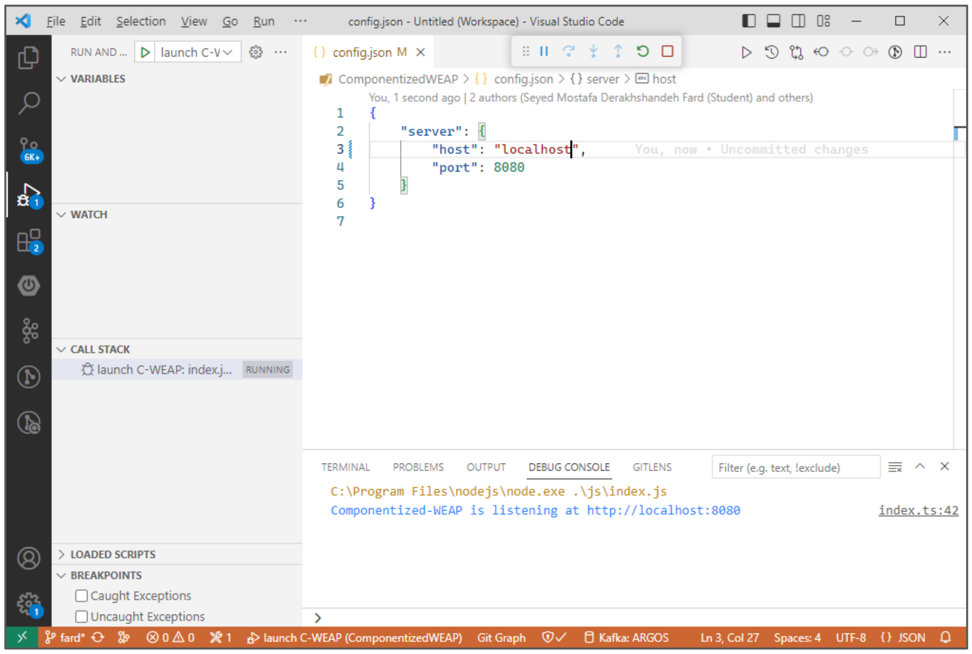
Be sure that all changes in the TypeScript files are converted to JavaScript before running the C-WEAP framework. The configuration to run the project saves in the "**./.vscode/launch.json**" file. The Run page in the VS-Code editor has a run button to execute the launch file. As can be seen in Figure 13, all open projects in the VS-Code are listed. A project must be selected in the drop-down list (e.g., Componentized-WEAP), then click on the start debugging button (▷) to run it.



```
// Use IntelliSense to learn about possible attributes.
// Hover to view descriptions of existing attributes.
// For more information, visit: https://go.microsoft.com/fwli
"version": "0.2.0",
"configurations": [
    {
        "type": "node",
        "request": "launch",
        "name": "launch",
        //"program": "${file}",
        "program": "${workspaceFolder}\\src\\index.ts",
        //"preLaunchTask": "tsc: build - tsconfig.json",
        "outFiles": [
            "${workspaceFolder}/js/**/*.js"
        ]
    }
]
}
```

Figure 13. Running a project in the VS-Code editor.

As shown in Figure 14, after running the C-WEAP framework, the “`Componentized-WEAP is listening at http://localhost:8080`” must be displayed in the DEBUG CONSOLE tab of the editor.


File Edit Selection View Go Run ···
config.json - Untitled (Workspace) - Visual Studio Code
RUN AND ... launch C-V
VARIABLES
WATCH
CALL STACK
launch C-WEAP: index.j... RUNNING
LOADED SCRIPTS
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
config.json M
ComponentizedWEAP > config.json > {} server > host
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE GITLENS
Filter (e.g. text, !exclude)
fard* launch C-WEAP (ComponentizedWEAP) Git Graph Kafka: ARGOS Ln 3, Col 27 Spaces: 4 UTF-8 JSON

You, 1 second ago | 2 authors (Seyed Mostafa Derakhshandeh Fard (Student) and others)

```
{
    "server": {
        "host": "localhost",      You, now • Uncommitted changes
        "port": 8080
    }
}
```

```
C:\Program Files\nodejs\node.exe .\js\index.js
Componentized-WEAP is listening at http://localhost:8080                index.ts:42
```

Figure 14. The C-WEAP framework after running in the VS-Code editor.

Note: The WEAP system must be running before running the C-WEAP framework. (Running the C-WEAP prior to running the WEAP system produces unexpected results).

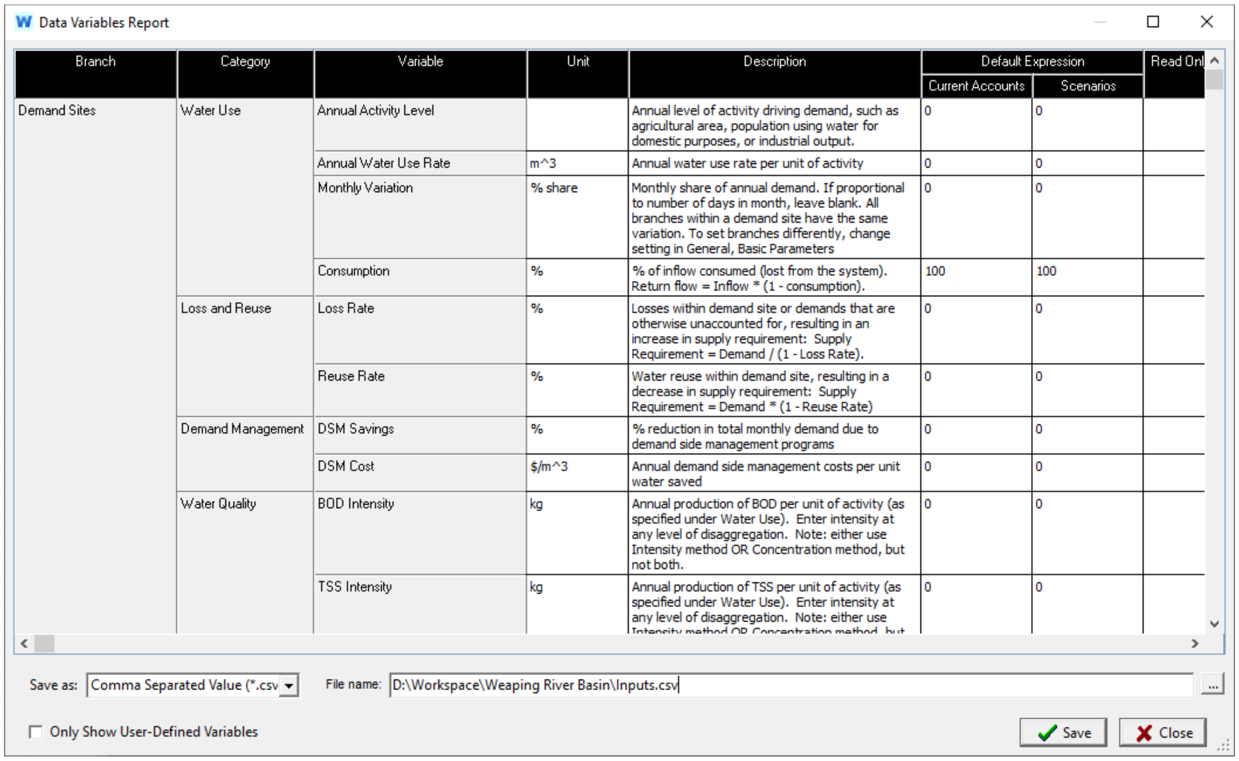
Note: In some cases, setting “localhost” for the host property in the config.json file does not work correctly. Setting host property to “[::1]” will solve the problem.

5. Define the C-WEAP Model Configuration

For each WEAP project that is going to be used in the C-WEAP framework, the required configuration files must be defined under the “Workspace” folder (see Section 3). First, a folder must be defined with the same name as the WEAP project (called the project folder). The “`Inputs.csv`” and “`Outputs.csv`” files must be defined under the project folder if we need to set some properties of the input and/or output variables of different components. The “Data Variable Report” form can be used to define the “`Inputs.csv`”, automatically. To do that, click on the menu items “`Edit->Data Variable->Report`“ in the WEAP IDE. As shown in Figure 15, use the “`Comma Separated Value (*.csv)`” for the “Save as” property, then store the csv file under the project folder. This csv file has a specific structure that the C-WEAP framework will parse during run time for extracting the *Min*, *Max*, *Time Scale*, and *User Defined* properties for the variables defined for different components (these variables are definable in the WEAP IDE, but there are not accessible via WEAP APIs). The “`Inputs.csv`” file has some extra properties (e.g., Category, Description, etc.), which are not important for the C-WEAP (Just leave them as they are).

The “`Outputs.csv`” file must be defined by the user, and it has six properties. They are six properties that are used in the “`Inputs.csv`” file, as well. The properties are “**Branch**“ to define the entity type, “**Variable**“ to define the variable name, “**Min**” and “**Max**” to define the acceptable range for the variable, “**Time Scale**” to define the time granularity of the variable, and “**User-Defined?**” to show that the variable is defined by user or it is a default variable in the WEAP system. They must be in the presented order (first Branch, then Variable, and so on).

Note: Without defining the inputs.csv/outputs.csv file, all input/output variables will have a *Yearly* time-step in the C-WEAP framework.



Branch	Category	Variable	Unit	Description	Default Expression		Read Onl
					Current Accounts	Scenarios	
Demand Sites	Water Use	Annual Activity Level		Annual level of activity driving demand, such as agricultural area, population using water for domestic purposes, or industrial output.	0	0	
		Annual Water Use Rate	m^3	Annual water use rate per unit of activity	0	0	
		Monthly Variation	% share	Monthly share of annual demand. If proportional to number of days in month, leave blank. All branches within a demand site have the same variation. To set branches differently, change setting in General, Basic Parameters	0	0	
		Consumption	%	% of inflow consumed (lost from the system). Return flow = Inflow * (1 - consumption).	100	100	
	Loss and Reuse	Loss Rate	%	Losses within demand site or demands that are otherwise unaccounted for, resulting in an increase in supply requirement: Supply Requirement = Demand / (1 - Loss Rate).	0	0	
		Reuse Rate	%	Water reuse within demand site, resulting in a decrease in supply requirement: Supply Requirement = Demand * (1 - Reuse Rate)	0	0	
	Demand Management	DSM Savings	%	% reduction in total monthly demand due to demand side management programs	0	0	
		DSM Cost	$/m^3	Annual demand side management costs per unit water saved	0	0	
	Water Quality	BOD Intensity	kg	Annual production of BOD per unit of activity (as specified under Water Use). Enter intensity at any level of disaggregation. Note: either use Intensity method OR Concentration method, but not both.	0	0	
		TSS Intensity	kg	Annual production of TSS per unit of activity (as specified under Water Use). Enter intensity at any level of disaggregation. Note: either use Intensity method OR Concentration method, but	0	0	

Save as: Comma Separated Value (*.csv

File name: D:\Workspace\Weaping River Basin\Inputs.csv

☐ Only Show User-Defined Variables

Figure 15. The C-WEAP framework after running in the VS-Code editor.

6. Modules

The C-WEAP APIs are categorized into six modules related to different parts of the WEAP system or a subset of WEAP entities. The modules are *Project*, *Version*, *Key*, *Node*, *Link*, and *Flow*. Each module has a set of APIs to read/write data from/to the WEAP system. The used WEAP APIs to develop the C-WEAP RESTful framework is listed in Appendix A.

