LTI's Enhanced Code Analysis Bot (eCAB) - Readme

Version: 1.0
Date: 04th Aug 2020

A Larsen & Toubro Group Company

Bot Overview

The code review process is a critical component of any Bot development lifecycle, and LTI's eCAB can assist by reducing the amount of manual effort to analyse code that might require correction. LTI believes that even the review activities can be best implemented through cooperation between Digital and Human Workers. Both plays a distinct role to execute the process most effectively and efficiently:

- **LTI's eCAB –** compares 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, LTI's eCAB 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.

Pre-Requisites

- Automation Anywhere Enterprise v11.x
- MS Excel 2019
- Outlook 2019

Installation

- Download the bot from Bot Store.
- Double click on the .msi file.
- On Welcome to Installation wizard, click Next to continue.
- Click I agree to the terms in the license agreement radio button to accept the agreement.
- Get/Copy the License key from Bot Store Downloads into License Key, click Next to continue.
- Click Install to begin the installation.
- Click Finish to complete the installation.
- To view the installation, go to 'My Tasks' folder on AAE Client to see bot files.

Uninstall

- Open Add/Remove Programs
- Select the Bot/Digital Worker to be installed.
- Click uninstall.

Folder structure of the content in the AA Dir:

<AA Directory>

- **My Tasks**
  - **Bot Store**
    - **My Tasks**
      - AAProfServices
        - Code Analysis Bot
          - ECAB_MAIN_CodeReviewAnalysis.atmx
          - ECAB_SUBTASK_CreateCharts.atmx
          - ECAB_SUBTASK_ExecuteCodeValidation.atmx
          - ECAB_SUBTASK_LogIssueToExcel.atmx
          - ECAB_SUBTASK_ReadConfigFile.atmx
        - LIBRARY
          - EventHandler.atmx
          - GlobalSettings.atmx
  - **My Docs**
    - **AAProfServices**
      - **Code Analysis Bot**
        - **Config**
          - Config.xml
          - EventHandling.xml
        - **NotificationTemplates**
          - standard_event_notification.htm
          - standard_event_notification2.htm
          - **standard_event_notification_files**
            - colorschememapping.xml
            - filelist.xml
            - themedata.thmx
        - **Templates**
          - CodeReview Template - eCAB.xlsx
  - **My Scripts**
    - **AAProfServices**
      - **Code Analysis Bot**
        - GetAllFileNames.vbs

How to Use the Bot?

Input

After installing the package from the Bot Store, you should find sample output in the 'My Tasks\Bot Store\EnhancedCodeAnalysisBot-Larsen&ToubroInfotech(LTI)\Sample Output' folder. Similarly, a file "**CodeReview Template - eCAB - Sample.xlsx"** is provided, that shows the sample output. Input to LTI's eCAB is the following:

- Input folder path (Having one or more atmx/text files)
- Output folder path
- 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
Convenience	.atmx files can be directly provided to LTI's eCAB; for .txt, each TaskBot must be opened in the AA Client, 'Save as Text'.
Accuracy/Output	.atmx files may not always produce accurate line number references for issues found. Note: this is not a significant issue, given the 'output' of LTI's eCAB 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.
Validation Scope	Some validations are only possible for specific formats. • Validation of unused variables – only available in .atmx • Validation of unused/disabled source code – only available in .atmx • Get all dependencies - only available in .atmx • Validation for specific Error handling – only available in .txt

Output

The 'CodeReview Template – eCAB' works as input file and provided a detailed analysis of the code along with the visualization of the critical parameters to be considered for prioritization of the issues, tasks and architectural capability to be addressed in the code. It can even help in grooming the team for specific architectural skill.

Output Summary Data.

