Code Analysis Bot (CAB)

Readme

Version 1.0

16/12/2019

Table of Contents

1. Introduction

The Code Analyzer Bot (referred hereafter as “CAB”) is an Attended Bot that scans Automation Anywhere TaskBot source code against bot development best practices, and provides a report on potential issues.

This manual includes a description of the capabilities, and step-by-step procedures for setup & configuration of the Bot.

1.1 Overview

Automation Anywhere’s approach of “freeing the humans” can be applied to any process. The Code Review process is a critical component of any Bot Development Lifecycle, and CAB can assist by reducing the amount of manual effort to analyze code that might require correction.

Note that the author of this solution believes that Code Reviews are best viewed as a cooperation between Digital and Human Workers. Each plays a distinct role to execute the process most effectively and efficiently:

- **Digital Worker** – compare source code to known Best Practices, and highlight line items that require rework or further review by a human
- **Human (ex. Solution Architect/Development Lead)** – review the code where more complex analysis is required, such as the over-all structure of the solution, use of TaskBots vs. MetaBots, or where judgement is required.

By reducing the time/manual effort to analyze code, CAB is able to help reduce the over-all cycle time to automate a process and shorten the “time to value” – ie. the automation deployed to Production and delivering business value.

1.2 Input

After installing the package from the Bot Store, you should find sample input/output in the My Tasks\Bot Store\CodeAnalysisBot-AutomationAnywhere\Sample Input and Output folder.

For sample input, a file **CAB_Sample_ProcessFile.atmx** has been provided. Note that this TaskBot is *not* intended to be runnable, but to illustrate the types of issues CAB will find in source code.

Similarly, a file **CodeReview Template - CAB - Sample.xlsx** is provided, that shows the corresponding output to the sample input file.

Input to CAB is the following:

- one or more ATMX or TXT files consisting of Taskbot source code
- a config.XML file,
- an EventHandling.xml file
- an Excel workbook template

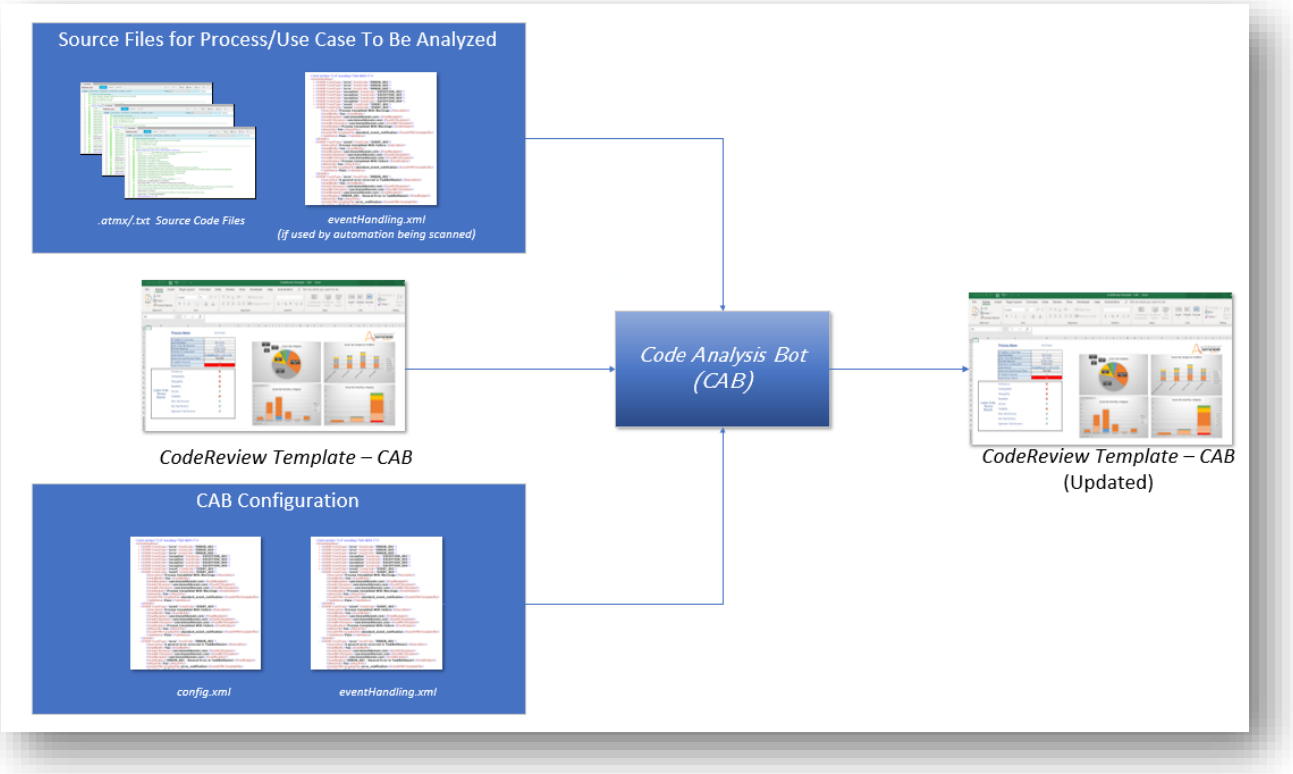
The following are important considerations regarding the choice of providing source code in .atmx vs. .txt format:

Factor For Consideration	Description	Preferred Format
Convenience	.atmx files can be directly provided to CAB; for .txt, each TaskBot must be opened in the AA Client, 'Save as Text', and .txt files provided to CAB	.atmx
Accuracy/Output	.atmx files may not always produce accurate line number references for issues found. Note: this is not a significant issue in reality, given the 'output' of CAB also indicates content of the line/command where the issue was found. In addition, in rare cases, some code from .atmx cannot be read and is therefore not available for analysis	.txt
Validation Scope	Some validations are only possible for specific formats. Ex. • Validation of unused variables – only available in .atmx • Validation of unused/disabled source code – only available in .atmx	Personal Preference (see recommendations below)

	• Validation for specific Error handling – only available in .txt	

1.3 Output

Output is an Excel workbook, detailing the results of the review.



1.3.1 CodeReview Template – Overview

The following worksheets are part of the CodeReview Template:

1. **QA Checklist** – this checklist represents a 'Best Practice' from Automation Anywhere for items that should be checked in a Code Review. In many cases the check can be performed by CAB – these items are indicated by column F as "Automated via CAB". In other cases, the check needs to be performed manually at this time, by a Solution Architect/Lead Developer who will have ultimate accountability for code quality prior to promotion to Production.