The URL patterns for five API types are shown in Table 1. The pattern inside each open and close pair bracket is optional. In the pattern of the URLs, constants are written in *PascalCase* style; parameters start with colons and written in *camelCase* style; query parameters (to apply to some filters on returned data) written after the question mark by *Key=Value* (*camelCase* style for the *Key* part). All URLs start with constant "`/Water`". The NodeType, LinkType, FlowType, VariableType, and subNodeType (which are bold) in the patterns must be replaced by a valid value from Table 2. In the Flow URLs, the `subNodeType` uses to access a specific collection of sub-nodes, and then use *:subNodeName* to select one. For example, the URL "`/Water/demo/DemandSites/phoenix`" returns the *phoenix* demand site's data of the *demo* project. The data of a variable can be retrieved by mentioning the name of the variable and the intended scenario. Query parameters can be used to filter the returned data (the years and time-steps).

Table 1. URL pattern for different types of APIs.

Category	URL Patterns
Project	/Water[/:projectName[/Run]]
Version	/Water/:projectName/Versions[/:versionName/Revert]
Key	/Water/:projectName/Keys[/:KeyName/:scenarioName[/Expression]]
Node	/Water/:projectName/**NodeType**[/:nodeName[/**VariableType**[/:variableName/:scenarioName[/Expression][?startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]]]]
Link	/Water/:projectName/**LinkType**[/:sourceName/:targetName[/**VariableType**[/:variableName/:scenarioName[/Expression][?startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]]]]
Flow	/Water/:projectName/**FlowType**[/:flowName[/**subNodeType**[/:subNodeName]][/**VariableType**[/:variableName/:scenarioName[/Expression][?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]]]]

Table 2. Type-Values for the patterns of the APIs.

Type	Values
NodeType	Catchments, DemandSites, Groundwaters, Reservoirs, OtherSupplies, WastewaterTreatments
LinkType	Transmissions, Runoffs, ReturnFlows
FlowType	Rivers, Diversions
VariableType	Inputs, Outputs
subNodeType	Reaches, Reservoirs, RunOfRiverHydros, StreamflowGauges, FlowRequirements

All the URLs contain http://(hostname):(port). In our examples, the hostname is "*localhost*", and the port is "*8080*". To test the APIs, the WEAP system and the C-WEAP RESTful framework run first. Then the APIs are called by the Postman tool. Also, the "*Weaping River Basin*" project is using as the WEAPWEAP project to test the APIs. The Schematic view of this project is presented in Figure 16.

The C-WEAP framework always checks the existence of all parameters (e.g., :projectName, :variableName, etc.) in the URL. For example, the C-WEAP framework first checks the existence of the project "*Weaping River Basin*", then the river "*Weaping River*", then the input variable "*Headflow*", and finally the scenario "*Current Accounts*" in the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/Rivers/Weaping%20River/Inputs/Headflow/Current%20Accounts`". The corresponding error message (with status code 404) will return in the case of not existing a parameter.

Also, for updating APIs, the new values must be set in the body of the request for the URL with PUT methods.

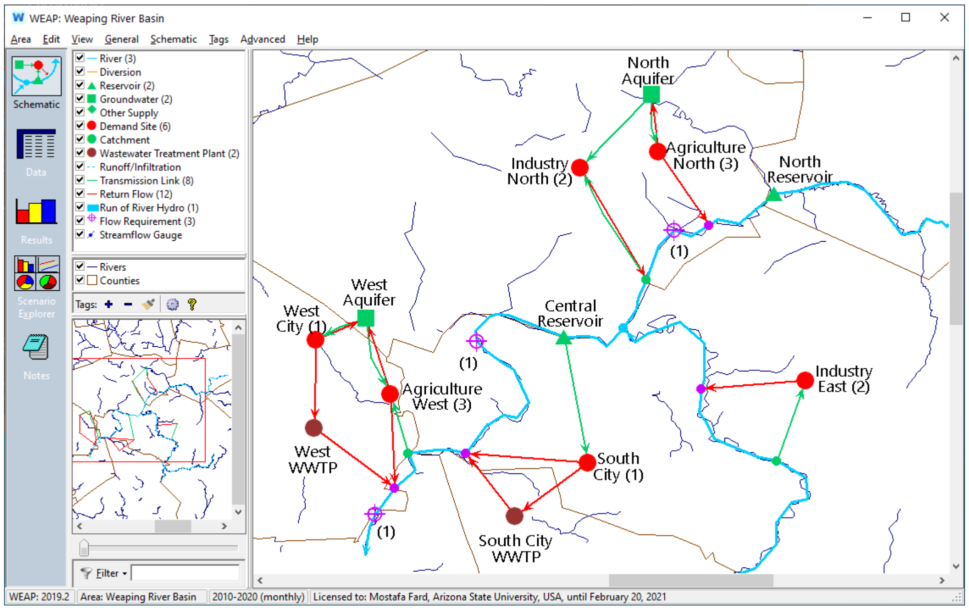


Figure 16. The “*Weaping River Basin*” project in the WEAP system.

Any application can be used to call the APIs (for example, typing a URL in the address bar of a web browser and hitting the Enter for the GET type requests), but we use the Postman tool (see https://www.postman.com/). As shown in Figure 17, the API method, the URL, the parameters, and the body of the request (for PUT requests) can specify in the Postman (and some other features that we do not use).

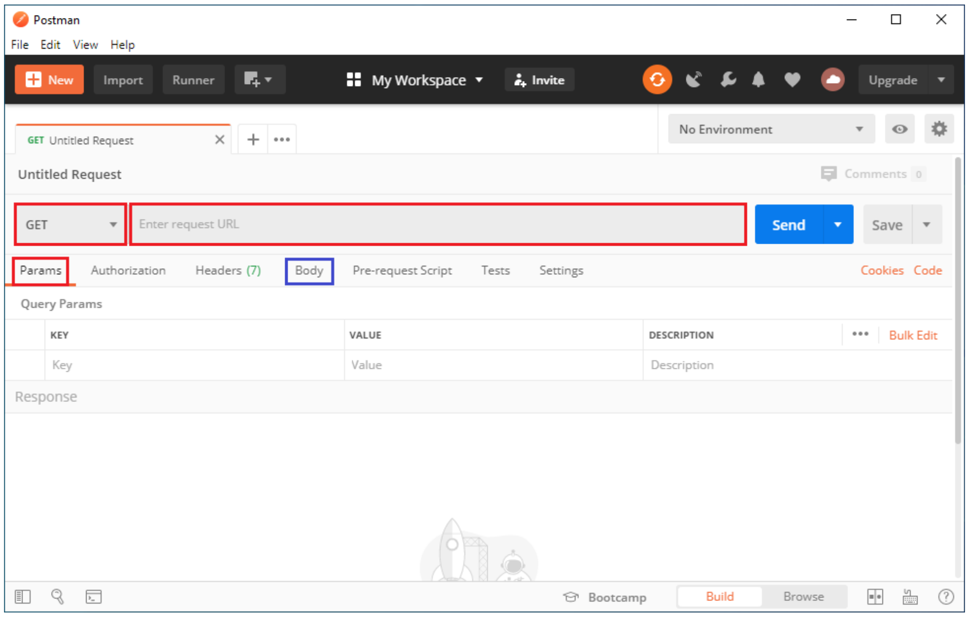
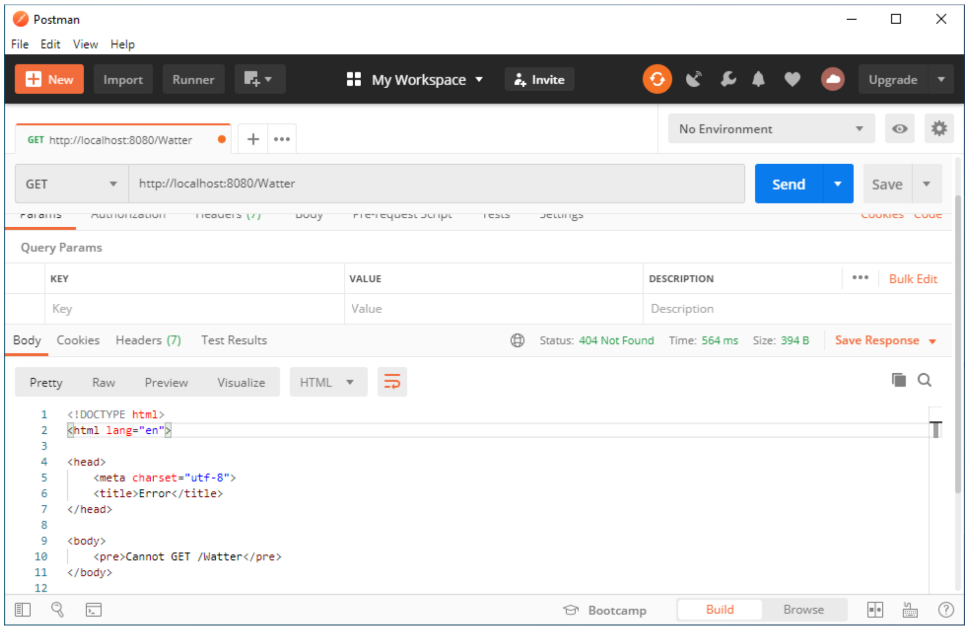


Figure 17. The Postman tool environment.

Note: Using an incorrect URL makes “`404 Not Found`” response (incorrect hostname, port, constant, etc., in the URL). For example, Figure 18 shows the situation that the URL “`http://localhost:8080/Watter`” that has a mistake. It shows the message “Cannot GET /Watter” in the web browser. Indeed, it is requesting an undefined API.



