Prepared for:

**Federal Communications Commission**

**CMS Alliance to Modernize Healthcare  
Federally Funded Research and Development Center**

Contract No. HHSM-5000-2012-000081

**Task Order No. FCC15D0002**

HashConfig – Automated Configuration Tool Guide

Version 1.0

February 24, 2017

The views, opinions, and/or findings contained in this report are those of The MITRE Corporation and should not be construed as official government position, policy, or decision unless so designated by other documentation.

Approved for Public Release; Distribution Unlimited. Case Number 17-0143.

© 2017, The MITRE Corporation. All Rights Reserved.

Record of Changes

| Version | Date | Author / Owner | Description of Change |
| --- | --- | --- | --- |
| 1.0 | February 24, 2017 | CAMH | Version 1.0 for Sponsor release |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. About the CMS Alliance to Modernize Healthcare

The Centers for Medicare & Medicaid Services (CMS) sponsors the CMS Alliance to Modernize Healthcare (CAMH), the first Federally Funded Research and Development Center (FFRDC) dedicated to strengthening our nation’s healthcare system.

The CAMH FFRDC enables CMS, the Department of Health and Human Services (HHS), and other government entities to access unbiased research, advice, guidance, and analysis to solve complex business, policy, technology, and operational challenges in health mission areas. The FFRDC objectively analyzes long-term health system problems, addresses complex technical questions, and generates creative and cost-effective solutions in strategic areas such as quality of care, new payment models, and business transformation.

Formally established under Federal Acquisition Regulation (FAR) Part 35.017, FFRDCs meet special, long-term research and development needs integral to the mission of the sponsoring agency—work that existing in-house or commercial contractor resources cannot fulfill as effectively. FFRDCs operate in the public interest, free from conflicts of interest, and are managed and/or administered by not-for-profit organizations, universities, or industrial firms as separate operating units.

The CAMH FFRDC applies a combination of large-scale enterprise systems engineering and specialized health subject matter expertise to achieve the strategic objectives of CMS, HHS, and other government organizations charged with health-related missions. As a trusted, not-for-profit adviser, the CAMH FFRDC has access, beyond what is allowed in normal contractual relationships, to government and supplier data, including sensitive and proprietary data, and to employees and government facilities and equipment that support health missions.

CMS conducted a competitive acquisition in 2012 and awarded the CAMH FFRDC contract to The MITRE Corporation (MITRE). MITRE operates the CAMH FFRDC in partnership with CMS and HHS, and maintains a collaborative alliance of partners from nonprofits, academia, and industry. This alliance provides specialized expertise, health capabilities, and innovative solutions to transform delivery of the nation’s healthcare services. Government organizations and other entities have ready access to this network of partners, including RAND Health, the Brookings Institution, and other leading healthcare organizations. This includes select qualified small and disadvantaged business.

The FFRDC is open to all CMS and HHS Operating Divisions and Staff Divisions. In addition, government entities outside of CMS and HHS can use the FFRDC with permission of CMS, CAMH’s primary sponsor.

1. Executive Summary

The Federal Communication Commission’s (FCC) Telecommunications Relay Service (TRS) Center of Expertise (COE) Project promotes the Commission’s goal to foster innovations that advance functionally equivalent telecommunications. Toward that end, the project ensures that the Telecommunications Relay Service employs improved technology for persons who are deaf, hard of hearing, deaf-blind, and/or have speech disabilities. The FCC has embraced a research-based approach to achieve this goal by engaging the Centers for Medicare & Medicaid Services (CMS) Alliance to Modernize Healthcare (CAMH) Federally Funded Research and Development Center (FFRDC), operated by The MITRE Corporation (MITRE), to conduct independent engineering assessments that promote and demonstrate TRS’s functional equivalence.

CAMH is independently assessing voice telephone services, video access services, and Internet Protocol (IP)-based captioning technology; improvements to TRS efficiency; solutions for direct communication between people with communication disabilities and other telephone users; and the effectiveness, efficiency, and consumer response to current and future approaches for delivering TRS.

This document presents an overview of “HashConfig,” an open source, automated configuration tool that can be used to configure CAMH-developed products. This standalone Node.js application automates the process of editing a JavaScript Object Notation (JSON) file. HashConfig supports a more automated approach to creating secure JSON configuration files, and can be extended more generally to any JSON file.