								Initial/Peer Review DD/MM/YY				Final Review (Pre-Production) DD/MM/YY			
Validation Ref #	Primary Category	Primary Sub Category	Checks	Automated or Manual Check?	Severity	Issue Tag	TaskNames	Instances	Action Needed	Open Actions?	TaskNames	Instances	Action Needed	Open Actions?
1.01	Architecture	Commands Hierarchy	Is hierarchy of Recorders/Commands followed? Recorders: Smart Recorder >> Web Recorder >> Screen Recorder Commands: Object Cloning >> Manage Web/Window Control >> Keystrokes >> etc.	Manual check	Medium	-	abc.atmx , xyz.atmx	2		No				
1.02	Architecture	Design	Is code following Process-To-Be/Target-Process-Path defined in Technical / Solution Design Document?	Manual check	High	-								
1.03	Architecture	Duplicate code	For any duplicate lines of code, can you modularize (Sub-Task) or place it in a Loop?	Manual check	Medium	-								
1.04	Architecture	Re-run logic	Is Re-run logic set for Task if it fails to run, so that it can pickup from the Transaction that failed? (if applicable)	Manual check	Medium	-								
1.05	Architecture	Scalable	Is Project scalable? (Proper Metabot usage, reusable Sub-Tasks for other Projects, etc.)	Manual check	High	-								
1.06	Architecture	Task	Is the code as modular as possible? Function wise separation of Tasks to increase reusability.	Manual check	Medium	-								
1.07a	Architecture	Task	Do any modules (ex. TaskBot) exceed 500 lines? Recommendation is not to exceed 500 lines, creating sub-tasks instead to improve readability/maintenance.	Automated via CAB	Medium	TaskBot exceeds 500 lines of code								
1.07b	Architecture	Task	No TaskBot should exceed 750 lines.	Automated via CAB	High	TaskBot exceeds 750 lines of code								
1.07c	Architecture	Task	No TaskBot should be less than 50 lines. Typically a separate TaskBot would not be required for this small amount of code.	Automated via CAB	Medium	TaskBot has too little code								
2.01	Configurability	Commands	Are Commands like Mouse Clicks or Image Recognition used only if no preferred Commands (Object Cloning, Manage Web/Window Control, Keystrokes) helped?	Automated via CAB	Low	Non-optimal commands found - alternatives?								
2.02	Configurability	Config file	External configuration file present (Recommendation is to use XML format,	Manual check	Medium	-								
2.03	Configurability	Config file	• File contains support email(s)	Manual check	Medium	-								
2.04	Configurability	Config file	• Database connection string (if applicable)	Manual check	Medium	-								
2.05	Configurability	Config file	• File paths (if applicable)	Manual check	Medium	-								
2.06	Configurability	Config file	• Website URLs (if applicable)	Manual check	Medium	-								
2.07	Configurability	Config file	• Any initialization variables (if applicable)	Manual check	Medium	-								
2.08	Configurability	Delay	Are the values for the Delay commands configurable (ex. vDelayInSecondsShort, vDelayInSecondsMed, vDelayInSecondsLong)	Automated via CAB	Medium	Hard-coded Delay Found								
2.09	Configurability	Dependencies	Have all the Task dependencies been identified and verified before uploading the Bot to Control Room? (e.g. Mapper file, Config file, Template	Manual check	Medium	-								
2.10	Configurability	Dependencies	Are the dependencies from "My Docs", "My Scripts", etc. added to Bot Dependencies Pane and referenced using $AAApplicationPath$ system variable?	Manual check	Medium	-								

Figure 1 - Sample - Detailed Results from Analysis

Columns in this worksheet are as follows:

a. **Validation Ref #** - a unique # representing the specific validation rule.
b. **Primary Category** – a category for the validation rule, for use in reporting (see 'Summary' worksheet for sample charts)
c. **Primary Sub Category**– a sub-category for the validation rule, for use in reporting (see 'Summary' worksheet for sample charts)
d. **Secondary Category** – (not used at this time) – for future support of a secondary 'hierarchy' for reporting issues
e. **Checks** – description of the validation
f. **Automated or Manual** – indicates if the Code Analysis Bot (CAB) supports the validation, or it currently must be executed manually
g. **Severity** – a rating of low, medium, high. See Interpreting Results section for explanation of severity levels.
h. **Issue Tag** – the text that will be used in the code review worksheet, when the issue is found in the source code
i. **"Initial/ Peer Review"** – this section of columns can be used by any feedback from those performing an initial/peer review of the source code

j. "**Final Review"** - this section of columns can be used by any feedback from those performing a final review/approval of the source code

2. Summary -

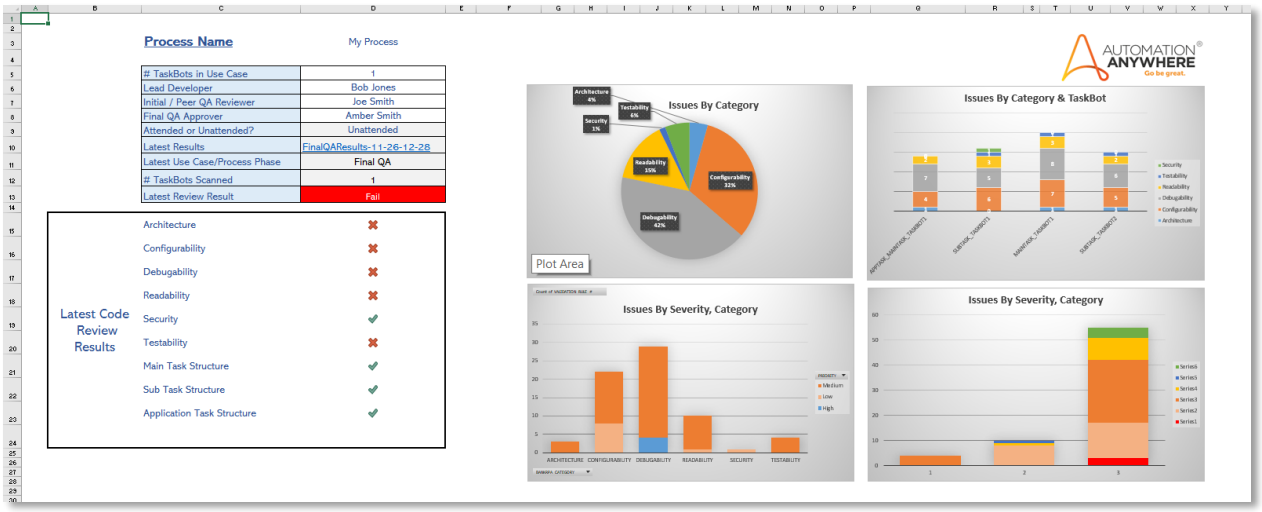


Figure 2- Sample - Summary Results from Analysis

Key information in this worksheet:

<u>The following should be populated manually before running CAB on source code</u>:

- **Process Name** – name of the process that has been automated
- **# TaskBots in the Use Case** – how many TaskBots in total are part of the use case/process?
- **Lead Developer** – list the name of the person in this role
- **Initial / Peer QA Reviewer** – list the name of the person in this role
- **Final QA Approver** - list the name of the person in this role

<u>The following is automatically populated by CAB during execution:</u>

- **Attended or Unattended?** – this information is relevant because some validations are only relevant for un-attended bots. Ex. Existence of MessageBox or Prompt
- **Latest Results** – name of the worksheet with results from the latest analysis
- **Latest Use Case/Process Phase** – name of the phase of the Use Case that was indicated during the last execution of the code analysis
- **# TaskBots Scanned** – total # of files that were scanned in the latest analysis

- **Latest Review Results** – Pass/Fail. A 'Fail' is indicated if there are any outstanding 'Medium' or 'High' severity findings.