GET http://localhost:8080/Watter

Status: 404 Not Found Time: 564 ms Size: 394 B

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Cannot GET /Watter</pre>
</body>

```

Figure 18. Calling an incorrect URL ("/Watter") in the Postman.

Figure 19 presents individual domain model classes defined in the C-WEAP framework for receiving/sending data from/to the API caller.

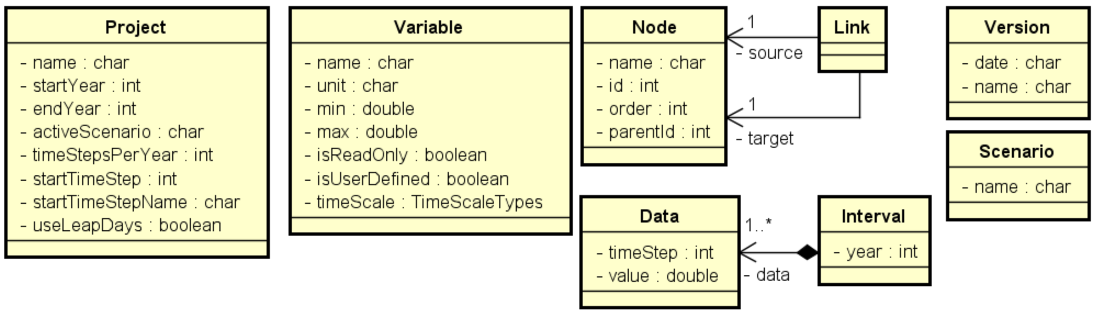


Figure 19. Domain Model classes in the C-WEAP framework.

6.1. Project

The C-WEAP APIs related to the Project category are listed in Table 3.

Table 3. List of APIs for the Project module.

#	Method	URL	Return Value/s	Description
P1	GET	`/Water`	`String[]`	Get the name of all projects
P2	GET	`/Water/:projectName`	Project	Get properties of a project
P3	GET	`/Water/:projectName/Run`	`Boolean`	Run a project
P4	PUT	`/Water/:projectName`	`Boolean`	Update properties of a project, by setting new values for the Project object in the body of the request

Example: As an example, the API P1 from Table 3 is presented here. Figure 20 shows the available projects in the WEAP system. Calling the URL "`http://localhost:8080/Water`" in Postman (or web browser) returns the project names while the C-WEAP is running (see Figure 21). The list of projects shown in Figure 21 varies depending on the projects that are available in the WEAP system. It is shown in the Postman that the status of the request is "`200 OK`", the time to get data is "`128 ms`" (which can be different in different calls), and the size of the response is "`361 Byte`". The time and response time measurements can vary depending on the host computer and other factors.

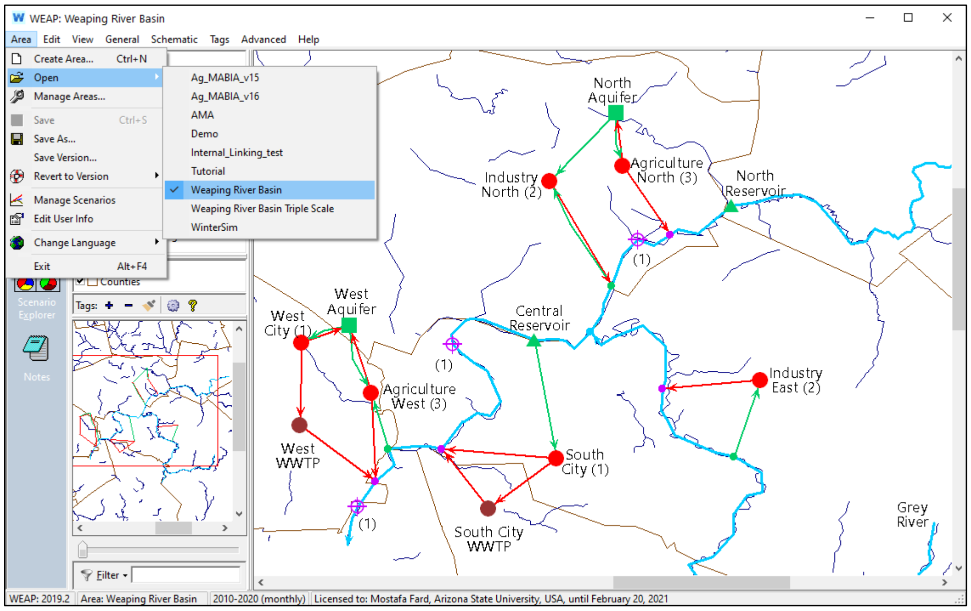
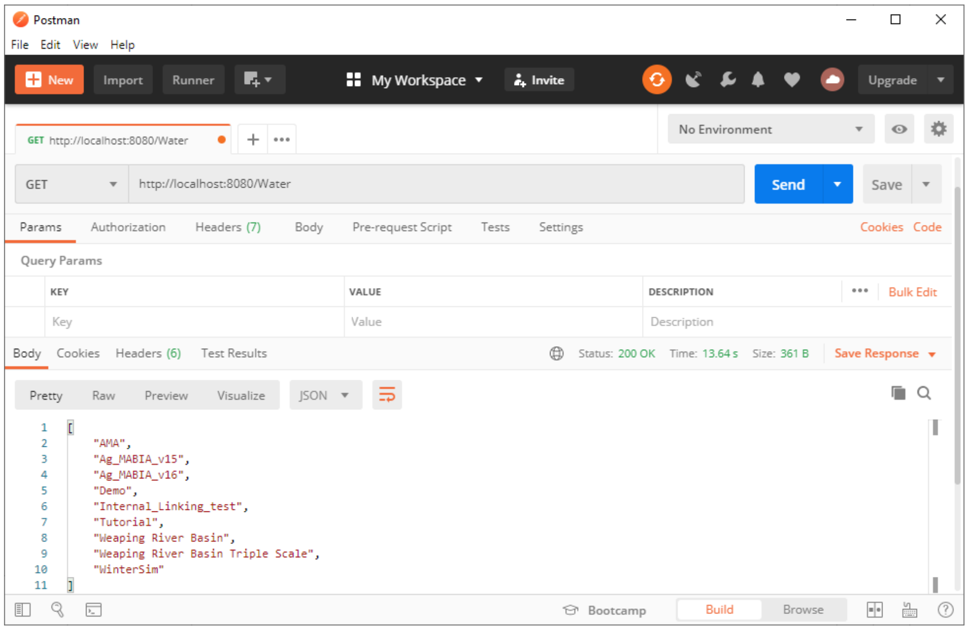


Figure 20. The projects in the WEAP system.



GET http://localhost:8080/Water

Status: 200 OK Time: 13.64 s Size: 361 B

```
[
    "AMA",
    "Ag_MABIA_v15",
    "Ag_MABIA_v16",
    "Demo",
    "Internal_Linking_test",
    "Tutorial",
    "Weaping River Basin",
    "Weaping River Basin Triple Scale",
    "WinterSim"
]
```

Figure 21. Calling the URL "`/Water`" in the Postman.

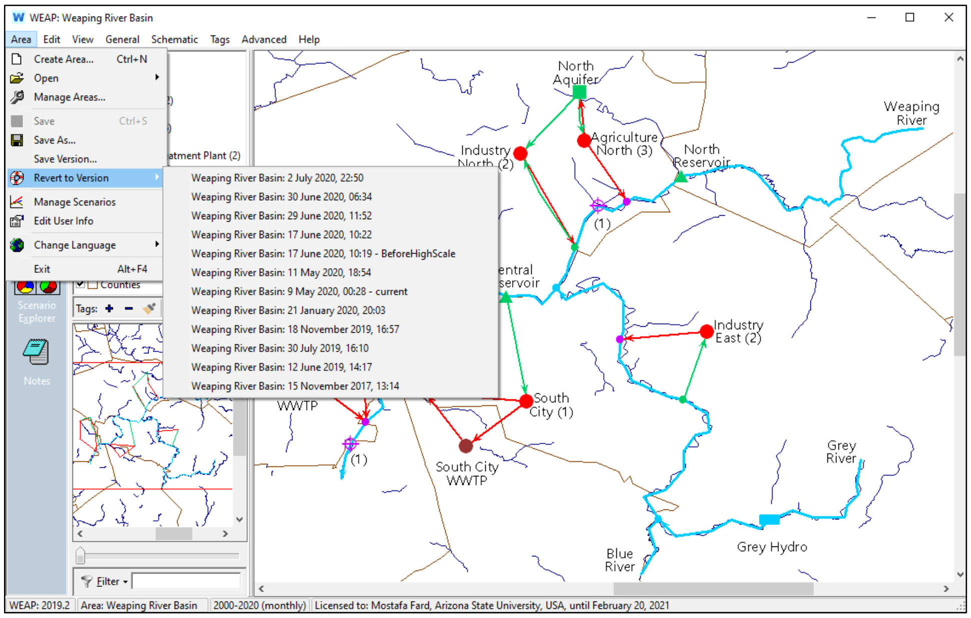
6.2. Version

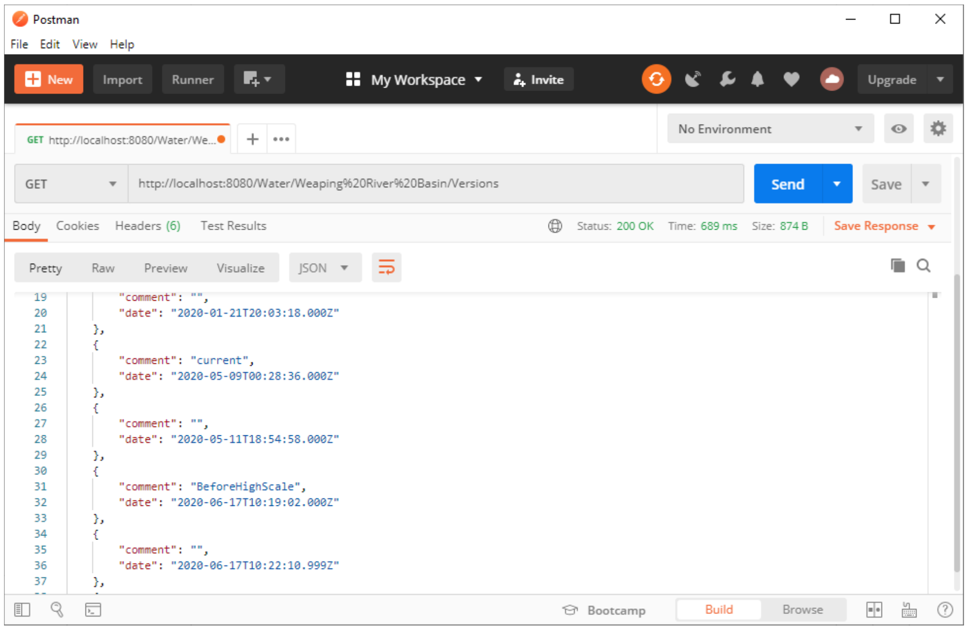
The C-WEAP APIs related to the Version category are listed in Table 4. The name of the version is the concatenated `date` and the `name` properties of the `Version` UML class shown in Figure 19.

Table 4. List of APIs for the Version module.

#	Method	URL	Return Value/s	Description
V1	`GET`	`/Water/:projectName/Versions`	`Version[]`	Get the list of all versions of a project
V2	`GET`	`/Water/:projectName/Versions/:versionName`	`Version`	Get a version of a project
V3	`PUT`	`/Water/:projectName/Versions/:versionName/Revert`	`Boolean`	Revert to a version for a project

Example: As an example, the API V1 from Table 4 is presented here. Figure 22 shows the available versions defined in the "*Weaping River Basin*" project in the WEAP system. Calling the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/Versions`" in Postman (or web browser) returns the list of versions for the project, ordered by ascending on the date (see Figure 23).


WEAP: Weaping River Basin
Area Edit View General Schematic Tags Advanced Help
Create Area... Ctrl+N
Open
Manage Areas...
Save Ctrl+S
Save As...
Save Version...
Revert to Version
Manage Scenarios
Edit User Info
Change Language
Exit Alt+F4
Weaping River Basin: 2 July 2020, 22:50
Weaping River Basin: 30 June 2020, 06:34
Weaping River Basin: 29 June 2020, 11:52
Weaping River Basin: 17 June 2020, 10:22
Weaping River Basin: 17 June 2020, 10:19 - BeforeHighScale
Weaping River Basin: 11 May 2020, 18:54
Weaping River Basin: 9 May 2020, 00:28 - current
Weaping River Basin: 21 January 2020, 20:03
Weaping River Basin: 18 November 2019, 16:57
Weaping River Basin: 30 July 2019, 16:10
Weaping River Basin: 12 June 2019, 14:17
Weaping River Basin: 15 November 2017, 13:14
Scenario Explorer
Notes
Counties
Tags:
Filter
North Aquifer
Weaping River
Agriculture North (3)
Industry North (2)
North Reservoir
(1)
entral servoir
Industry East (2)
South City (1)
WWTP
South City WWTP
Grey River
Grey Hydro
Blue River
WEAP: 2019.2 | Area: Weaping River Basin | 2000-2020 (monthly) | Licensed to: Mostafa Fard, Arizona State University, USA, until February 20, 2021
Figure 22. The versions of "*Weaping River Basin*" project in the WEAP system.


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
GET http://localhost:8080/Water/We...
No Environment
GET
Send
Save
Body Cookies Headers (6) Test Results
Status: 200 OK Time: 689 ms Size: 874 B
Save Response
Pretty Raw Preview Visualize JSON
Bootcamp
Build
Browse