The HashConfig tool reads a template JSON file and extracts all of its terminal field-value pairs. For each field, it then prompts the user to either accept the given default value or to provide a custom value. At the end of the user input process, the software generates a new configuration file with the specified values written as a secure hashed value.

HashConfig can be used to configure two CAMH-developed Accessible Communications for Everyone (ACE) products—ACE Connect Lite and ACE Direct. All of these products are available for download or reproduction by the public at <http://mitre.github.io/>.

Table of Contents

[1. Introduction 1](#_Toc474771021)

[1.1 Background 1](#_Toc474771022)

[1.2 Automated Software Configuration and Deployment 1](#_Toc474771023)

[2. Getting Started 2](#_Toc474771024)

[2.1 Code Structure 2](#_Toc474771025)

[2.2 How to Use HashConfig 2](#_Toc474771026)

[Acronyms 8](#_Toc474771027)

List of Figures

[Figure 1. Using the Command Prompt to Install Node.js Modules 2](#_Toc474771028)

[Figure 2. Running hconfig.js and Entering the Prompted User Input 3](#_Toc474771029)

[Figure 3. Running hconfig.js with the “--force” Command Line Argument 3](#_Toc474771030)

[Figure 4. Running hconfig.js with the “--decode-input” Command Line Argument and   
Entering the Prompted User Input 3](#_Toc474771031)

[Figure 5. Running hconfig.js with the “--decode-output” Command Line Argument and   
Entering the Prompted User Input 4](#_Toc474771032)

[Figure 6. Example Output config\_new.json File with the “--decode-output” Command Line Argument 4](#_Toc474771033)

[Figure 7. Running hconfig.js with the “--help” Command Line Argument 5](#_Toc474771034)

[Figure 8. Example Output config\_new.json File 5](#_Toc474771035)

[Figure 9. Running updateconfig.js with the “--decode-input” Command Line Argument   
and Entering the Prompted User Input 6](#_Toc474771036)

[Figure 10. Running updateconfig.js with the “--help” Command Line Argument 6](#_Toc474771037)

[Figure 11. Example Output config\_updated.json File 7](#_Toc474771038)

# Introduction

The Federal Communications Commission (FCC) Telecommunications Relay Service (TRS) Center of Expertise (COE) Project promotes the Commission’s goal to foster innovations that advance functionally equivalent telecommunications. Toward that end, the project ensures that the Telecommunications Relay Service employs improved technology for persons who are deaf, hard of hearing, deaf-blind, and/or have speech disabilities.

## Background

The FCC has embraced a research-based approach to achieve this goal by engaging the Centers for Medicare & Medicaid Services (CMS) Alliance to Modernize Healthcare (CAMH) Federally Funded Research and Development Center (FFRDC), operated by The MITRE Corporation (MITRE), to conduct independent engineering assessments that promote and demonstrate TRS’s functional equivalence. CAMH independently assesses voice telephone services, video access services, and Internet Protocol (IP)-based captioning technology; improvements to TRS efficiency; solutions for direct communication between people with communication disabilities and other telephone users; and the effectiveness, efficiency, and consumer response to current and future approaches for delivering TRS. To demonstrate the viability of open source solutions to facilitate direct communication between people with communication disabilities and other telephone users, CAMH successfully developed and released two open source products, ACE Connect Lite and ACE Direct, which are available for public download at <https://github.com/FCC/ACEConnectLite> and <https://github.com/FCC/ACEDirect>.

ACE Connect Lite and ACE Direct leverage other open source libraries and products to deliver required functionality. In addition, they require that multiple configuration parameters be properly defined to operate in a specific environment. The installation and configuration of multiple software packages can be a formidable task. To address the configuration challenges, CAMH developed HashConfig, an open source utility to automate and simplify the configuration of the ACE products.

## Automated Software Configuration and Deployment

Automating software deployment involves automating the installation, configuration, and execution of a specific version of software. It requires leveraging tools and writing custom scripts that hide the installation, configuration, and execution details from the installer. Primary steps include downloading a specific release of all the software components from the source code repository, installing the components on potentially different servers, configuring the software, starting all servers, and executing sanity tests to check if the system is running properly.