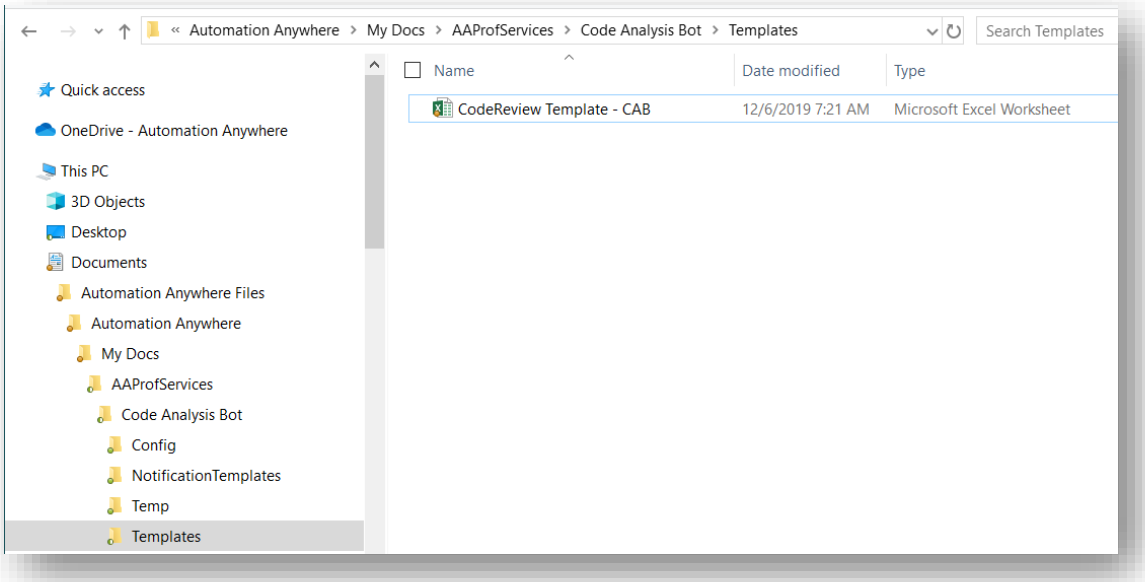
The following section gives a visual indicator of an 'X' by each categories with at least (1) finding.

1.3.2 Protecting the Structure/Content of CodeReview Template

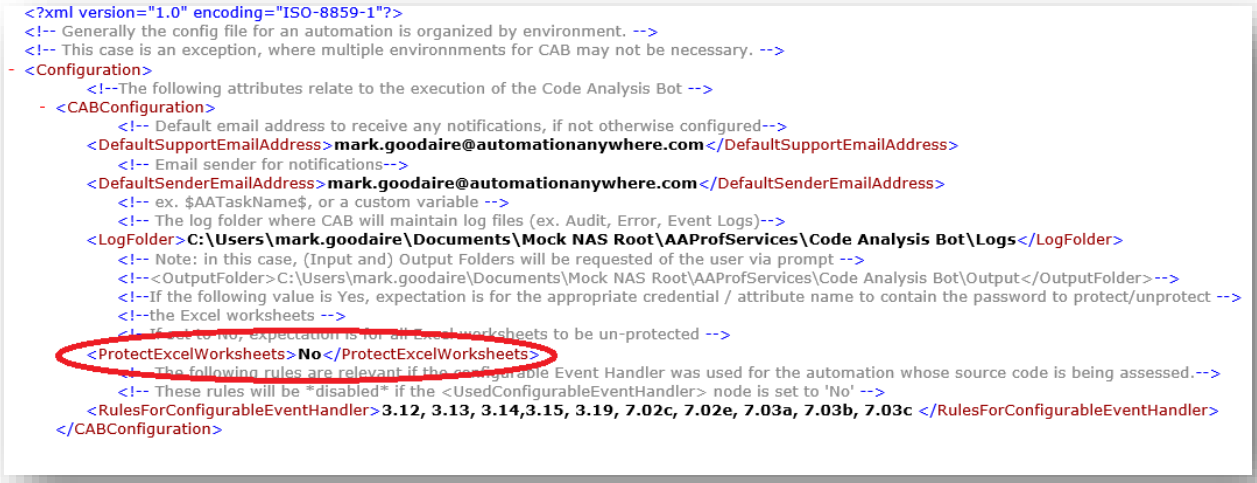
It is recommended that worksheets be locked/protected in Excel, for the purpose of protecting the integrity of the worksheet structure (ie. preventing deletion of rows/columns).

By default, worksheets are *not* protected. To enable protection, follow these steps:

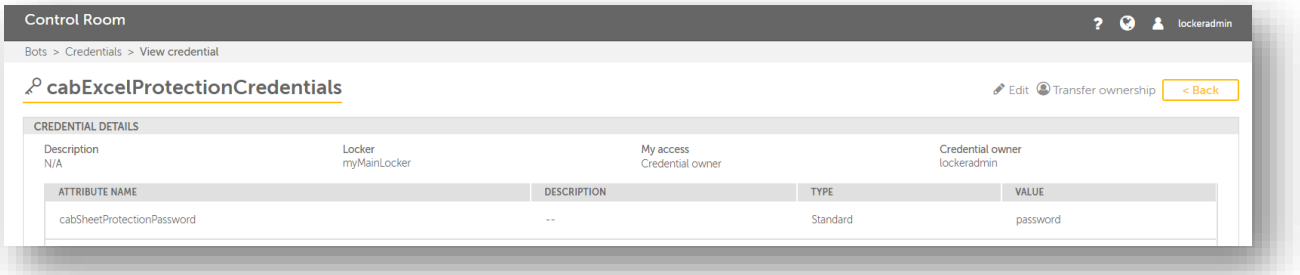
1. In the 'Templates' folder for CAB, lock the 'Summary' and 'Code Analyzer' worksheets with a password – note that the 'Code Analyzer' worksheet is typically hidden by default so you may have to first 'un-hide' it. In the latest version of Excel you can protect the sheets by right-clicking in Excel on the worksheet name, choosing 'Protect Sheet', and entering a password. Save the workbook back to the 'Templates' folder.



2. Update config.XML file to indicate 'Yes' for the 'ProtectExcelWorksheets' node.



3. In your Automation Anywhere Control Room to which CAB will connect, set up credential $cabExcelProtectionCredentials, and attribute $cabSheetProtectionPassword$. Ensure the Bot Runner that will analyze code has access to the appropriate locker/credential.