```
http://localhost:8080/Water/Weaping%20River%20Basin/Versions
```

```
        "comment": "",
        "date": "2020-01-21T20:03:18.000Z"
    },
    {
        "comment": "current",
        "date": "2020-05-09T00:28:36.000Z"
    },
    {
        "comment": "",
        "date": "2020-05-11T18:54:58.000Z"
    },
    {
        "comment": "BeforeHighScale",
        "date": "2020-06-17T10:19:02.000Z"
    },
    {
        "comment": "",
        "date": "2020-06-17T10:22:10.999Z"
    },
```

Figure 23. Calling the URL "`/Water/Weaping%20River%20Basin/Versions`" in the Postman.

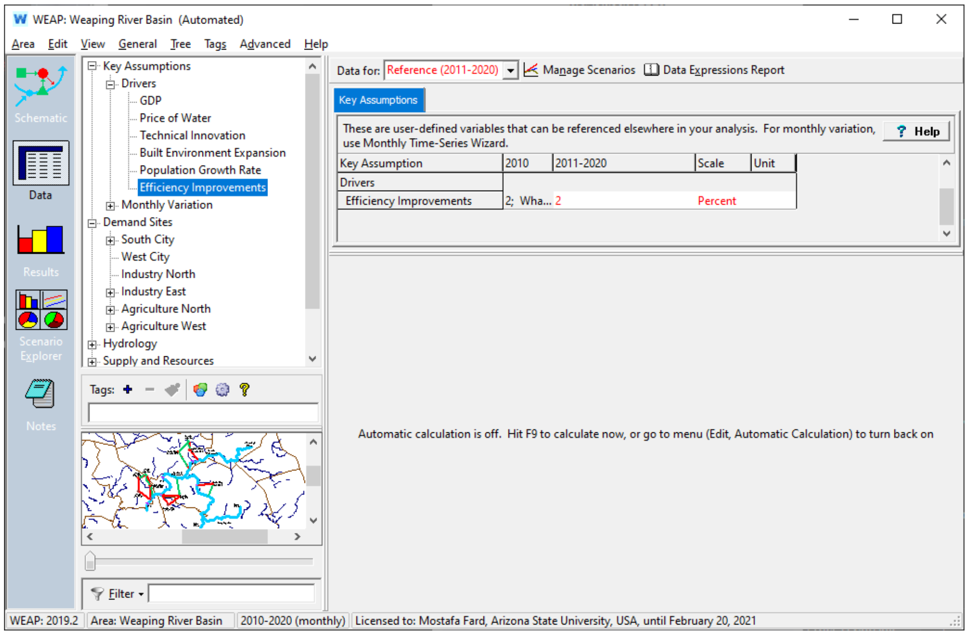
6.3. Key

The C-WEAP APIs related to the Key category are listed in Table 5.

Table 5. List of APIs for the Key module.

#	Method	URL	Return Value/s	Description
K1	`GET`	`/Water/:projectName/Keys`	`String[]`	Get the list of all keys in a project
K2	`GET`	`/Water/:projectName/Keys/:keyName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of a key in a project
K3	`GET`	`/Water/:projectName/Keys/:keyName/:scenarioName/Expression`	`String`	Get the expression of a key in a project
K4	`PUT`	`/Water/:projectName/Keys/:keyName/:scenarioName`	`Boolean`	Update the values of a key in a project, by setting new values in the body of the request
K5	`PUT`	`/Water/:projectName/Keys/:keyName/:scenarioName/Expression`	`Boolean`	Update the expression of a key in a project, by setting new value in the body of the request

Example: As an example, the API K5 from Table 5 is presented here. Figure 24 shows the value of the "Efficiency Improvements" key defined in the "*Weaping River Basin*" project for "*Reference*" scenario in the WEAP system. We sare going to change the value. So, by calling the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/Keys/Efficiency%20Improvements/Reference/Expression`" and set the body of the request to {"value":"3.5"} in Postman (see Figure 25). This URL will change the current value of the Efficiency Improvements to 3.5 (see Figure 26), and returns true if the API executes successfully.



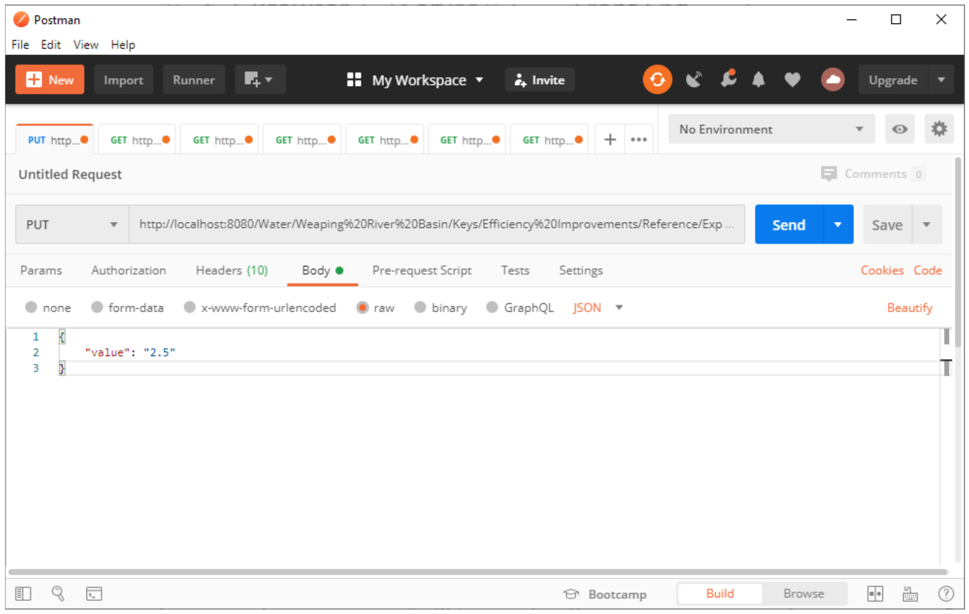
These are user-defined variables that can be referenced elsewhere in your analysis. For monthly variation, use Monthly Time-Series Wizard.

Key Assumption	2010	2011-2020	Scale	Unit
Drivers				
Efficiency Improvements	2; Wha...	2		Percent

Automatic calculation is off. Hit F9 to calculate now, or go to menu (Edit, Automatic Calculation) to turn back on

WEAP: 2019.2 | Area: Weaping River Basin | 2010-2020 (monthly) | Licensed to: Mostafa Fard, Arizona State University, USA, until February 20, 2021

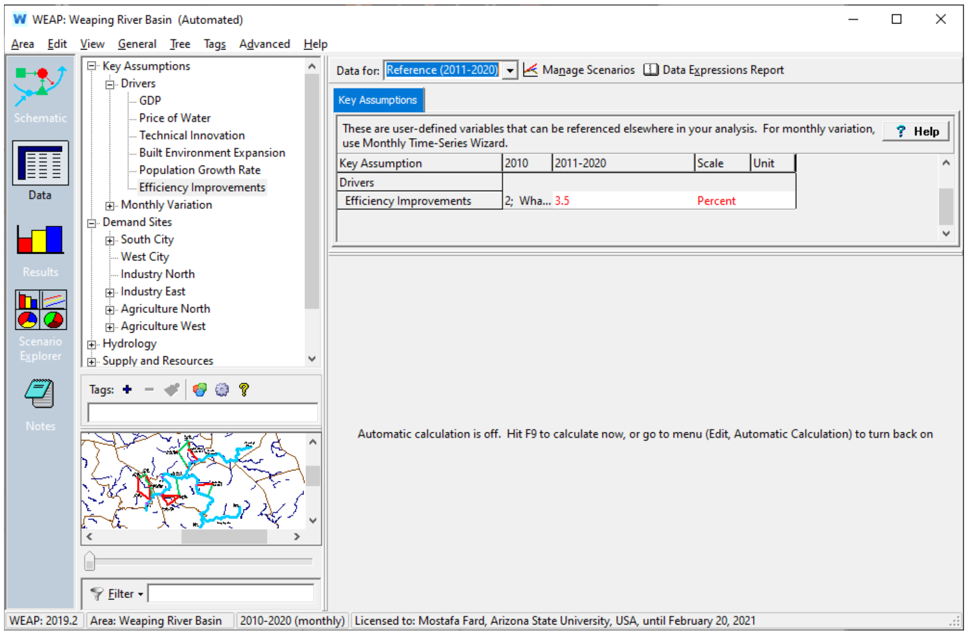
Figure 24. The Efficiency Improvements key in the "*Weaping River Basin*" project in the WEAP system.



PUT http://localhost:8080/Water/Weaping%20River%20Basin/Keys/Efficiency%20Improvements/Reference/Exp ...