Figure 2- Sample - Summary Results from Analysis

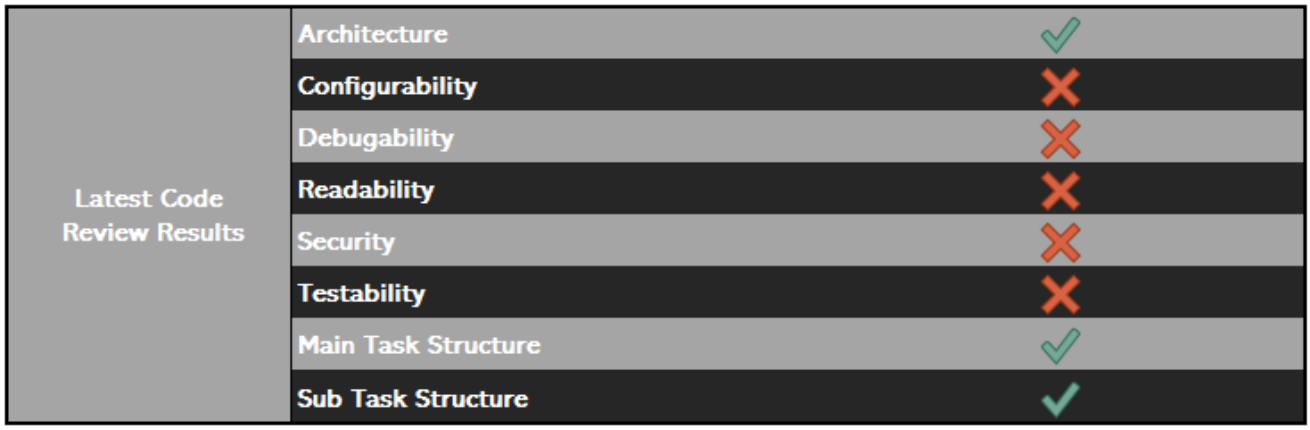
Key information in this worksheet: The following should be populated manually before running LTI's eCAB on source code:

- **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

The following is automatically populated by LTI's eCAB during execution:

- **Attended or Unattended?** – this information is relevant because some validations are only relevant for un-attended bots. Ex. Existence of Message Box 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 category with at least (1) finding.



1) Provide list of the commands in the configuration file, for that user needs to apply check for hardcode-path validation (8.05). (Xml-node: CommandsWithPathParameter)

```
    <!-- the Excel worksheets    -->
    <!-- If set to No, expectation is for all Excel worksheets to be un-protected   -->
    <ProtectExcelWorksheets>No</ProtectExcelWorksheets>
    <!--  The following rules are relevant if the configurable Event Handler was used for the automation whose source code is being
    <!--  These rules will be *disabled* if the <UsedConfigurableEventHandler> node is set to 'No'   -->
    <RulesForConfigurableEventHandler>3.12, 3.13, 3.14,3.15, 3.19, 7.02c, 7.02e, 7.03a, 7.03b, 7.03c</RulesForConfigurableEventHandler>
    <RulesForUnattendedBot>8.01</RulesForUnattendedBot>
    <CommandsWithPathParameter>If File,If Folder,Session,Create File,Create Folder, Delete File, Delete Folder, Copy File, Copy
      Folder,Open</CommandsWithPathParameter>
  </CABConfiguration>
  <!-- The remainder of the XML sections here relate to standards, naming conventions, or other parameters for executing the validat
- <GeneralCodingStandards>
```