4. In your ‘Output’ folder, move or delete any existing Excel templates.
    a. Note: If no template is found in the ‘Output’ folder when CAB runs to analyze code, it will copy the template from the ‘Templates’ folder.

Now, run CAB again to analyze source code – the Excel worksheets should be successfully unlocked using the credentials from the credential vault, and re-protected/locked when CAB completes processing.

1.4 Supported Validations

The following set of validations are supported by CAB. Note that additional validations will be added in future as part of continuous improvement.

Validation Ref #	Primary Category	Primary Sub Category	Checks	Required Source Code Format	Comments
1.07a	Architecture	Task	Do any modules (ex. TaskBot) exceed 500 lines? Recommendation is not to exceed 500 lines, creating sub-tasks instead to improve readability/maintenance.	.atmx or .txt	Configurable threshold min/max for lines count
1.07b	Architecture	Task	No TaskBot should exceed 750 lines.	.atmx or .txt	Configurable threshold min/max for lines count
1.07c	Architecture	Task	No TaskBot should be less than 50 lines. Typically a separate TaskBot would not be required for this small amount of code.	.atmx or .txt	Configurable threshold min/max for lines count
2.01	Configurability	Commands	Are Commands like Mouse Clicks or Image Recognition used only if no preferred Commands (Object Cloning, Manage Web/Window Control, Keystrokes) helped?	.atmx or .txt	Any
2.08	Configurability	Delay	Are the values for the Delay commands configurable (ex. vDelayInSecondsShort, vDelayInSecondsMed, vDelayInSecondsLong)	.atmx or .txt	Any
2.23	Configurability	Portability	Are all "Run Task", "Run Script", etc. Commands referring to files hosted in AA Folders using $AAApplicationPath$ for portability? Do those referenced Task/Script files actually exist?	.atmx or .txt	Any
2.26	Configurability	Session	Are Session Names unique and meaningful? Do not use "Default" name.	.atmx or .txt	Any
2.30	Configurability	Variables	Have all variables been created and named as per the required standard format? (refer to naming-convention standards)	.atmx or .txt	Configurable variable naming standards
2.32	Configurability	Variables	Have variables been used for numerical values, vs. hard-coding?	.atmx or .txt	Any
2.39	Configurability	Variables	Any use of Automation Anywhere 'default' variables (ex. Prompt-Assignment) should be minimized, in favor of variables with meaningful names.	.atmx or .txt	Any

2.41	Configurability	Variables	Are variables used to reference URLs, vs. static references?	.atmx or .txt	Any
2.42a	Configurability	Variables	Are variables in the Variable Manager being used?	**.atmx ONLY**	Any
2.42b	Configurability	Variables	Are variables in the Variable Manager being un-necessarily passed to Sub-Tasks?	**.atmx ONLY**	Any
2.35	Configurability	Window title	Are Commands handling Window Titles properly? (Use wildcards "*" if part of Title is non-static)	.atmx or .txt	Any
2.40	Configurability	Window title	Are variables used to reference Window Titles (if not using 'currently active window')	.atmx or .txt	Any
2.38b	Configurability	Windows	Are Window Close commands included in a Loop to ensure they are closed?	.atmx or .txt	Any
3.01	Debugability	Audit Log	Is Audit Log generated to track the flow of task and better reference of runs? Are events such as Start Timestamp, End Timestamp, # of total input Transactions, # of Transactions succeeded, # of Transactions failed, etc. captured?	.atmx or .txt	Configurable audit log naming
3.11	Debugability	Error handling	Does Task have every Command scoped under Error handling?	.atmx or .txt	Any
3.12	Debugability	Error handling	For all the list of events (errors via Error Handling, Exceptions, Events) documented, is code handling them using Event Handler?	.atmx or .txt	Any
3.13	Debugability	Error handling	Are all Event Codes named per standard based on Event Type (Error vs. Exception vs. Event)?	.atmx or .txt	ONLY if Configurable Event Handler used
3.14	Debugability	Error handling	For all the events (errors via Error Handling, Exceptions, Events) listed in the code, are they defined in the Event Handler XML?	.atmx or .txt	ONLY if Configurable Event Handler used
3.15	Debugability	Error handling	For all the Event Codes defined in the Event Handler XML, are they also utilized in the code?	.atmx or .txt	ONLY if Configurable Event Handler used
3.17a	Debugability	Error Log	Has Error Log been maintained for capturing details of the Error?	.atmx or .txt	Any
3.17b	Debugability	Error Log	Does the Error Log capture details of the Error (Timestamp, AA Task Name, Error Line Number and Error Description)?	.atmx or .txt	Any
3.17c	Debugability	Error Log	Are snapshots captured of the error?	.atmx or .txt	Any
3.22a	Debugability	Error Log	In Begin Error Handing, if using Stop Task, is information logged to ErrorLog and a variable set to return to calling Task?	.atmx or .txt	Any
3.22b	Debugability	Error Log	In Begin Error Handing, if using Continue Task, is a variable set that can be checked after End Error Handling?	.atmx or .txt	Any
3.22c	Debugability	Error Log	After Error Handing, if using Continue Task in Begin Error Handling, is a variable checked to see if an error occurred?	**.txt ONLY**	Any

3.18	Debugability	Event handling / notification	Has Event Log been maintained for capturing details on Errors/Exceptions/Events, that Event Handler can process to action events?	.atmx or .txt	ONLY if Configurable Event Handler used
3.19	Debugability	Event handling / notification	Is the XML-configurable Event Handler being called in all TaskBots to managed all Events / Exceptions / Errors raised during processing?	.atmx or .txt	ONLY if Configurable Event Handler used
3.21a	Debugability	Event handling / notification	Is the return indicator after a Sub-Task (ie. tpo see if an error occurred (ex. $xIsError$)?	.atmx or .txt	Configurable 'return variable' names
3.21b	Debugability	Event handling / notification	Is the return indicator after running Metabot Logic (ie. Run Logic) being checked to see if an error occurred (ex. $vOutput$)?	.atmx or .txt	Configurable 'return variable' names
3.21c	Debugability	Event handling / notification	Is the return indicator after script execution (ie. Run Script) being checked to see if an error occurred (ex. $vOutput$)?	.atmx or .txt	Configurable 'return variable' names
5.03	Readability	Comments	Have 'Default Comments' added by Automation Anywhere been removed/replaced with meaningful comments?	.atmx or .txt	Any
5.05a	Readability	Comments	Over-all, is there sufficient commenting to describe the behaviour of the TaskBot? (Recommendation: at least 25% of lines are comments)	.atmx or .txt	Any
5.05b	Readability	Comments	Source code for every TaskBot *must* have at least 15% of lines that are comments.	.atmx or .txt	Configurable minimum % of comment lines
5.06	Readability	Naming Convention	All Tasks should use naming convention to indicate whether Main Task, Sub-Task, Application Task	.atmx or .txt	Configurable TaskBot Types/Naming
6.02	Security	Personal Info	No personal Email ID (gmail/yahoo/etc) configured in commands such as Event handling, Send Mail?	.atmx or .txt	Any
6.03	Security	Personal Info	All PII in automation commands and variables removed	.atmx or .txt	Configurable PII patterns
6.05	Security	Personal Info	No PII stored in logs	.atmx or .txt	Configurable PII patterns
6.06	Security	Personal Info	No PII shown in Message Boxes	.atmx or .txt	Configurable PII patterns
7.02c	Task Structure	Main Task Structure	No business exceptions thrown	.atmx or .txt	ONLY if Configurable Event Handler used
7.02e	Task Structure	Main Task Structure	At least (1) ERROR Event thrown.	.atmx or .txt	ONLY if Configurable Event Handler used
7.03b	Task Structure	Sub Task Structure	At least (1) EXCEPTION Event thrown.	.atmx or .txt	ONLY if Configurable Event Handler used
7.03c	Task Structure	Sub Task Structure	At least (1) ERROR Event thrown.	.atmx or .txt	ONLY if Configurable