```
{
    "value": "2.5"
}
```

Figure 25. Calling the URL "/Water/Weaping%20River%20Basin/Keys/Efficiency%20Improvements/Reference/Expression " in the Postman.



These are user-defined variables that can be referenced elsewhere in your analysis. For monthly variation, use Monthly Time-Series Wizard.

Key Assumption	2010	2011-2020	Scale	Unit
Drivers				
Efficiency Improvements	2;	Wha... 3.5		Percent

Automatic calculation is off. Hit F9 to calculate now, or go to menu (Edit, Automatic Calculation) to turn back on

WEAP: 2019.2 | Area: Weaping River Basin | 2010-2020 (monthly) | Licensed to: Mostafa Fard, Arizona State University, USA, until February 20, 2021

Figure 26. The key changing in the "*Weaping River Basin*" project after using the URL in Figure 25.

6.4. Node

The C-WEAP APIs related to the Node category are listed in Table 6. As mentioned before, one of the values from Table 2 (Node Type) must be replaced with the **NodeType** in the URLs shown in Table 6.

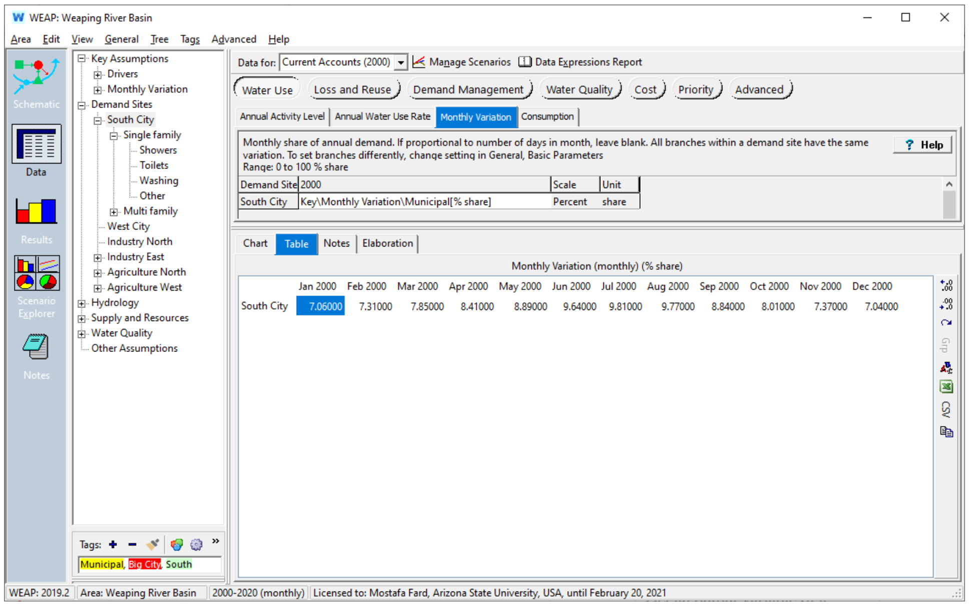
Table 6. List of APIs for the Node module.

#	Method	URL	Return Value/s	Description
N1	`GET`	`/Water/:projectName/`**`NodeType`**	`Node[]`	Get the list of all nodeType components in a project
N2	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName`	`Node`	Get a nodeType component in a project
N3	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs`	`Variable[]`	Get a list of all input variables of a nodeType component in a project
N4	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs/:variableName`	`Variable`	Get an input variable of a nodeType component in a project
N5	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an input variable of a nodeType component in a project
N6	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs/:variableName/:scenarioName/Expression`	`String`	Get the expression of an input variable of a nodeType component in a project
N7	`PUT`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs/:variableName/:scenarioName`	`Boolean`	Update the values of an input variable of a nodeType component in a project, by setting new values in the body of the request
N8	`PUT`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Inputs/:variableName/:scenarioName/Expression`	`Boolean`	Update the expression of an input variable of a nodeType component in a project, by

				setting new value in the body of the request
N9	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Outputs`	`Variable[]`	Get a list of all output variables of a nodeType component in a project
N10	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Outputs/:variableName`	`Variable`	Get an output variable of a nodeType component in a project
N11	`GET`	`/Water/:projectName/`**`NodeType`**`/:nodeName/Outputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an output variable of a nodeType component in a project

Note: The returned values for APIs N5 and N11 can be filtered using query parameters. It means, adding the "`?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N`" at the end of the URL.

Example: As an example, the API N3 from Table 6 for the DemandSite is presented here. Figure 27 shows the input variables of the "*South City*" demand site in the "*Weaping River Basin*" project in the WEAP system. Calling the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/DemandSites/South%20City/Inputs`" in Postman (or web browser) returns the list of variables (see Figure 28).


WEAP: Weaping River Basin

Data for: Current Accounts (2000)

Monthly share of annual demand. If proportional to number of days in month, leave blank. All branches within a demand site have the same variation. To set branches differently, change setting in General, Basic Parameters
Range: 0 to 100 % share

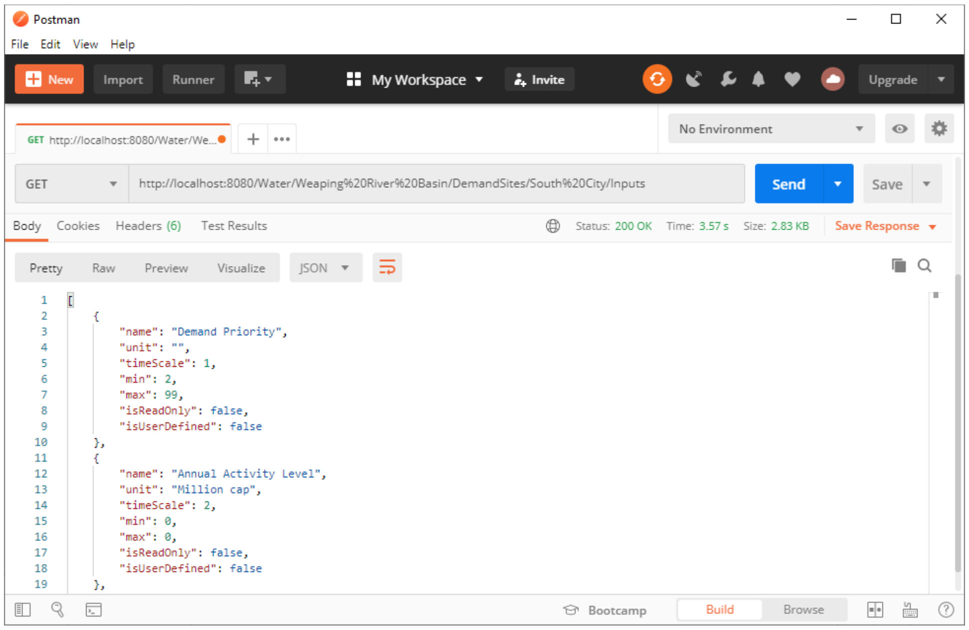
Demand Site	2000	Scale	Unit
South City	Key\Monthly Variation\Municipal[% share]	Percent	share

Monthly Variation (monthly) (% share)

	Jan 2000	Feb 2000	Mar 2000	Apr 2000	May 2000	Jun 2000	Jul 2000	Aug 2000	Sep 2000	Oct 2000	Nov 2000	Dec 2000
South City	7.06000	7.31000	7.85000	8.41000	8.89000	9.64000	9.81000	9.77000	8.84000	8.01000	7.37000	7.04000

WEAP: 2019.2 | Area: Weaping River Basin | 2000-2020 (monthly) | Licensed to: Mostafa Fard, Arizona State University, USA, until February 20, 2021

Figure 27. The input variables of the "*South City*" demand site in the "*Weaping River Basin*" project in the WEAP system.



GET http://localhost:8080/Water/Weaping%20River%20Basin/DemandSites/South%20City/Inputs

Status: 200 OK Time: 3.57 s Size: 2.83 KB

```
[
    {
        "name": "Demand Priority",
        "unit": "",
        "timeScale": 1,
        "min": 2,
        "max": 99,
        "isReadOnly": false,
        "isUserDefined": false
    },
    {
        "name": "Annual Activity Level",
        "unit": "Million cap",
        "timeScale": 2,
        "min": 0,
        "max": 0,
        "isReadOnly": false,
        "isUserDefined": false
    },
```

Figure 28. Calling the URL "/Water/Weaping%20River%20Basin/DemandSites/South%20City/Inputs/Monthly%20Variation/Current%20Accounts?&startYear=2000&endYear=2000" in the Postman.

6.5. Link

The C-WEAP APIs related to the Link category are listed in Table 7. As mentioned before, one of the values from Table 2 (Link Type) must be replaced with the **LinkType** in the URLs in Table 7.

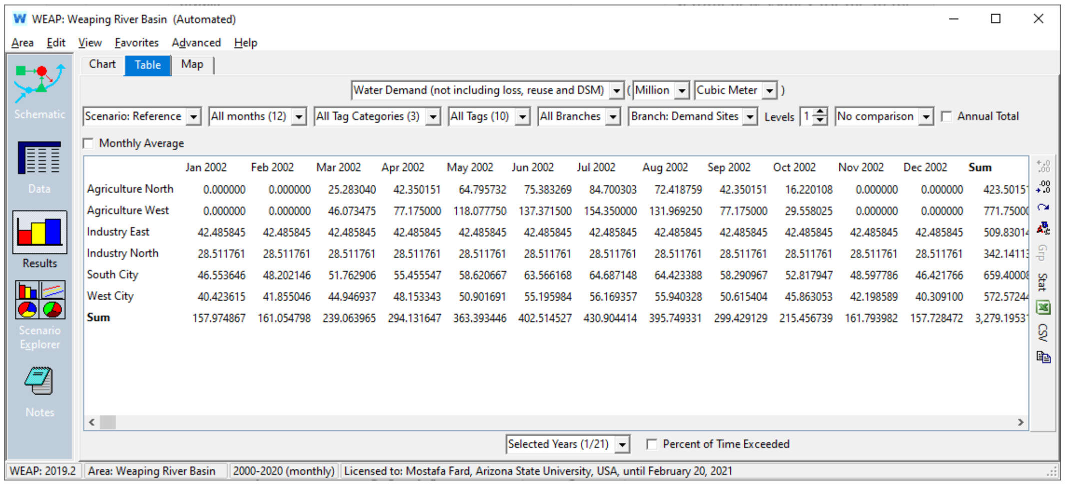
Table 7. List of APIs for the Link module.

#	Method	URL	Return Value/s	Description
L1	`GET`	`/Water/:projectName/`**`LinkType`**	`Link[]`	Get the list of all linkType components in a project
L2	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName`	`Link`	Get a linkType component in a project
L3	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs`	`Variable[]`	Get a list of all input variables of a linkType component in a project
L4	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs/:variableName`	`Variable`	Get an input variable of a linkType component in a project
L5	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an input variable of a linkType component in a project
L6	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs/:variableName/:scenarioName/Expression`	`String`	Get the expression of an input variable of a linkType component in a project
L7	`PUT`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs/:variableName/:scenarioName`	`Boolean`	Update the values of an input variable of a linkType component in a project, by setting new values in the body of the request

L8	`PUT`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Inputs/:variableName/:scenarioName/Expression`	`Boolean`	Update the expression of an input variable of a linkType component in a project, by setting new value in the body of the request
L9	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Outputs`	`Variable[]`	Get a list of all output variables of a linkType component in a project
L10	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Outputs/:variableName`	`Variable`	Get an output variable of a linkType component in a project
L11	`GET`	`/Water/:projectName/`**`LinkType`**`/:sourceName/:targetName/Outputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an output variable of a linkType component in a project

Note: Like Node APIs, filtering can be applied to the link APIs L5 and L11 in Table 7.

Example: As an example, the API L11 from Table 7 for the DemandSite is presented here. Figure 29 shows the "*Water Demand*" output variable for the "*Reference*" scenario of the "*Weaping River Basin*" project in the WEAP system. Calling the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/DemandSites/South%20City/Outputs/Water%20Demand/Reference?&startYear=2011&endYear=2011`" in Postman (or web browser) returns the list of intervals filtered for the year 2011 (see Figure 30).


WEAP: Weaping River Basin (Automated)
Area Edit View Favorites Advanced Help
Chart Table Map
Schematic
Data
Results
Scenario Explorer
Notes
Grp Stat CSV

Water Demand (not including loss, reuse and DSM) (Million Cubic Meter)

Scenario: Reference | All months (12) | All Tag Categories (3) | All Tags (10) | All Branches | Branch: Demand Sites | Levels 1 | No comparison | Annual Total

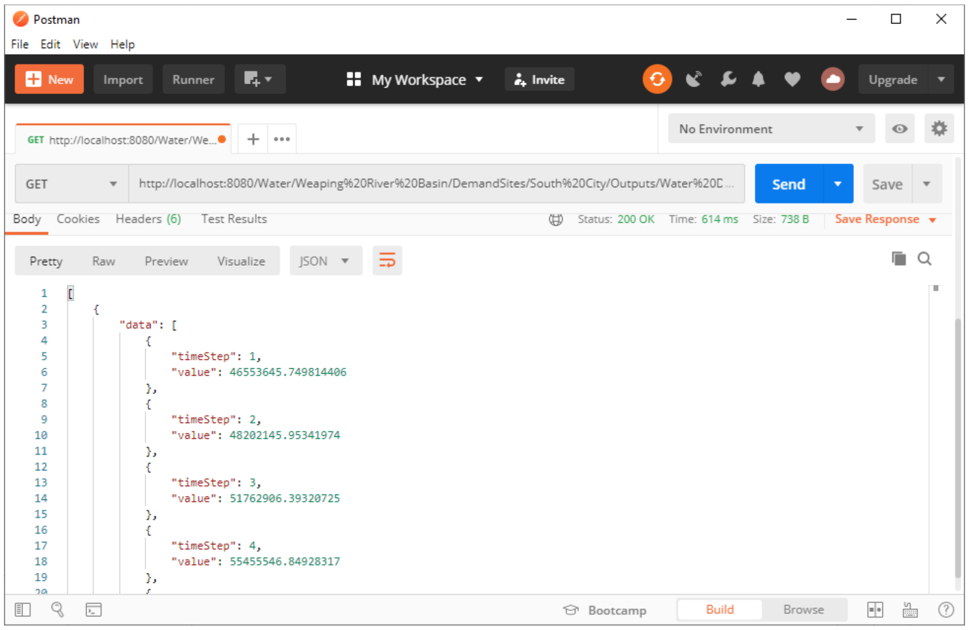
Monthly Average

	Jan 2002	Feb 2002	Mar 2002	Apr 2002	May 2002	Jun 2002	Jul 2002	Aug 2002	Sep 2002	Oct 2002	Nov 2002	Dec 2002	**Sum**
Agriculture North	0.000000	0.000000	25.283040	42.350151	64.795732	75.383269	84.700303	72.418759	42.350151	16.220108	0.000000	0.000000	423.5015
Agriculture West	0.000000	0.000000	46.073475	77.175000	118.077750	137.371500	154.350000	131.969250	77.175000	29.558025	0.000000	0.000000	771.7500
Industry East	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	42.485845	509.8301
Industry North	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	28.511761	342.1411
South City	46.553646	48.202146	51.762906	55.455547	58.620667	63.566168	64.687148	64.423388	58.290967	52.817947	48.597786	46.421766	659.4000
West City	40.423615	41.855046	44.946937	48.153343	50.901691	55.195984	56.169357	55.940328	50.615404	45.863053	42.198589	40.309100	572.5724
Sum	157.974867	161.054798	239.063965	294.131647	363.393446	402.514527	430.904414	395.749331	299.429129	215.456739	161.793982	157.728472	3,279.1953

Selected Years (1/21) | Percent of Time Exceeded

WEAP: 2019.2 | Area: Weaping River Basin | 2000-2020 (monthly) | Licensed to: Mostafa Fard, Arizona State University, USA, until February 20, 2021

Figure 29. The "*Water Demand*" output variable of the demand sites for the "*Reference*" scenario of the "*Weaping River Basin*" project in the WEAP system.



GET http://localhost:8080/Water/Weaping%20River%20Basin/DemandSites/South%20City/Outputs/Water%20D...

Status: 200 OK Time: 614 ms Size: 738 B