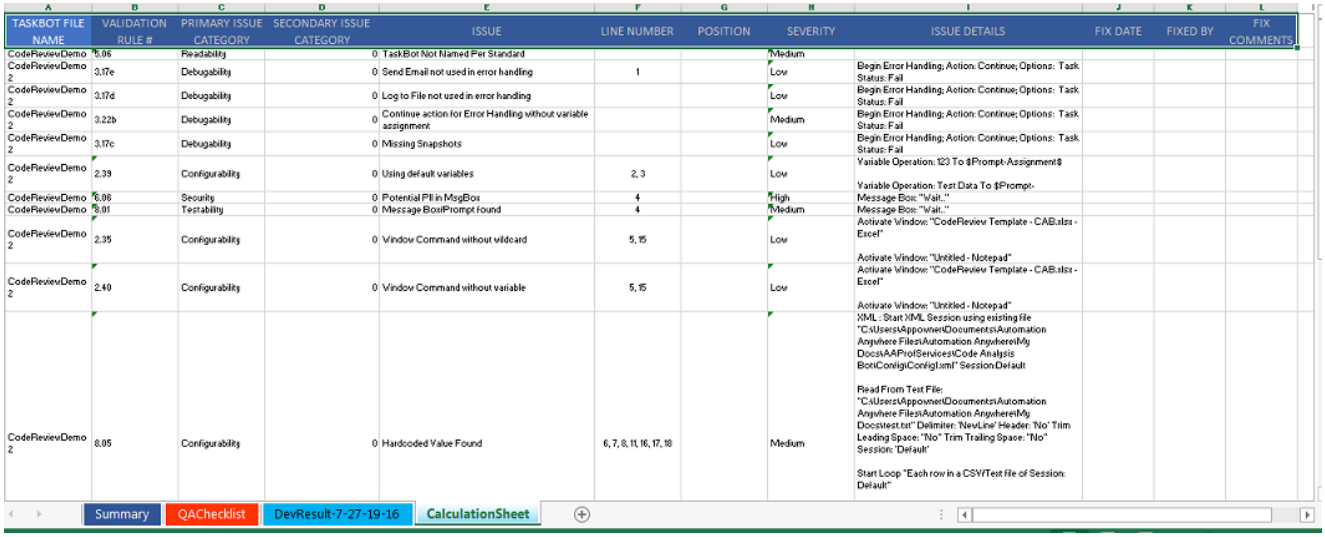
2) User can provide following parameters in configuration file according to project requirements-

I. Minimum line of code.
II. Preferred max line of code.
III. Absolute max line of code.
IV. Preferred minimum percentage of lines of comments in the code.
V. Absolute minimum percentage of lines of comments in the code.

```
- <LineCounts>
    <MinLinesOfCodeInTaskBot>49</MinLinesOfCodeInTaskBot>
    <PreferredMaxLinesOfCodeInTaskBot>500</PreferredMaxLinesOfCodeInTaskBot>
    <AbsoluteMaxLinesOfCodeInTaskBot>750</AbsoluteMaxLinesOfCodeInTaskBot>
  </LineCounts>
  <!-- The following are used by validation rules that relate to the extent of commenting in a TaskBot   -->
- <Commenting>
    <PreferredMinPercentageOfLinesThatAreComments>25</PreferredMinPercentageOfLinesThatAreComments>
    <AbsoluteMinPercentageOfLinesThatAreComments>15</AbsoluteMinPercentageOfLinesThatAreComments>
  </Commenting>
  <!-- The following are used by validation rules that check for standard naming conventions applied to variak
```

3) In your 'Output' folder, move or delete any existing Excel templates.

Note: If no template is found in the 'Output' folder when LTI's eCAB runs to analyse code, it will copy the template from the 'Templates' folder.