					Event Handler used
8.01	Testability	Message Box and Prompt	Is the Task testable? (For example, 'Un-attended' Bot code should not contain any "Message Boxes", "Prompts" etc..)	.atmx or .txt	Any
4.03	Testability	Task run	Does the code properly execute 'desktop clean-up' to kill applicable applications, before and after execution?	.atmx or .txt	Configurable name of 'cleanup' Bot
9.97	Testability	Task run	Is the path name using $AAApplicationPath$, but with an excessive character length?	.atmx or .txt	Any
9.99	Testability	Task run	Missing TaskBot or Script referenced in Run Task/Run Script	.atmx or .txt	Must execute CAB on Bot Runner where Bot being analyzed will be deployed

2. Requirements & Prerequisites

2.1 System Requirements

There is no specific hardware requirement to use CAB.

2.2 Prerequisites

The Code Analysis Bot has been successfully tested with Automation Anywhere v11.3.x. Feel free to report any issues with other versions, and the developer will happily adjust to support other versions wherever possible.

2.3 Security Measures

N/A

2.4 Disclaimers

N/A

3. Getting Started

3.1 Skill Matrix

N/A

CAB consists of the following TaskBots:

- CAB_SUBTASK_ExecuteCodeValidation
- CAB_SUBTASK_LogIssueToExcel
- CAB_SUBTASK_ReadConfigFile
- CAB_MAIN_CodeReviewAnalysis

The following Bots are also included (all obtained from the Bot Store):

- “Perform Multiple Microsoft Excel Operations” -
- “Perform Various Excel Operations”
- “Configurable Event Handler”
- “Perform various Array Operations”

See appendix for more information on these Bots.

3.2 Installation Hierarchy

3.2.1 Process Log Folder (Shared Drive)

The following structure should be created on your shared drive (NAS) – wherever you generally store log files related to automation.

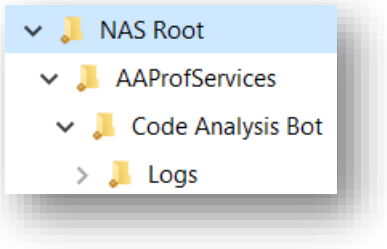


Figure 3 - Folder structure for Log files generated by CAB

3.2.2 Bot Runner

CAB can run on any Bot Runner. There are at least (2) potential options for deploying CAB on a Bot Runner:

1. **Deploy CAB to all Bot Runners**. Use CAB to scan source code deployed to that Bot Runner.
    a. One advantage of this approach is that CAB can validate that any TaskBots (.atmx files) referenced in 'Run Task' commands actually exist. This option is only possible if CAB is run on the same Bot Runner where the source code has been deployed.

2. **Deploy CAB to a central Bot Runner**. Copy any related source code (ex. atmx/txt) to the central Bot Runner and run CAB from there.
    a. This is a more simple deployment option. Prior to running CAB to analyze source code, ensure that the config.XML is updated as necessary for any Use Case-specific parameters.

3.3 Quick Start

3.3.1 Setup

The Code Analysis Bot is designed to be reusable / shared, such that it can be used by Bots supporting any Use Case to identify issues impacting code quality.

Deploying CAB to enable analysis of source code involves the following steps:

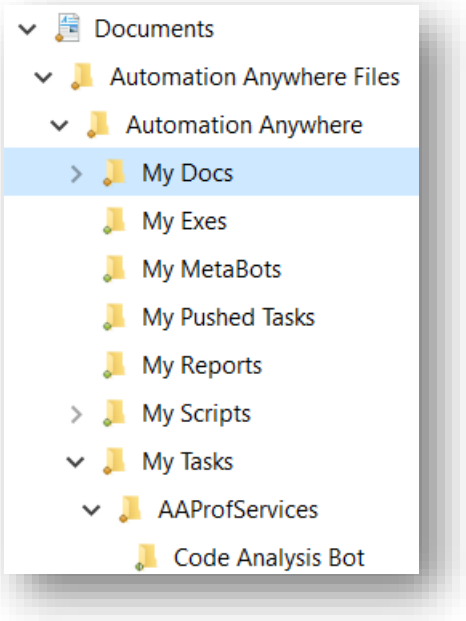
- Setup of CAB Folders and Files
- Configuration
- Running CAB on TaskBot Code

3.3.2 Setup of CAB Folders and Files

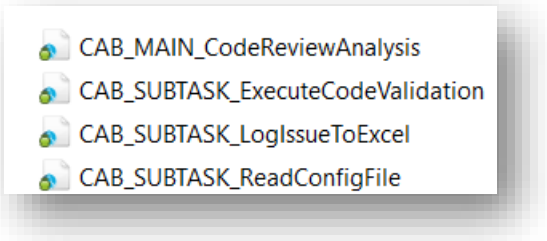
CAB is designed to be executed from any Automation Anywhere client.

