C:\Users\STEVEH~1\AppData\Local\Temp\SNAGHTML2368fa01.PNG

Commvault Email Pylint (CVEmailPylint)

A Commvault Engineering Design Document



CONTENTS

[Introduction 4](#_Toc464046863)

[User Benefits 4](#_Toc464046876)

[Concepts and Terminology 4](#_Toc464046864)

[Product/System Requirements 5](#_Toc464046865)

[Assumptions 5](#_Toc464046866)

[Architecture and High Level Design 5](#_Toc464046867)

[Coding Considerations 6](#_Toc464046868)

[UI Screens and User Experience 6](#_Toc464046869)

[Install Changes 6](#_Toc464046870)

[Upgrade and Backward compatibility 6](#_Toc464046871)

[Changes to Other Subsystems 6](#_Toc464046872)

[Product Support 6](#_Toc464046873)

[Phases of Project 7](#_Toc464046874)

[Open Issues 7](#_Toc464046875)

[Conclusions 7](#_Toc464046876)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Authors | Comments |
| 06/28/2018 | 0.1 | Komal Dawkhar | Initial draft |
|  |  |  |  |
|  |  |  |  |

Design Reviews

* The people who attended the design review.
* The date of the review.
* Any review notes.

# Introduction

CVEmailPylint Package has designed to send Pylint report of python files in update form through email. Pylint Report contains Pylint Score, Convention, Refactor, Warning, Error count and Errors in the form of table.

In Pylint Email Package we have created a folder name Pylint on [\devshare\devl\CoreAutomation\Pylint\](file:///\\devshare\devl\CoreAutomation\Pylint\\) this path.

When we run the CVEmailPylint package by giving arguments as formid, buildid, mountpath through command line arguments, then initially it searches all files in that form. After that, it validates the directories structure like mount path appending vaultcx\Source\tools\Automation\ and vaultcx\Source\tools\ cvpysdk\ and checks file is only python file or not.

On those python files only it runs pylint and stores that pylint output in text file on [\devshare\devl\CoreAutomation\Pylint\](file:///\\devshare\devl\CoreAutomation\Pylint\) folder with separate folder containing folder name as form id.

By connecting to UpdateCenter database we are accessing no. of files in update form and accessing forms Dev Owner, Developer, Additional developer, Code Reviewers (Developer Choice & System Enforced) to send that Pylint Report over email.

There is batch file name PylintBatch.bat to run this package with arguments.

Run CVEmailPylint as:

PylintBatch.bat -formid 50888 -buildid 1100080 -mountpath F:\PC\test-mount

Pylint Clean Up Script

This clean up script is for to clean up all the folders which have created on \devshare\devl\CoreAutomation\Pylint\ folder same name as form id name. When the form gets in official state then to avoid extra memory space usage we are removing those folders here after specific time automatically.

# User Benefits

Benefits to Users:

* The User can easily get whatever issues in python code files.
* The User can easily check the Pylint Score, Convention, Refactor, Error count in the file through email
* The user can able to see whatever the errors through email itself.
* In this way it would be easy to find and fix the errors in file.
* Any OS with Python running with above Python 3.6 version can use this Pylint feature.

# 

# Concepts and Terminology

* Pylint
* Python 3.6

# 

# Product/System Requirements

Requires Python 3.6 version installed on the system to work with the following packages:

As Pylint supports the version above python 3 .

* “**Pylint**” Python package
* “**Pip**” Python package

# Assumptions

* User should be acquired with basic Python, Pip, Pylint knowledge.
* User should have basic understanding of Update Forms

# Architecture and High Level Design

* Design level:
* Divide the EmailPylint package into two files as uc\_helper.py and cvemail\_pylint.py for easier operability as well as understandability, and code maintainability.
* uc\_helper.py file is the main file. The object of uc\_helper.py calls execute () method which

again creates object of cvemail\_pylint.py class and calls it’s execute () method again.

* Separate file for **cvemail\_pylint.py** :

In this execute () is a main method which contains all methods inside.

* initialize\_logger() - Method which initializes logger.
* run\_pylint() - Method runs the pylint on given python files and stores it in a list.
* pylint\_text() - Method creates text file and stores pylint output in it. Text files get stored

on \ devshare\ CoreAutomation\Pylint\ folder in separate folder name with

same name as update form id.

* store\_pylint() - Method which determines the pylint score from the output and generates the

html message out of it.