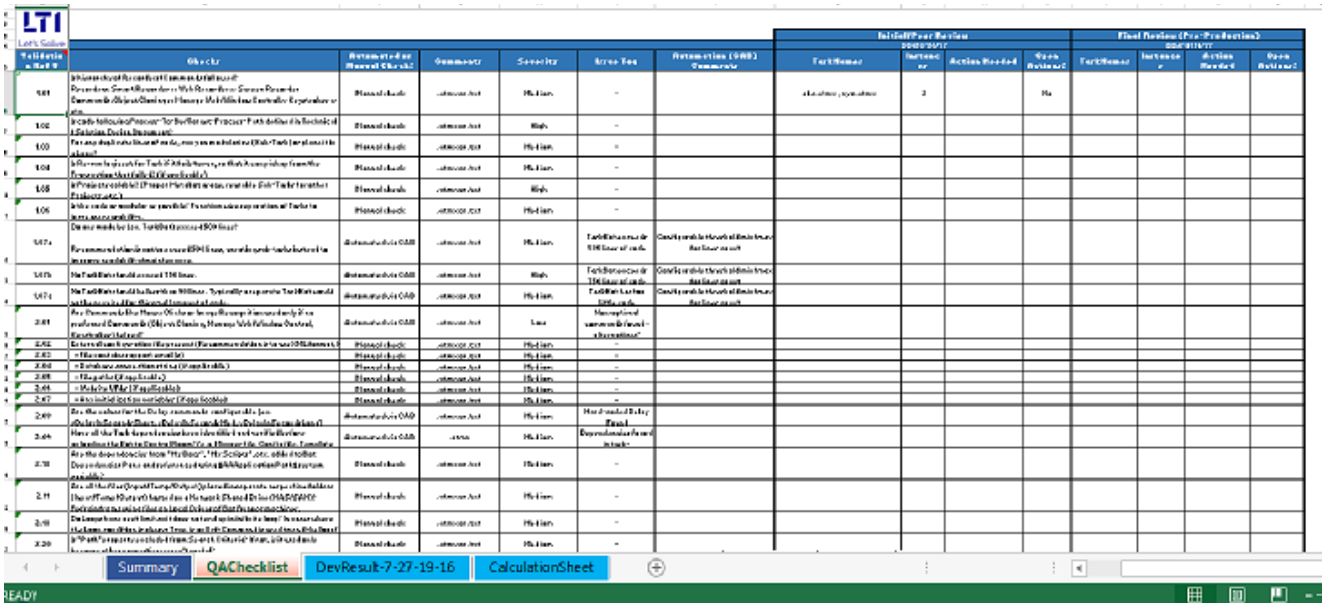


TASKBOT FILE NAME	VALIDATION RULE #	PRIMARY ISSUE CATEGORY	SECONDARY ISSUE CATEGORY	ISSUE	LINE NUMBER	POSITION	SEVERITY	ISSUE DETAILS	FIX DATE	FIXED BY	FIX COMMENTS
CodeReviewDemo	5.06	Readability	0	TaskBot Not Named Per Standard			Medium				
CodeReviewDemo 2	3.17e	Debugability	0	Send Email not used in error handling	1		Low	Begin Error Handling; Action: Continue; Options: Task Status: Fail			
CodeReviewDemo 2	3.17d	Debugability	0	Log to File not used in error handling			Low	Begin Error Handling; Action: Continue; Options: Task Status: Fail			
CodeReviewDemo 2	3.22b	Debugability	0	Continue action for Error Handling without variable assignment			Medium	Begin Error Handling; Action: Continue; Options: Task Status: Fail			
CodeReviewDemo 2	3.17c	Debugability	0	Missing Snapshots			Low	Begin Error Handling; Action: Continue; Options: Task Status: Fail			
CodeReviewDemo 2	2.33	Configurability	0	Using default variables	2, 3		Low	Variable Operation: 123 To $Prompt-Assignment$ Variable Operation: Test Data To $Prompt-			
CodeReviewDemo	5.08	Security	0	Potential PII in MsgBox	4		High	Message Box: "Wait.."			
CodeReviewDemo	5.01	Testability	0	Message Box/Prompt found	4		Medium	Message Box: "Wait.."			
CodeReviewDemo 2	2.35	Configurability	0	Window Command without wildcard	5, 15		Low	Activate Window: "CodeReview Template - CAB.xlsx - Excel" Activate Window: "Untitled - Notepad"			
CodeReviewDemo 2	2.40	Configurability	0	Window Command without variable	5, 15		Low	Activate Window: "CodeReview Template - CAB.xlsx - Excel" Activate Window: "Untitled - Notepad"			
CodeReviewDemo 2	8.05	Configurability	0	Hardcoded Value Found	6, 7, 8, 11, 16, 17, 18		Medium	XML : Start XML Session using existing file "C:\Users\Appowner\Documents\Automation Anywhere Files\Automation Anywhere\My Docs\AAProf\Services\Code Analysis Bot\Config\Config1.xml" Session:Default Read From Text File: "C:\Users\Appowner\Documents\Automation Anywhere Files\Automation Anywhere\My Docs\test.txt" Delimiter: 'NewLine' Header: 'No' Trim Leading Space: 'No' Trim Trailing Space: 'No' Session: 'Default' Start Loop "Each row in a CSV/Text file of Session: Default"			