```
[
    {
        "data": [
            {
                "timeStep": 1,
                "value": 46553645.749814406
            },
            {
                "timeStep": 2,
                "value": 48202145.95341974
            },
            {
                "timeStep": 3,
                "value": 51762906.39320725
            },
            {
                "timeStep": 4,
                "value": 55455546.84928317
            },
            {
```

Figure 30. Calling the URL "/Water/Weaping%20River%20Basin/DemandSites/South%20City/Outputs/Water%20Demand/Reference?&startYear=2002&endYear=2002" in the Postman.

6.6. Flow

The C-WEAP APIs related to the Flow category are listed in Table 8. As mentioned before, one of the values from Table 2 (Flow Type) must be replaced with the **FlowType** in the URLs in Table 8.

Table 8. List of APIs for the Flow module.

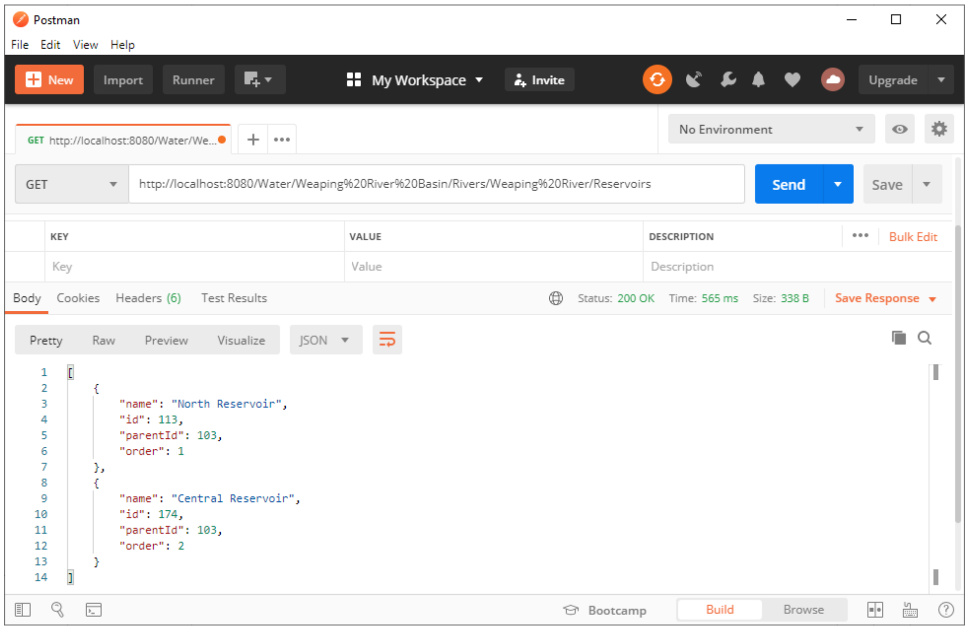
#	Method	URL	Return Value/s	Description
F1	GET	`/Water/:projectName/`**`FlowType`**	`Node[]`	Get the list of all flowType components in a project
F2	GET	`/Water/:projectName/`**`FlowType`**`/:flowName`	`Node`	Get a flowType component in a project
F3	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs`	`Variable[]`	Get a list of all input variables of a flowType component in a project
F4	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs/:variableName`	`Variable`	Get an input variable of a flowType component in a project
F5	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an input variable of a flowType component in a project
F6	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs/:variableName/:scenarioName/Expression`	`String`	Get the expression of an input variable of a flowType component in a project
F7	PUT	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs/:variableName/:scenarioName`	`Boolean`	Update the values of an input variable of a flowType component in a project, by setting new values in the body of the request

F8	PUT	`/Water/:projectName/`**`FlowType`**`/:flowName/Inputs/:variableName/:scenarioName/Expression`	Boolean	Update the expression of an input variable of a flowType component in a project, by setting new value in the body of the request
F9	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Outputs`	Variable[]	Get a list of all output variables of a flowType component in a project
F10	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Outputs/:variableName`	Variable	Get an output variable of a flowType component in a project
F11	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/Outputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	Interval[]	Get a list of all values of an output variable of a flowType component in a project
F12	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**	Node[]	Get the list of all subNodeType components in a flowType components in a project
F13	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName`	Node	Get a subNodeType component of a flowType component in a project
F14	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs`	Variable[]	Get a list of all input variables of a subNodeType component of a flowType component in a project
F15	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs/:variableName`	Variable	Get an input variable of a subNodeType component of a flowType component in a project
F16	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	Interval[]	Get a list of all values of an input variable of a subNodeType component of a flowType component in a project
F17	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs/:variableName/:scenarioName/Expression`	String	Get the expression of an input variable of a subNodeType component of a flowType component in a project
F18	PUT	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs/:variableName/:scenarioName`	Boolean	Update the values of an input variable of a subNodeType component of a flowType component in a project, by setting new values in the body of the request
F19	PUT	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Inputs/:variableName/:scenarioName/Expression`	Boolean	Update the expression of an input variable of a subNodeType component of a flowType component in a project, by setting new value in the body of the request
F20	GET	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`sunNodeType`**`/:subNodeName/Outputs`	Variable[]	Get a list of all output variables of a subNodeType component of a flowType component in a project

F21	`GET`	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`subNodeType`**`/:subNodeName/Outputs/:variableName`	`Variable`	Get an output variable of a subNodeType component of a flowType component in a project
F22	`GET`	`/Water/:projectName/`**`FlowType`**`/:flowName/`**`subNodeType`**`/:subNodeName/Outputs/:variableName/:scenarioName[?&startYear=N&endYear=N&startTimeStep=N&endTimeStep=N]`	`Interval[]`	Get a list of all values of an output variable of a subNodeType component of a flowType component in a project

Note: Like Node and Link APIs, filtering can be applied on the Flow APIs F5, F11, F16, and F22 in Table 8.

Example: As an example, the API F12 fromTable 8 the *Rivers* `FlowType` and the *Reservoirs* for the `subNodeType` are presented here. Figure 16 shows the Schematic view for the "*Weaping River Basin*" project in the WEAP system. As can be seen, there are two reservoirs on the "*Weaping River*" river. Calling the URL "`http://localhost:8080/Water/Weaping%20River%20Basin/Rivers/Weaping%20River/Reservoirs`" in Postman (or web browser) returns the list of nodes (shown in Figure 31).



Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
GET
http://localhost:8080/Water/Weaping%20River%20Basin/Rivers/Weaping%20River/Reservoirs
Send
Save
KEY
VALUE
DESCRIPTION
Bulk Edit
Body Cookies Headers (6) Test Results
Status: 200 OK Time: 565 ms Size: 338 B
Save Response
Pretty Raw Preview Visualize JSON
Bootcamp Build Browse