* mail\_pylint() - Method which sends Pylint report through email
* Separate file for **uc\_helper.py :**
* In this execute() method is a main method which contains all the methods inside and here we have created the object of cvemail\_pylint class also which calls it’s execute() method from CvemailPylint class.
* initialize\_logger() - Method which initializes logger.
* read\_args() - reads the arguments from command line as formid, buildid and mountpath. Where

formid – update form id for which pylint score is to be determined.

buildid – build id reference for which the form is being build.

mountpath – path on the build machine where the source files specified in the form

are mounted.

* get\_files\_list() – method to get all the files from given update form by using database connection
* validate\_files() - Filter out the python files from the source files list which are placed under

Automation and cvpysdk directory.

* email\_receiver() - Method iterates over all stake holders of the form and determine the users list to

whom the email is to be sent.

* email\_receivers\_alias() – method determines the full alias of email id’s of all users whom the email is

to be sent.

* generate\_json() – generates dictionary containing files list in update form and email content.

* send\_notification\_email() – Method to send notification email if code fails.
* Separate file for Clean up script as **pylint\_cleanup\_script.py :**

In this execute() method is a main method which contains all methods inside as:

* check\_path() – Method checks the share file path on devshare
* check\_form\_state() – Method checks the forms state as official by using database connection.
* cleanup\_script() – Method which clean ups the folders on given share path after specific time

# Coding Considerations

* Used Composition over Inheritance for cleaner and easier maintainable code
* Proper documentation of all the modules
* Doc Strings for all classes and its methods
* Code compliant with PEP8 style guide

# UI Screens and User Experience

* No changes are required here

# Install Changes

* Python 3.6 zip
* Pip
* Pylint

# Upgrade and Backward compatibility

* Will this feature work on upgraded setup?
  + This feature will work on any setup with Python 3, Pylint installed
* Is this feature backward compatible?
  + This feature is compatible with Python 3 or any later release
  + This feature requires Pylint installation

Flow Chart:

CVEmailPylint

Start

object creation of uc\_helper.py file

Object calls execute method which contains all methods inside

execute() calls initialize\_logger() method which starts logger for uc\_helper class

After it calls read\_args() which reads arguments from command line as formid, buildid & mountpath

Later execute() calls get\_files\_list() which accesses all files in the form by using DB connection

Later it calls validate\_files() which validates python files in directory

Later it calls email\_receiver() which accesses devowner, dev, add dev, code reviewers(Dev choice & System enforced) to send Pylint email

Later to get full email alias it calls email\_receivers\_alias() method

It creates object of cvemail\_pylint class which calls execute() of cvemail\_pylint.py file

execute() calls initialize\_logger() of class to start logger

It calls run\_pylint() which runs pylint on python files

After it calls pylint\_text() which creates text file to store pylint o/p in it on \devshare\CoreAutomation\Pylint\ folder same name as form id no

It calls store\_pylint() which determines pylint report from the o/p and generates reports in format of table

After it calls mail\_pylint() which sends pylint report in html table format over email

END

Pylint CleanUp Script:

Start

Object creation of pylint\_cleanup\_script.py file

Object calls execute method which contains all methods inside

Execute calls check\_path method first which checks the path on devshare

Check form’s state i.e. check first forms state

if state is official

No

Yes

Call clean\_up() which cleans all folders which are official

End

# Changes to Other Subsystems

* Integrated Python 3.6 zip file to <https://cvs.commvault.com/source/xref/11.0-cte/CTELibraries/Libraries/python/win/> for Pylint support

# Phases of Project

* If this project will be done in multiple phases, include those details here.

# Open Issues

* List any parts of the design that were not addressed and remain open.

# Conclusions

* Any concluding remarks.

**Visit the** [**Commvault Documentation**](http://documentation.commvault.com/commvault/v10/relatedLinks?action=knowledgebase) **web site for complete documentation of Commvault products.**

©2016 Commvault Systems, Inc. All rights reserved. Commvault, Commvault and logo, the “CV” logo, Commvault Systems, Solving Forward, SIM, Singular Information Management, Simpana, Simpana OnePass, Commvault Galaxy, Unified Data Management, QiNetix, Quick Recovery, QR, CommNet, GridStor, Vault Tracker, Innervault, QuickSnap, QSnap, Recovery Director, CommServe, CommCell, IntelliSnap, ROMS, Commvault Edge, and Commvalue, are trademarks or registered trademarks of Commvault Systems, Inc. All other third party brands, products, service names, trademarks, or registered service marks are the property of and used to identify the products or services of their respective owners. All specifications are subject to change without notice.