The following initial steps should be taken to setup CAB for use in analyzing code:

1. Run the installation package from the Bot Store
2. Move the .atmx files in the **My Tasks** folder to the following location, under My Tasks\AAProfServices\Code Analysis Bot:



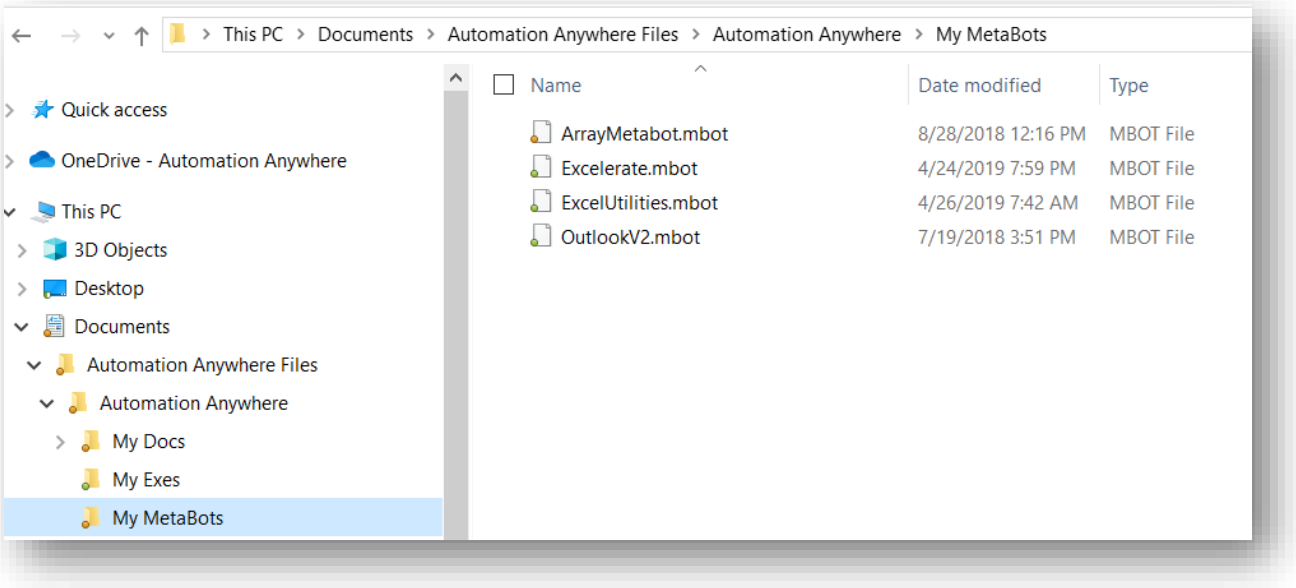
A total of (4) .atmx files should be present.



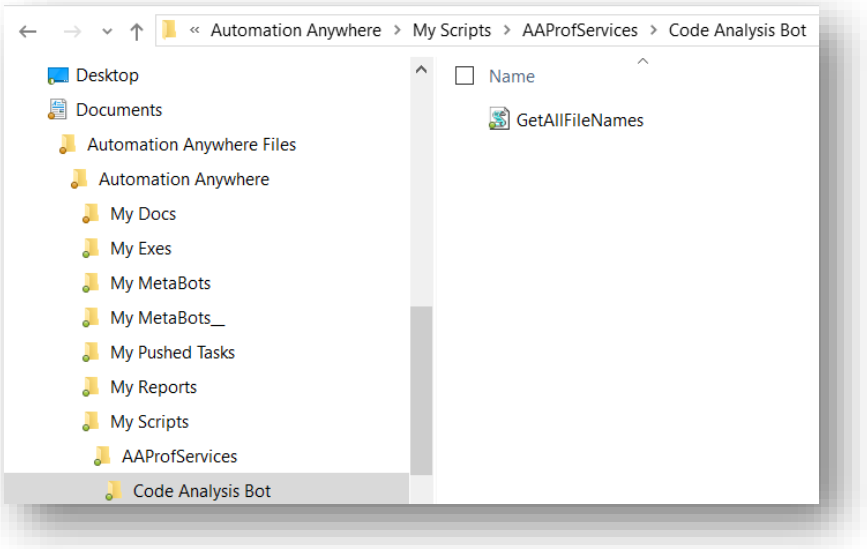
3. Confirm that the following Metabot files are available in the **My Metabots** folder:

   A total of (4) .mbot files should be present:

   - ArrayMetabot
   - Excelerate
   - ExcelUtilities
   - OutlookV2



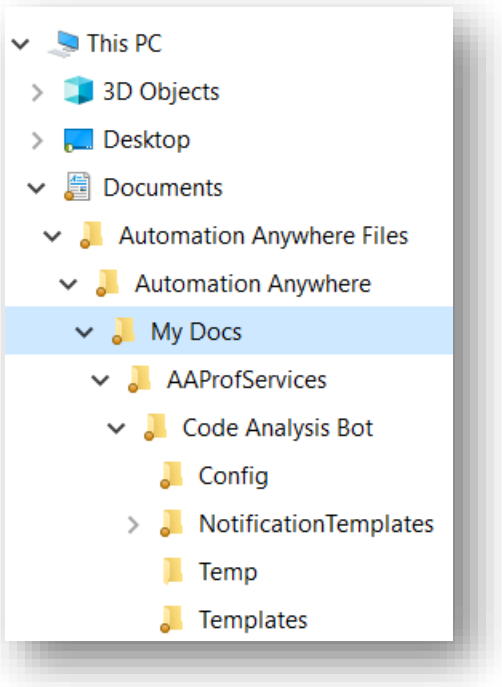
4. Move the VB script “GetAllFileNames.vbs” to “My Scripts\AAProfServices\Code Analysis Bot”



5. Setup folders under 'My Docs' path

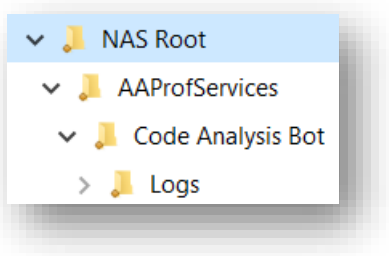
   Move the (4) folders from 'Input Folders' to the 'My Docs' path.

   After copying, your folder structure should look as follows:



6. Define or create (if necessary) an 'input' folder that will contain all TaskBots to be analyzed – note: this directly should contain *only* .atmx files.
    a. NOTE: if the code you wish to analyze uses the 'Configurable Event Handler' from the Bot Store, the EventHandling.XML file will also be stored in this 'input' folder.

7. Create a folder for CAB to maintain 'log' files – generally on a shared drive (NAS) location:



NOTE: this folder path is configurable in the config.XML file.

8. Create an 'output' folder that will contain the Excel workbook with results from the code analysis

That's all for initial setup of folders and files – just a little configuration, and you will be ready to start analyzing code.