```
[
    {
        "name": "North Reservoir",
        "id": 113,
        "parentId": 103,
        "order": 1
    },
    {
        "name": "Central Reservoir",
        "id": 174,
        "parentId": 103,
        "order": 2
    }
]
```

Figure 31. Calling the URL "`/Water/Weaping%20River%20Basin/Rivers/Weaping%20River/Reservoirs`" in the Postman.

6.7. A simple example

The following is an example illustrating using the C-WEAP framework for making changes to the "*Weaping River Basin*" model's configuration, reaching an optimal solution.

Problem: Given the default "*Weaping River Basin*" project, what is the optimal value for the "*Efficiency Improvements*" key value to have the "*Water Demand*" result for the "*West City*" demand site in the year 2020 between 495,000,000 m^3 and 505,000,000 m^3 (500,000,000 ± 1%).

6.7.1. Model configuration

Based on the WEAP calculation, the "*Water Demand*" result for a demand site uses the "*Annual Activity Level*" and "*Annual Water Use Rate*" input values. In the "*Reference*" scenario of the project, the "*Annual Water Use Rate*" is using the "*Technical Innovation*" key, and the "*Technical Innovation*" key is using the "*Efficiency Improvements*" key to define the input data (the default value for the "*Efficiency Improvements*" key is 2). Using these value, the "*Water Demand*" is calculated by changing the "*Efficiency Improvements*" value.

6.7.2. Simulation Execution

The basic algorithm presented in Figure 32 to find the optimal "*Efficiency Improvements*" for the defined problem based on using different APIs from the C-WEAP framework. In step 1, the water demand is to be determined (calculate the optimal range for the "*Water Demand*") for the water demand ranging from 495,000,000 (WD_{lower_bound}) to 505,000,000 (named WD_{upper_bound}) values. In step 2, the default key "*Efficiency Improvements*" value is read from the WEAP system by calling the K3 API from the C-WEAP (`http://localhost:8080/Water/Weaping%20River%20Basin/Keys/Efficiency%20Improvements/Reference/Expression`). In step 3, the WEAP is executed (i.e., simulated) using the P3 (`http://localhost:8080/Water/Weaping%20River%20Basin/Run`). In step 4, the "*Water Demand*" result values for the demand site "*West City*" and scenario "*Reference*" for year 2020 are read using the N11 API from the C-WEAP (`http://localhost:8080/Water/Weaping%20River%20Basin/DemandSites/West%20City/Outputs/Water%20Demand/Reference?StartYear=2020&endYear=2020`). Also, in this step, the sum of the water demand for year 2020 must be calculated (called WD_{2020}), because the return type of calling N11 is an array of the Intervals for all time-steps values.

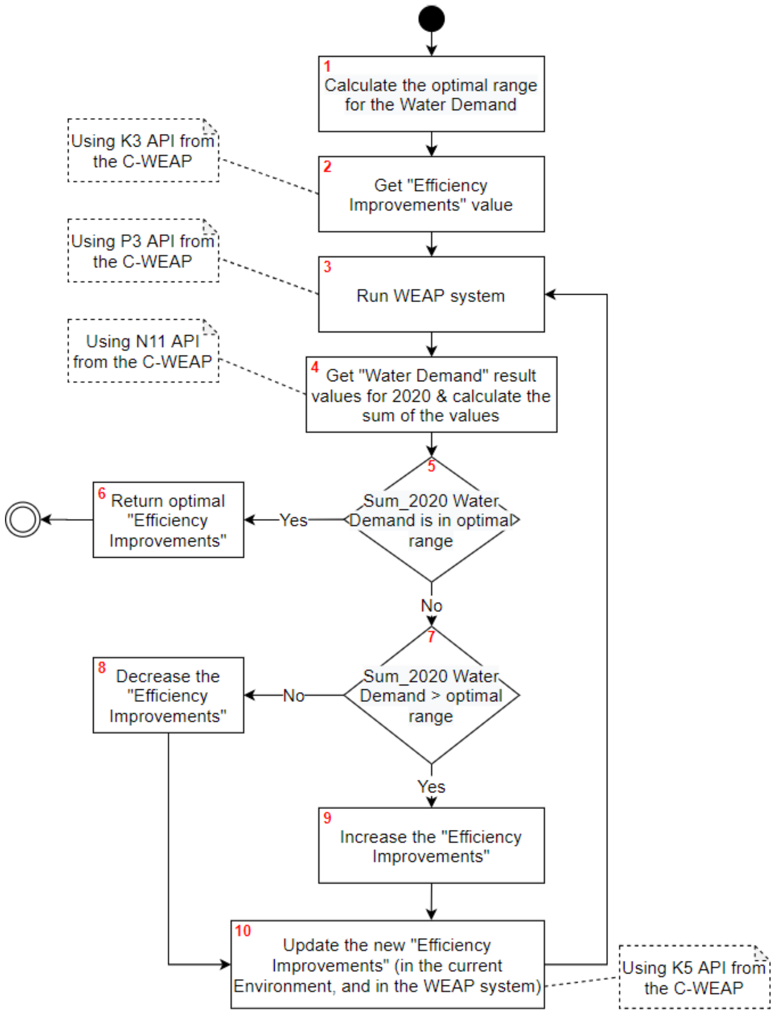

1 Calculate the optimal range for the Water Demand
2 Get "Efficiency Improvements" value
Using K3 API from the C-WEAP
3 Run WEAP system
Using P3 API from the C-WEAP
4 Get "Water Demand" result values for 2020 & calculate the sum of the values
Using N11 API from the C-WEAP
5 Sum_2020 Water Demand is in optimal range
Yes
6 Return optimal "Efficiency Improvements"
No
7 Sum_2020 Water Demand > optimal range
No
8 Decrease the "Efficiency Improvements"
Yes
9 Increase the "Efficiency Improvements"
10 Update the new "Efficiency Improvements" (in the current Environment, and in the WEAP system)
Using K5 API from the C-WEAP

Figure 32. The flowchart to solve the defined problem in section 6.7 to find optimal efficiency.

In step 5, the WD_{2020} is checked to be in the optimal water demand range (between WD_{lower_bound} and WD_{upper_bound}). If the WD_{2020} is in the optimal range, the current "*Efficiency Improvements*" value is returned as a result, in step 6. Otherwise, the "*Efficiency Improvements*" key value must be checked to be decreased/increased using a simple evaluation of the WD_{2020} value to be smaller/larger than the WD_{lower_bound}/WD_{upper_bound} value in step 7. In steps 8/steps 9, the decrease/increase amounts for the "*Efficiency Improvements*" are calculated. In step 10, the new value for the "*Efficiency Improvements*" updates using the K5 API from the C-WEAP (`http://localhost:8080/Water/Weaping%20River%20Basin/Keys/Efficiency%20Improvements/Reference/Expression`) and then the algorithm returns to step 3. The algorithm is expected to terminate with an optimal value for the "*Efficiency Improvements*".

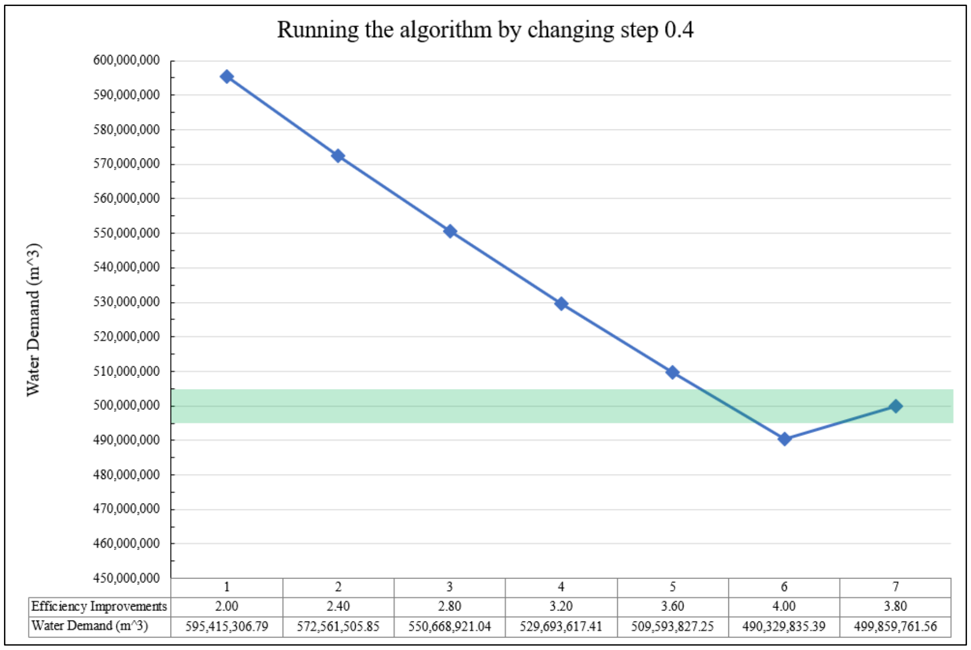
In this simulation execution, a value k for changing the "*Efficiency Improvements*" (if the current DW_{2020} value is out of the acceptance range) is set initially. Then, the k value is divided by 2 if DW_{2020} value is outside the acceptance range defined for the "*Water Demand*". For example, suppose the current DW_{2020} is larger than WD_{upper_bound}, then the value of the "*Efficiency Improvements*" will increase by x, and the WEAP system is executed. Now, suppose the DW_{2020} is smaller than WD_{lower_bound}, so the value of "*Efficiency Improvements*" will decrease by $\frac{x}{2}$ and the WEAP system is executed.

6.7.3. Model Implementation

A Java maven application is used to implement the illustrated algorithm in Figure 32. Also, the Jersey framework is used to define the RESTful API for the Java application. The full implementation is presented in Appendix B. In the source code, lines 1 to 84 are the implementation for the algorithm in Figure 32. Lines 85 to 207 are the RESTful API (using Jersey framework) for the Project, Key, and Node categories. Finally, lines 208 to 312 are the classes to define the required domain models.

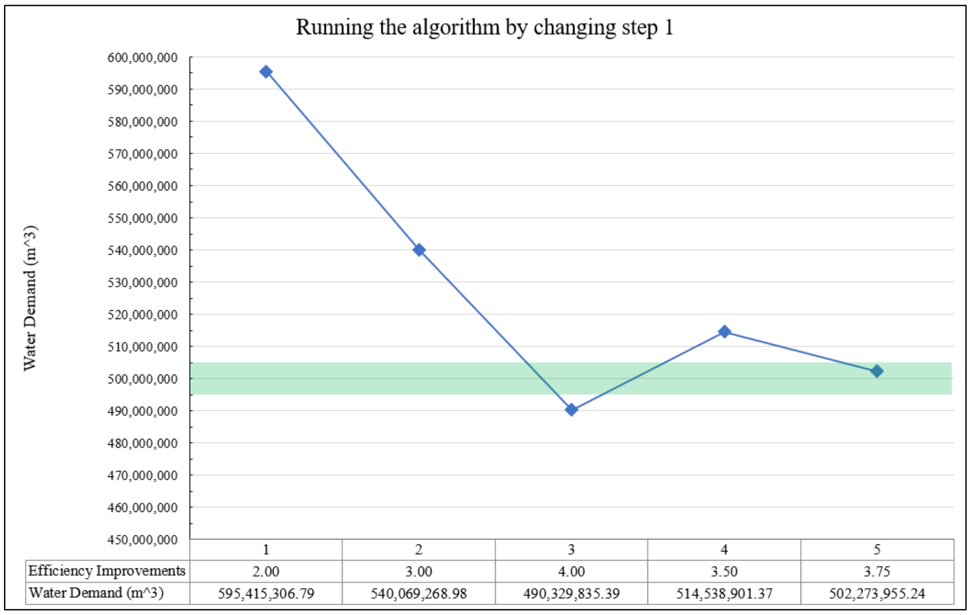
6.7.4. Simulation Results

A set of suitable values should be selected for k, the amounts by which is to be changed as long as the water demand is outside of the desired range. Three changing steps equal to 0.4, 1, and 2 are selected. The results and the trends of water demand to reach the desired ranges for three configurations are shown in Figure 33. The "*Efficiency Improvements*" is initialized to 2, and the WD_{2020} is 595,415,306.79 m^3. In Figure 33(a) the "*Efficiency Improvements*" increases by 0.4 through the sixth simulation cycle for the WD_{2020} to smaller than WD_{lower_bound}. For the seventh simulation cycle, the "*Efficiency Improvements*" decreases by 0.2. The result is the WD_{2020} falls within the acceptance range (499,859,761.56 m^3), with the final efficiency improvement equal to 3.8. Different results are shown in Figure 33(b) and (c) for changing steps 1 and 2, respectively. For both settings, the optimal efficiency improvement is 3.75. The desired range for the "*Water Demand*" values (495,000,000 to 505,000,000) is shown in the green areas in Figure 33(b) and (c).



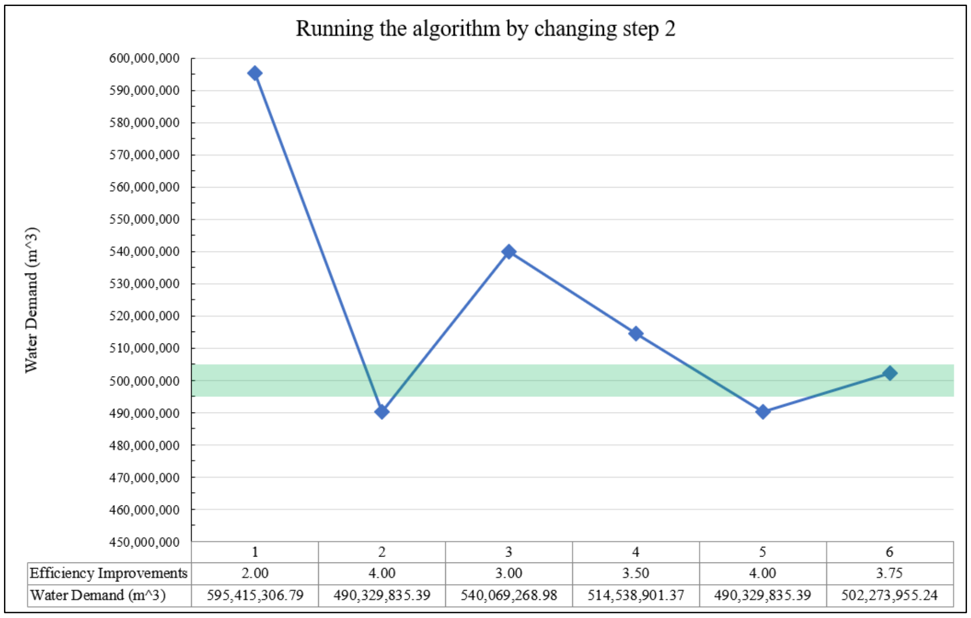
	1	2	3	4	5	6	7
Efficiency Improvements	2.00	2.40	2.80	3.20	3.60	4.00	3.80
Water Demand (m^3)	595,415,306.79	572,561,505.85	550,668,921.04	529,693,617.41	509,593,827.25	490,329,835.39	499,859,761.56

(a)



	1	2	3	4	5
Efficiency Improvements	2.00	3.00	4.00	3.50	3.75
Water Demand (m^3)	595,415,306.79	540,069,268.98	490,329,835.39	514,538,901.37	502,273,955.24

(b)



	1	2	3	4	5	6
Efficiency Improvements	2.00	4.00	3.00	3.50	4.00	3.75
Water Demand (m^3)	595,415,306.79	490,329,835.39	540,069,268.98	514,538,901.37	490,329,835.39	502,273,955.24

(c)

Figure 33. The result of running simulation with different changing step values (a) changing step equals 0.4 unit, (b) changing step equals 1 unit, (c) changing step equals 2 unit.

7. References

[1] ACIMS, "Componentized-WEAP RESTful Framework," 20 June 2020. [Online]. Available: https://acims.asu.edu/software/c-weap. [Accessed 20 June 2020].

[2] M. D. Fard and H. S. Sarjoughian, "A Web-service Framework for the Water Evaluation and Planning System," *Spring Simulation Conference (SpringSim),* pp. 1-12, 2019.

[3] M. D. Fard and H. S. Sarjoughian, "Coupling WEAP and LEAP Models using Interaction Modeling," in *SpringSim Conference*, Fairfax, VA, USA, 2020.

Appendix A

The WEAP APIs used in developing the Componentized-WEAP framework

#	API	Return Object	Category
1	`WEAP.ActiveArea`	`Area`	WEAP
2	`WEAP.ActiveArea.Name`	`String`	
3	`WEAP.WaterYearStart`	`Integer`	
4	`WEAP.ActiveScenario`	`Scenario`	
5	`WEAP.BaseYear`	`Integer`	
6	`WEAP.EndYear`	`Integer`	
7	`WEAP.TimeStepName(Id)`	`String`	
8	`WEAP.NumTimeSteps`	`Integer`	
9	`WEAP.View`	`String`	
10	`WEAP.Calculate(LastYear, LastTimestep, AlwaysCalculate)`	-	
11	`WEAP.ResultValue(BranchName:VariableName, Year, TimeStep, ScenarioName)`	`Double`	
12	`WEAP.Areas(Id)`	`Area[]`	Area
13	`WEAP.Areas.Count`	`Integer`	
14	`WEAP.Versions.Count`	`Integer`	Version
15	`WEAP.Versions(Name/Id)`	`Version`	
16	`WEAP.Versions.Exist(VersionName)`	`Boolean`	
17	`WEAP.SaveVersion(VersionName)`	-	
18	`WEAP.Versions(VersionName).Revert()`	-	
19	`WEAP.Scenarios(Id)`	`Scenario[]`	Scenario
20	`WEAP.Scenarios.Exists(ScenarioName)`	`Boolean`	
21	`WEAP.Scenarios.Add(ScenarioName)`	-	
22	`WEAP.Scenarios(ScenarioName).Delete()`	-	
23	`WEAP.Branch(BranchName)`	`Branch`	Branch
24	`WEAP.BranchExists(BranchName)`	`Boolean`	
25	`WEAP.Branch(BranchName).Children`	`Branch[]`	
26	`WEAP.Branch(BranchName).Variables`	`Variable[]`	
27	`WEAP.Branch(BranchName).Variables.Exists(VariableName)`	`Boolean`	

Appendix B

