

 Creating RSA Keys for Cloud Based Password Vault for WDG

*Version 0.0*

*Rough Draft*

*Authors:*

*Ryan Sparks –* *rmsparks@us.ibm.com*

*Zach Silverstein -* *zachary.silverstein@ibm.com*

**Table of Contents**

[1. Introduction 3](#_Toc48658858)

[Use case 3](#_Toc48658859)

[Prerequisites 3](#_Toc48658860)

[2. Create the Public and Private Keys 4](#_Toc48658861)

[3. Upload Keys to WDG Automation Web Portal 6](#_Toc48658862)

[4. Create Credential in WDG Web Portal 8](#_Toc48658863)

[5. Access Credential from WDG Studio 9](#_Toc48658864)

# Introduction

This document will detail the steps to create a public and private key using OpenSSL creating a new credential in the web portal, and retrieving the newly created credential using WDG Automation Studio.



Figure . WBG Automation – Automation Studio

## Use case

Like many automation solutions you will need to access and coordinate with other applications and tools. Often times these tools will require some kind of credential to be used whether that be usernames, passwords, api keys, and any other sensitive data. It is best practice to avoid placing these credentials in scripts or places that can be exposed to unintended audiences. Using the cloud based vault that comes with WDG we can securely store and access these credentials through scripting to be used during automation.

## Prerequisites

* Personal WDG Automation Account activated
* Access to WDG Automation Web Portal
* Access to WDG Automation Studio
* Admin access in WDG Automation Web Portal
* OpenSSL installed on your computer

# Create the Public and Private Keys

Run OpenSSL, navigate to the target folder in which you want to store the key, and type the following command to create the private key. The name of the key is specified by the -out parameter.

*openssl genrsa -out privkey.pem 4096*





You will notice that a private key has been created and placed in the folder path specified by the command.

Next we will create the public key using the private key as input. Run the following command to create the public key. The name of the key is specified by the -out parameter.

*openssl rsa -in privkey.pem -out rsa.public -pubout -outform PEM*





For WDG to use the keys they need to be the correct file type. WDG supports the following encoding types for RSA keys.

Public: .pem, .xml, .der, .crt, and .cer

Private: Certificate, PEM, PKCS#12, and XML

We need to change the file type of the public key to a valid filetype. We will change it .pem as shown below.



# Upload Keys to WDG Automation Web Portal

Navigate to the WDG web portal and sign in using your credentials. Navigate to the Tenant Configuration by selecting the dropdown in the top left corner.



Select the Credential tab.



Upload the public and private keys that were created in the previous step. You will need to specify the directory location of the private key that exists on the host machine.

\*\*The public key **IS** shared by the Tenant and only needs to be uploaded a single time.

\*\*The private key is **NOT** is not shared across computers in the same tenant and will need to exist on every machine that intends to run bots that will access the cloud based password vault.



# Create Credential in WDG Web Portal

Create a credential in the WDG web portal. Navigate to the Credential section on the left had side.



Select “New Credential”

You will now enter the information for the credential you wish to create

**Name**: Name used to identify the credential.

**Username**: Username portion of the credential.

**Password**: Password portion of the credential.



After selecting “Save” it will create the new credential. We can now see it as an entry in the credential list.



# Access Credential from WDG Studio

We will create a simple script that will access and retrieve a credential from the cloud based password vault that was created in the previous step.

Open WDG Automation Studio and create a new .wal file.



On the left had side select the “Get Vault Item” action. We will update the parameters to pull the recently created credential.

Name: The name of the credential to be retrieved. In our case it is “Test Credential.”

System: This determines which vault to access. When toggled to true it will use the cloud based vault.

Success: Output variable used to determine retrieval success.

User Name: Username portion of the credential.

Password: Password portion of the credential.



We will now add logging to view the success and username variable output from the retrieval.

On the left had side select the “Log Message” action. Place the following in the message field

“Success: ${sucess}

UserName: ${userName}”

Because the password is a secure string we will not be able to log and view its value. It can be stored in a variable and used later in the script when the credential is needed.



Select “Save” and run the script.



Confirm that the logs show the credential retrieval is successful.