To facilitate the configuration and deployment of ACE Connect Lite and ACE Direct, CAMH developed an automated configuration utility, HashConfig. CAMH’s design of HashConfig, however, does not limit it to the automated configuration and installation of ACE Connect Lite and ACE Direct. Rather, this open source, standalone Node.js application automates the process of editing a JavaScript Object Notation (JSON) file. CAMH intended this tool as part of a more automated and secure approach to create JSON configuration files and for extension to any JSON file.

HashConfig software reads a template JSON file and extracts all its terminal field-value pairs. For each field, it then prompts the user to either accept the given default value or to provide a custom value. At the end of the user input process, the software generates a new configuration file with the specified values.

HashConfig provides users with a faster, simpler, and more secure way to generate JSON configuration files. By default, it encodes all field values in the new configuration file using simple Base64 encoding for protection of sensitive information. The user may also elect to decode the new configuration file for easier readability when such security measures are not needed. Other available options include automatically accepting all default values, decoding an encoded template file, and using an additional helper software to update or modify the user’s new values.

The HashConfig software and documentation is available for public download at <http://mitre.github.io/>.

# Getting Started

1. Open the parent directory where the HashConfig repository will be placed.
2. Clone this repository and go to the hashconfig folder by running “cd hashconfig” in the command prompt.

Install the required node.js modules by running “npm install” in the command prompt.

Figure 1 is a sreenshot of the command prompt for installing Node.js modules.

Figure . Using the Command Prompt to Install Node.js Modules

## Code Structure

* package.json contains a brief description of the program along with a list of the required modules.
* hconfig.js is the main file, which reads the config.json\_TEMPLATE file, prompts the user to either accept the default configuration values or input custom values, and outputs a configuration file called config\_new.json.
* updateconfig.js is an additional file that can be used for making small changes to the config\_new.json file.

## How to Use HashConfig

1. Open the appropriate local directory in the command prompt.
2. Type “node hconfig.js” in the command line.

Figure 2 is a screenshot of the command prompt for entering the prompted user input.

Figure . Running hconfig.js and Entering the Prompted User Input

1. (Optional) Adding the argument '--force' or '-f' (“node hconfig.js –force”) automatically generates config\_new.json using the values given in config.json\_TEMPLATE.

Figure 3 is a screenshot for the "--force" command line argument.

Figure . Running hconfig.js with the “--force” Command Line Argument

1. (Optional) Adding the argument '--decode-input' or '-i' (“node hconfig.js --decode-input”) decodes an encoded config template file.

Figure 4 shows the screenshot for the 
"--decode-input" command line argument and entering the prompted user input.

Figure . Running hconfig.js with the “--decode-input” Command Line Argument and Entering the Prompted User Input

1. (Optional) Adding the argument '--decode-output' or '-o' (“node hconfig.js --decode-output”) returns an output file that is not encoded.

Figure 5 shows the screenshot for the "--decode-ouput" command line argument and entering the prompted user input.

Figure . Running hconfig.js with the “--decode-output” Command Line Argument and Entering the Prompted User Input

Figure 6 presents a screenshot of the example output with the "--decode-output" command line argument.

Figure . Example Output config\_new.json File with the “--decode-output” Command Line Argument

Follow prompts for user input. The user may also add any combination of the optional arguments. **Note:** If arguments are entered incorrectly, or if the user inputs the '--help' or '-h' argument, help text will appear and the program will exit.

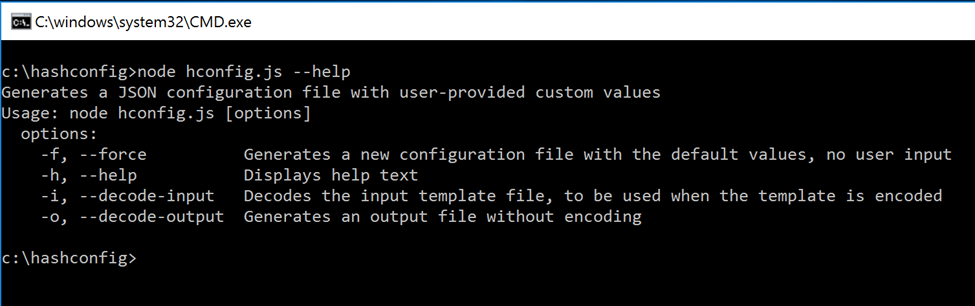


Figure 7. Running hconfig.js with the “--help” Command Line Argument

1. After index.js has finished running, open config\_new.json in the same directory to see the new encoded configuration file.