```java
public class App
{
    public static void main(String[] args) {
        OEIC oeic= new OEIC("Weaping River Basin", 500000000, 5000000, 2);
        oeic.calculate();
    }
}
```

```java
// OEIC stands for Optimal Efficiency Improvement Calculator
public class OEIC {
    private String _projectName = null;
    private double _changeEfficiencyValue = 0;

    private double _startWD = 0;
    private double _endWD = 0;

    public OEIC(String projectName, double optimalEfficiency, double efficiencyThreshold, double changeEff
    iciencyValue) {
        this._projectName = projectName;
        this._changeEfficiencyValue = changeEfficiencyValue;

        this._startWD = optimalEfficiency - efficiencyThreshold;
        this._endWD = optimalEfficiency + efficiencyThreshold;
    }

    public double calculate() {
        WaterService waterService = new WaterService();
        waterService.run(this._projectName);

        KeyService keyService = new KeyService(this._projectName);
        NodeService nodeService = new NodeService(this._projectName,ComponentTypes.DemandSite);

        double efficiencyValue = Double.valueOf(keyService.getExpression("Efficiency Improvements", "Refer
        ence"));
        Interval[] waterDemandValues = nodeService.getOutputValues("West City", "Water Demand", "Reference
        ", new FilterParams(2020, 2020));
        double sum2020 = this.getSum(waterDemandValues);

        double changeEfficiency = this._changeEfficiencyValue;
        boolean isBigger = false;
        if (sum2020 > this._endWD)
            isBigger = true;

        while ((sum2020 < this._startWD) || (sum2020 > this._endWD)) {
            if (sum2020 < this._startWD) {
                if (isBigger) {
                    changeEfficiency /= 2;
                    isBigger = false;
                }

                efficiencyValue -= changeEfficiency;
            }
            else {
                if (!isBigger) {
                    changeEfficiency /= 2;
                    isBigger = true;
                }

                efficiencyValue += changeEfficiency;
            }

            keyService.setExpression("Efficiency Improvements", "Reference", String.valueOf(efficiencyValu
            e));

            waterService.run(this._projectName);

            waterDemandValues = nodeService.getOutputValues("West City", "Water Demand", "Reference", new
            FilterParams(2020, 2020));
            sum2020 = this.getSum(waterDemandValues);
        }

        return efficiencyValue;
    }

    private double getSum(Interval[] values) {
        double result = 0;
        for (Interval interval : values) {
            for (Data data : interval.getData()) {
                result += data.getValue();
            }
        }
        return result;
    }
}
```

```
public abstract class AbstractWebService {
    private static final String SYSTEM = "Water";
    private String projectName = "";
    private ComponentTypes componentType = null;

    public AbstractWebService() {
    }

    public AbstractWebService(String projectName, ComponentTypes componentType) {
        this.projectName = projectName.replaceAll("\\s", "%20");
        this.componentType = componentType;
    }

    private URI getBaseUrl() {
        UriBuilder uri = UriBuilder.fromUri("http://localhost:8080/" + SYSTEM);
        if (this.componentType != null)
            uri.path(this.projectName).path(this.componentType.getTitle());
        return uri.build();
    }

    protected <T> T Get(Class<T> type) {
        return this.Get(type, "", null);
    }

    protected <T> T Get(Class<T> type, String path) {
        return this.Get(type, path, null);
    }

    protected <T> T Get(Class<T> type, String path, FilterParams filters) {
        path = path.replaceAll("\\s", "%20");
        if (filters != null) {
            path += this.getQueryParameters(filters);
        }
        String url = this.getBaseUrl() + path;
        Builder builder = ClientBuilder.newClient().target(url).request(MediaType.APPLICATION_JSON);
        Response res = builder.get();

        return res.readEntity(type);
    }

    protected boolean Put(String path, Object values) {
        String url = this.getBaseUrl() + path.replaceAll("\\s", "%20");
        Builder builder = ClientBuilder.newClient().target(url).request(MediaType.APPLICATION_JSON);
        Response res = builder.put(Entity.entity(values, MediaType.APPLICATION_JSON), Response.class);

        return res.readEntity(boolean.class);
    }

    private String getQueryParameters(FilterParams filters) {
        String start = "?";
        String result = start;

        if (filters.getStartYear() != 0) {
            if (result == start)
                result += "startYear=" + filters.getStartYear();
            else
                result += "&startYear=" + filters.getStartYear();
        }

        if (filters.getEndYear() != 0) {
            if (result == start)
                result += "endYear=" + filters.getEndYear();
            else
                result += "&endYear=" + filters.getEndYear();
        }

        if (filters.getStartTimeStep() != 0) {
            if (result == start)
                result += "startTimeStep=" + filters.getStartTimeStep();
            else
                result += "&startTimeStep=" + filters.getStartTimeStep();
        }

        if (filters.getEndTimeStep() != 0) {
            if (result == start)
                result += "endTimeStep=" + filters.getEndTimeStep();
            else
                result += "&endTimeStep=" + filters.getEndTimeStep();
        }

        return result;
    }
}
```

```
public class WaterService extends AbstractWebService {
    public WaterService() {
        super();
    }

    public boolean run(String projectName) {
        String path = "/" + projectName + "/Run";
        return (boolean) super.Get(boolean.class, path);
    }
}
```

```
public class KeyService extends AbstractWebService {
    public KeyService(String projectName) {
        super(projectName, ComponentTypes.Key);
    }

    public String getExpression(String compName, String scenarioName) {
        String path = "/" + compName + "/" + scenarioName + "/Expression";
        String temp = super.Get(String.class, path);
        return (String) temp.substring(1, temp.length() - 1);
    }

    public boolean setExpression(String compName, String scenarioName, String expression) {
        String path = "/" + compName + "/" + scenarioName + "/Expression";
        return super.Put(path, new ExpressionValue(expression));
    }
}
```

```
public class NodeService extends AbstractWebService {
    public NodeService(String projectName, ComponentTypes componentType) {
        super(projectName, componentType);
    }

    public Interval[] getOutputValues(String compName, String portName, String scenarioName, FilterParams
    filters) {
        String path = "/" + compName + "/Outputs/" + portName + "/" + scenarioName;
        return (Interval[]) super.Get(Interval[].class, path, filters);
    }
}
```

```
public class Interval {
    private int year;
    private List<Data> data;

    public Interval() {
        this.data = new ArrayList<>();
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public List<Data> getData() {
        return data;
    }

    public void addData(Data data) {
        this.data.add(data);
    }

    public void addData(List<Data> data) {
        this.data.addAll(data);
    }
}
```

```
public class Data {
    private int timeStep;
    private double value;

    public Data() {
    }

    public Data(int timeStep, double value) {
        this.timeStep = timeStep;
        this.value = value;
    }

    public int getTimeStep() {
        return timeStep;
    }

    public double getValue() {
        return value;
    }

    public void setValue(double value) {
        this.value = value;
    }

    public void setTimeStep(int timeStep) {
        this.timeStep = timeStep;
    }
}
```

```
public class FilterParams {
    private int startYear;
    private int endYear;
    private int startTimeStep;
    private int endTimeStep;

    public FilterParams(int startYear, int endYear) {
        this(startYear, endYear, 0, 0);
    }

    public FilterParams(int startYear, int endYear, int startTimeStep, int endTimeStep) {
        this.setStartYear(startYear);
        this.setEndYear(endYear);
        this.setStartTimeStep(startTimeStep);
        this.setEndTimeStep(endTimeStep);
    }

    public int getStartYear() {
        return startYear;
    }

    public void setStartYear(int startYear) {
        this.startYear = startYear;
    }

    public int getEndYear() {
        return endYear;
    }

    public void setEndYear(int endYear) {
        this.endYear = endYear;
    }

    public int getStartTimeStep() {
        return startTimeStep;
    }

    public void setStartTimeStep(int startTimeStep) {
        this.startTimeStep = startTimeStep;
    }

    public int getEndTimeStep() {
        return endTimeStep;
    }

    public void setEndTimeStep(int endTimeStep) {
        this.endTimeStep = endTimeStep;
    }
}
```