3.3.3 Configuration

Configuration involves updating the following (2) files:

- Config.XML – general configuration for CAB, including parameters related to the Use Case/Process whose source code you want to analyze
- EventHandling.XML (optional) – the file for configuration of events, if the Use Case/Process being analyzed used the Configurable Event Handler

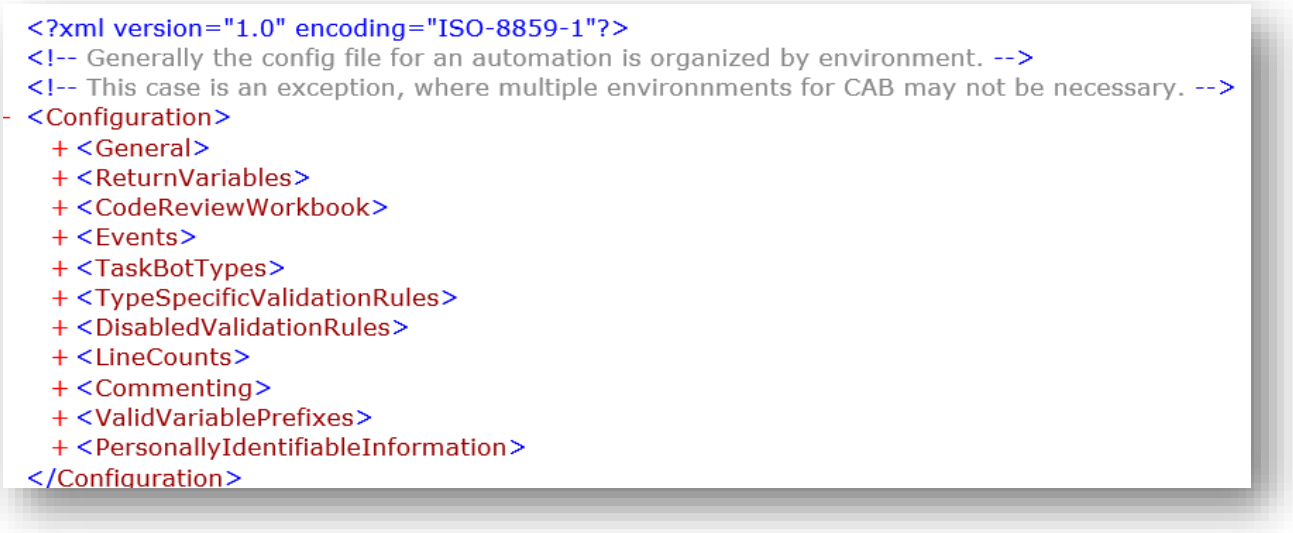
Config.XML

The config.XML supports configuration of the following:

- Variable name prefixes
- Code Review (Excel) workbook structure (ex. Cell / column references)
- TaskBot "Types" – (ex. Main vs. SubTask) and rules that should be only executed for that 'Type'
- TaskBot naming convention
- Return variable names (ex. Variables returning results from execution of Run Task/Script/Logic
- Disabling specific rules
- Tolerances for total line counts in a TaskBot
- Required commenting (as a % of total lines of code)

Commenting in the supplied config.xml file gives examples of how to configure.

The following is a sample of the high-level structure of config.XML:



Element	Description
	CABConfiguration
DefaultSupportEmailAddress	Default email recipient for any errors encountered during execution of the code analysis.
DefaultSenderEmailAddress	Default sender for any errors encountered during execution of the code analysis.
LogFolder	Folder where logs generated by CAB are stored.

ProtectExcelWorksheets	'Yes' to indicate that worksheets in the Excel workbook are protected (require being unlocked to make updates). If locked, the password should be stored in an accessible locker in the AA Credential Vault. NOTE: By default, worksheets are *not* locked. If you wish to lock specific worksheets, which is recommended, follow the steps in Protecting Excel Worksheets If you wish NOT to have the worksheets protected, edit and remove protection from the sheets in the Excel workbook template in the /Templates folder. Default password is 'password'.
GeneralCodingStandards	
TaskNameReference	The reference for name of the Task that is expected in Log To File statements (ex. $AATaskName$, $vTaskName$
AuditLogPrefix	Ex. AuditLog_
ErrorLogPrefix	Ex. ErrorLog_
EventLogPrefix	Ex. EventLog_
DesktopCleanerBot	The name of the TaskBot used to cleanup/sanitize the desktop before/after execution of an automated process
MaxCharacterPathLength	
UseCaseConfig	
MyDocRootPath	For the Use Case whose source code is being analyzed, what is the root path for documents? Ex. $AAApplicationPath$\Automation Anywhere\My Docs\Finance\FinanceProcessName
MyScriptRootPath	For the Use Case whose source code is being analyzed, what is the root path for scripts? Ex. $AAApplicationPath$\Automation Anywhere\My Scripts\Finance\FinanceProcessName
MyTaskRootPath	For the Use Case whose source code is being analyzed, what is the root path for tasks?

	Ex. $AAApplicationPath$\Automation Anywhere\My Tasks\Finance\FinanceProcessName
ReturnVariables	
VariableName	The name of a variable used for a value returned from a Sub-Task (TaskBot) or Metabot (Logic)
CodeReviewWorkbook	
Version	A version reference for the Code Review Workbook. In a future version of CAB, this can be referenced against the CAB code as a control point to ensure the latest version of the process & artifacts/code are used.
FileNamePrefix	Prefix for the name of the Code Review Workbook
ResultsTemplateWorkSheetName	Name of the worksheet that is a template where results of the code review will be updated
ChecklistWorkSheetName	Name of the worksheet containing the checklist used for the code review, both ones executed by CAB, and ones that could be assessed manually.
SummaryWorkSheetName	Name of the worksheet where results of the code review will be summarized
CellRanges	This section has a series of nodes with cell references within the Code Review Workbook
ChecklistWorkSheetStructure	This section has a series of nodes with column references within the Code Review Workbook