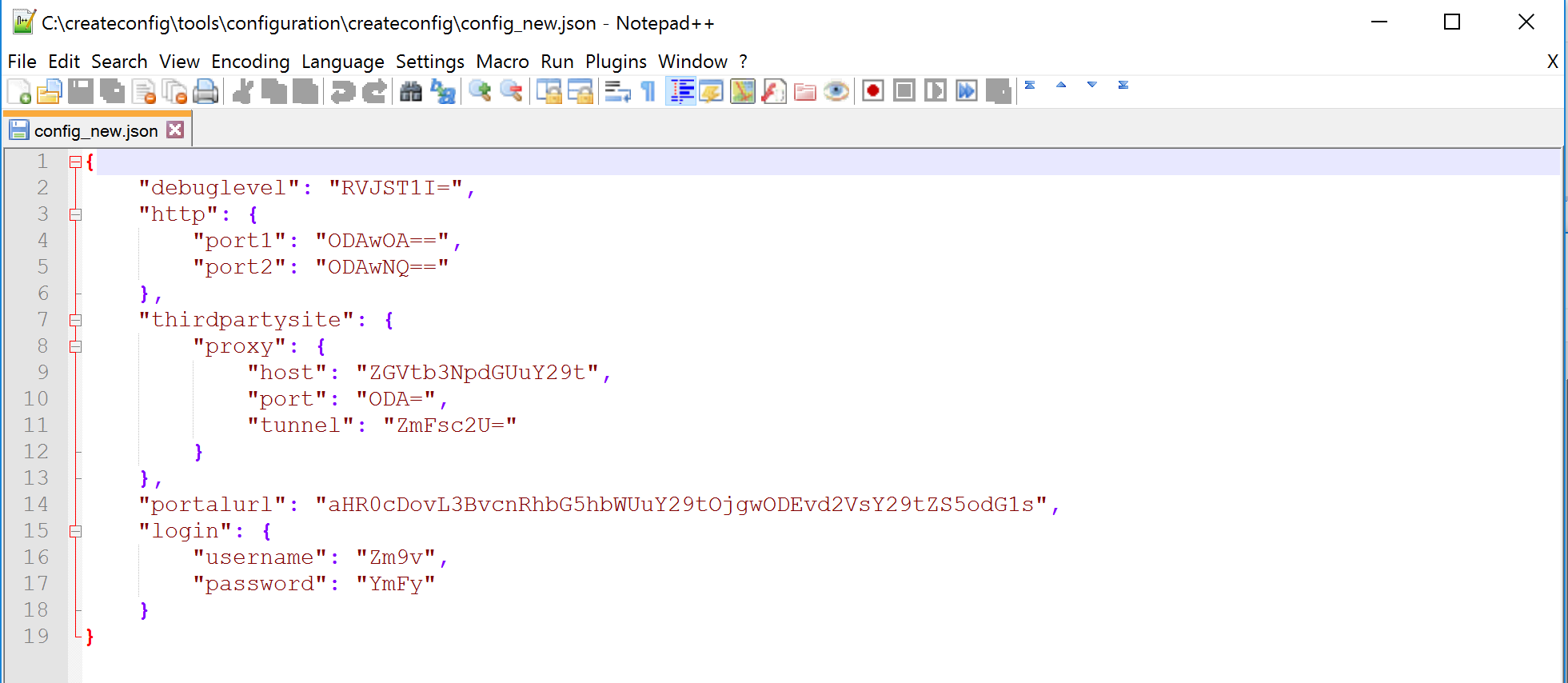


Figure 8. Example Output config\_new.json File

1. (Optional) If a field needs to be changed in config\_new.json, type “node updateconfig.js” in the command line and follow prompts for user input. This program uses the '--decode-input' and '--decode-output' command line arguments in the same manner as hconfig.js. For example, the encoded config\_new.json file created in step 2 can be modified using the '--decode-input' argument.

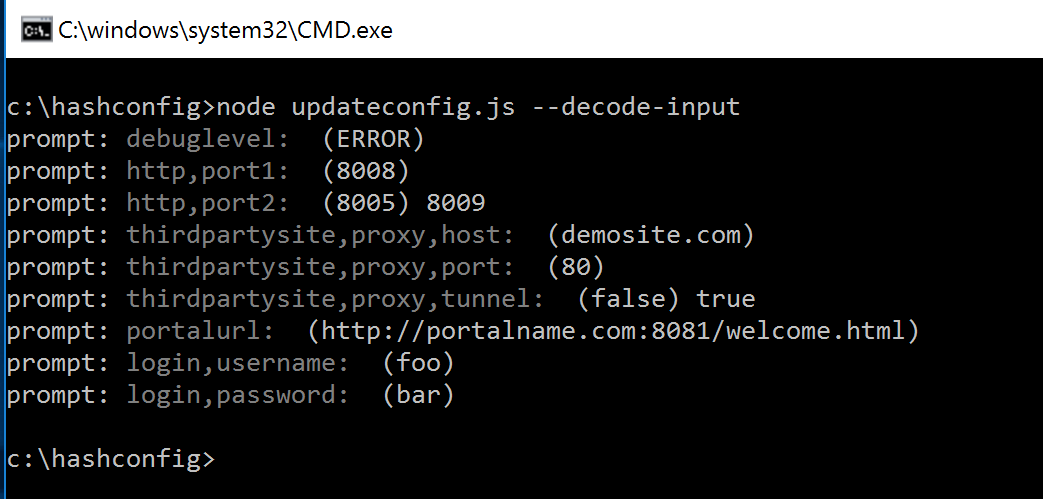


Figure 9. Running updateconfig.js with the “--decode-input” Command Line Argument and Entering the Prompted User Input

1. Follow prompts for user input. **Note:** If arguments are entered incorrectly, or if the user inputs the '--help' or '-h' argument, help text will appear and the program will exit.

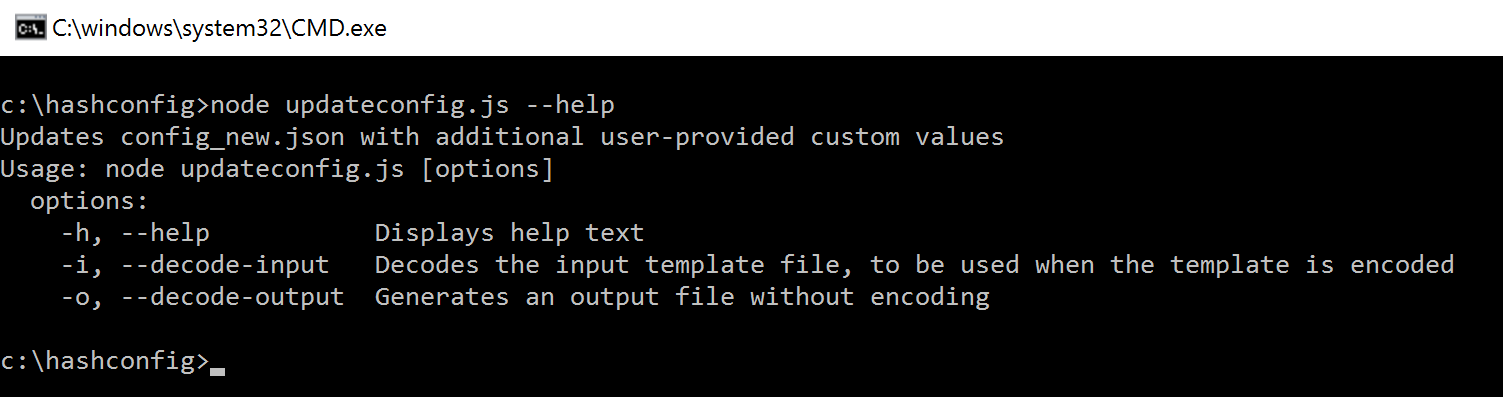


Figure 10. Running updateconfig.js with the “--help” Command Line Argument

1. After updateconfig.js has finished running, open config\_updated.json in the same directory to see the new encoded configuration file.