EventHandling.XML

The EventHandling.xml is used by the Event Handler to action event raised by CAB.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <EventHandling>
  - <EVENT Category="error" EventCode="ERROR_001">
      <Description>Error 001 occurred</Description>
      <EmailNotify>Yes</EmailNotify>
      <EmailCCRecipient/>
      <EmailBCCRecipient>mark.goodaire@automationanywhere.com</EmailBCCRecipient>
      <EmailRecipient>mark.goodaire@automationanywhere.com</EmailRecipient>
      <EmailSubject>Code Analysis Bot - A general failure occurred in 'Main' Taskbot</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Fail</TaskStatus>
    </EVENT>
  - <EVENT Category="error" EventCode="ERROR_002">
      <Description>Error 002 occurred</Description>
      <EmailNotify>Yes</EmailNotify>
      <EmailCCRecipient/>
      <EmailBCCRecipient>mark.goodaire@automationanywhere.com</EmailBCCRecipient>
      <EmailRecipient>mark.goodaire@automationanywhere.com</EmailRecipient>
      <EmailSubject>Code Analysis Bot - Failure Writing to Excel workbook</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Fail</TaskStatus>
    </EVENT>
  - <EVENT Category="exception" EventCode="EXCEPTION_001">
      <Description>Exception 001 occurred</Description>
      <EmailNotify>No</EmailNotify>
      <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
      <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
      <EmailRecipient>user@emaildomain.com</EmailRecipient>
      <EmailSubject>Exception 001 occurred</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Fail</TaskStatus>
    </EVENT>
  - <EVENT Category="event" EventCode="EVENT_001">
      <Description>Code QA PASSED - No Issues Found</Description>
      <EmailNotify>No</EmailNotify>
      <EmailRecipient>user@emaildomain.com</EmailRecipient>
      <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
      <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
      <EmailSubject>Code QA PASSED</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Pass</TaskStatus>
    </EVENT>
  - <EVENT Category="event" EventCode="EVENT_002">
      <Description>Code QA PASSED - Only Low Priority Issues Found</Description>
      <EmailNotify>No</EmailNotify>
      <EmailRecipient>user@emaildomain.com</EmailRecipient>
      <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
      <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
      <EmailSubject>Code QA PASSED</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Pass</TaskStatus>
    </EVENT>
  - <EVENT Category="event" EventCode="EVENT_003">
      <Description>Code QA FAILURE - Medium Priority Issues Found</Description>
      <EmailNotify>No</EmailNotify>
      <EmailRecipient>user@emaildomain.com</EmailRecipient>
      <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
      <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
      <EmailSubject>Code QA FAILURE</EmailSubject>
      <AttachFile>Yes</AttachFile>
      <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
      <TaskStatus>Pass</TaskStatus>
    </EVENT>
  - <EVENT Category="event" EventCode="EVENT_004">
      <Description>Code QA FAILURE - High Priority Issues Found</Description>
```

3.3.4 Running CAB to Analyze Source Code

After installing CAB on a Bot Runner, perform the following steps to analyze TaskBot source code:

1. Validate & update the config.XML file where required
2. Put the source code (.atmx or .text files) in the 'input' folder. Also include the EventHandling.xml if you are using the Configurable Event Handler (see Appendix A)
3. Put the Excel Code Review workbook for the Use Case/Process in the 'output' folder
    a. Note: this step can be skipped if this is the first time the code has been analyzed using CAB.
4. Execute the 'Main' TaskBot from the Automation Anywhere client.

The following notifications may appear:

1. CAB will notify you if there is no EventHandling.xml file in the 'Input' folder. In this case, validation of that XML against the source code will not be possible.
2. CAB will request you choose one of the following run modes:
    a. **Developer** – use this mode if you are the developer of the code
    b. **Initial/Peer QA** – use if this is a review after completion of Development (ie. Prior to execution of test cycles)
    c. **Final QA/Approval** – use if this is the 'final' review prior to Production deployment

    NOTE: The results worksheet populated in Excel is named with the 'run mode' selected, for the purposes of tracking a history of the results as the Use Case moves through its LifeCycle.

4. Interpreting the Results

At the conclusion of execution, the detailed results worksheet will have been created showing results of the analysis, and the 'Summary' worksheet will be updated with results from the latest execution.

The 'Pass/Fail' is based on what rules failed, and their associated severity. If any medium or high severity issue is found, the result will be a 'Fail'.

Note: It is recommended that the outcome of the Code Review process have a clear pass/fail, to ensure there is no ambiguity regarding the standards applied, and any actions required to meet/exceed the standard prior to promoting code.

"**Low**" severity is reserved for situations where the finding may be 'subjective', and requires a review by a Solution Architect/Lead Developer to determine any necessary actions.

The presence of a 'Medium' or 'High' severity finding would cause a failure of the Code Review and indicate a need for action.

5. Communicating The Results

CAB uses the Configurable Event Handler, which enables it to send an email communication with code review results attached. Note that email communication is enabled *only* for the 'Post-Dev' (Initial QA) and 'Pre-Prod' (Final QA) modes, *not* the 'Developer' mode.

The following Events are defined in the EventHandling.xml for CAB:

- EVENT_001 – raised if the analysis completes, with no findings.
- EVENT_002 – raised if the analysis completes, with only low severity findings.
- EVENT_003 – raised if the analysis completes, with medium severity findings.
- EVENT_004 – raised if the analysis completes, with high severity findings.

To enable email communication of the code analysis results, following the following steps to configure event codes accordingly:

1. Edit the Eventhandling.xml file, and set the <EmailNotify> value to 'Yes' for the appropriate event codes.
2. Update the email recipient, CC recipient, BCC recipient, subject as necessary to define the recipient(s) for the communication
3. Ensure the <AttachFile> value is **Yes** to ensure the code review results are attached to the email when those events occur

```
<EVENT EventCode="EVENT_001" Category="event">
    <Description>Code QA PASSED - No Issues Found</Description>
    <EmailNotify>No</EmailNotify>
    <EmailRecipient>user@emaildomain.com</EmailRecipient>
    <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
    <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
    <EmailSubject>Code QA PASSED</EmailSubject>
    <AttachFile>Yes</AttachFile>
    <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
    <TaskStatus>Pass</TaskStatus>
</EVENT>


<EVENT EventCode="EVENT_002" Category="event">
    <Description>Code QA PASSED - Only Low Priority Issues Found</Description>
    <EmailNotify>No</EmailNotify>
    <EmailRecipient>user@emaildomain.com</EmailRecipient>
    <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
    <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
    <EmailSubject>Code QA PASSED</EmailSubject>
    <AttachFile>Yes</AttachFile>
    <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
    <TaskStatus>Pass</TaskStatus>
</EVENT>

<EVENT EventCode="EVENT_003" Category="event">
    <Description>Code QA FAILURE - Medium Priority Issues Found</Description>
    <EmailNotify>No</EmailNotify>
    <EmailRecipient>user@emaildomain.com</EmailRecipient>
    <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
    <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
    <EmailSubject>Code QA FAILURE</EmailSubject>
    <AttachFile>Yes</AttachFile>
    <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
    <TaskStatus>Pass</TaskStatus>
</EVENT>

<EVENT EventCode="EVENT_004" Category="event">
    <Description>Code QA FAILURE - High Priority Issues Found</Description>
    <EmailNotify>No</EmailNotify>
    <EmailRecipient>user@emaildomain.com</EmailRecipient>
    <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
```

6. Logs

As CAB processes, several types of logs are generated.

- Audit Log – a general log of the processing of CAB as it analyzes source code
- Event Log – events such as the completion of processing are recorded as 'events' in this log, which is then read by the Configurable Event Handler to process them accordingly.
- Error Log – used to record any 'error' events (ie. errors caught by Error Handling code blocks)

7. Troubleshooting & Support

7.1 Support

Should you have any additional requirements for your organization that you would like to be incorporated into the Code Analysis Bot, please reach out to your Customer Success/Account Manager at Automation Anywhere, who will support you in engaging the Professional Services team.

Appendix A: Record of Changes

No.	Version Number	Date of Change	Author	Notes
1	*1.0*	*Dec 10th, 2019*	*AA Professional Services*	*Initial Submission to Bot Store*

Appendix B: Acronyms

No.	Acronym	Description

Appendix C: References

No.	Topic	Reference Link
1.	Configurable Event Handler	https://botstore.automationanywhere.com/bot/configurable-event-handler/
2.	"Perform Multiple Microsoft Excel Operations"	https://botstore.automationanywhere.com/bot/perform-multiple-microsoft-excel-operations/
3.	"Perform Various Excel Operations"	https://botstore.automationanywhere.com/bot/perform-various-excel-operations-2/
4.	"Perform various Array Operations"	https://botstore.automationanywhere.com/bot/perform-various-array-operations/