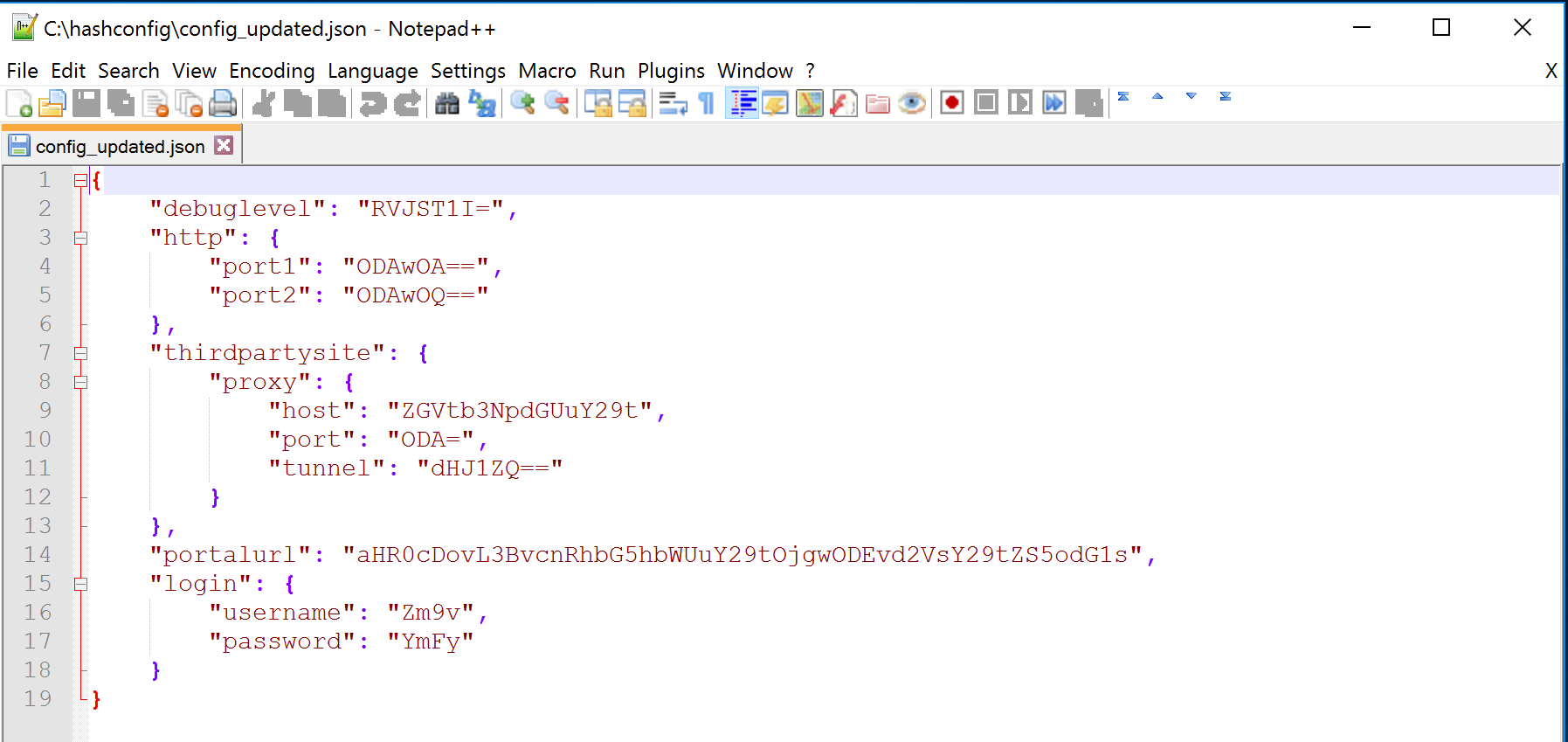


Figure 11. Example Output config\_updated.json File

Acronyms

|  |  |
| --- | --- |
| ACE | Accessible Communications for Everyone |
| CAMH | CMS Alliance to Modernize Healthcare |
| CMS | Centers for Medicare & Medicaid Services |
| COE | Center of Expertise |
| FCC | Federal Communications Commission |
| FFRDC | Federally Funded Research and Development Center |
| HHS | Department of Health and Human Services |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| TRS | Telecommunications Relay Service |
|  |  |