Summary | QAChecklist | DevResult-7-27-19-16 | CalculationSheet

CodeReview Template – Overview

The following worksheets are part of the CodeReview Template:



QA Checklist – this checklist represents a 'Best Practices' for items that should be checked in a Code Review. In many cases the check can be performed by LTI's eCAB – these items are indicated by column F as "Automated via LTI's eCAB". 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.

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. **Checks:** description of the validation
e. **Automated or Manual:** indicates if the Code Analysis Bot (LTI's eCAB) supports the validation, or it currently must be executed manually
f. **Severity:** a rating of low, medium, high. See Interpreting Results section for explanation of severity levels.
g. **Issue Tag:** the text that will be used in the code review worksheet, when the issue is found in the source code
h. **Initial/ Peer Review:** this section of columns can be used by any feedback from those performing an initial/peer review of the source code
i. **Final Review:** this section of columns can be used by any feedback from those performing a final review/approval of the source code

Supported validations

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

Validation Ref #	Primary Category	Primary Sub Category	Checks	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.	Configurable threshold min/max for lines count
1.07b	Architecture	Task	No TaskBot should exceed 750 lines.	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.	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?	Any
2.08	Configurability	Delay	Are the values for the Delay commands configurable (ex. vDelayInSecondsShort, vDelayInSecondsMed, vDelayInSecondsLong)	Any
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 Docs, Scripts)	This validation supports .atmx format only.
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 reference Task/Script files actually exist?	Any
2.26	Configurability	Session	Are Session Names unique and meaningful? Do not use "Default" name.	Any
2.30	Configurability	Variables	Have all variables been created and named as per the required	Configurable variable naming standards

			standard format? (refer to naming-convention standards)	
2.32	Configurability	Variables	Have variables been used for numerical values, vs. hard-coding?	Any
2.3	Configurability	Variables	Any use of Automation Anywhere 'default' variables (ex. Prompt- Assignment) should be minimized, in favour of variables with meaningful names.	Any
2.41	Configurability	Variables	Are variables used to reference URLs, vs. static references?	Any
2.42a	Configurability	Variables	Are variables in the Variable Manager being used?	This validation supports .atmx format ONLY.
2.42b	Configurability	Variables	Are variables in the Variable Manager being un-necessarily passed to Sub-Tasks?	This validation supports .atmx format ONLY.
2.35	Configurability	Window title	Are Commands handling Window Titles properly? (Use wildcards "*" if part of Title is non-static)	Any
2.40	Configurability	Window title	Are variables used to reference Window Titles (if not using 'currently active window')	Any
2.38b	Configurability	Windows	Are Window Close commands included in a Loop to ensure they are closed?	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?	Configurable audit log naming
3.11	Debugability	Error handling	Does Task have every Command scoped under Error handling?	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?	Any
3.13	Debugability	Error handling	Are all Event Codes named per standard based on Event Type (Error vs. Exception vs. Event)?	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?	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?	ONLY if Configurable Event Handler used
3.17a	Debugability	Error Log	Has Error Log been maintained for capturing details of the Error?	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)?	Any
3.17c	Debugability	Error Log	Are snapshots captured of the error?	Any
3.17d	Debugability	Error Log	Is Log to File used in error handling?	Any
3.17e	Debugability	Error Log	Is Send Email used in Error Handling?	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?	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?	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?	This validation supports .txt format ONLY.
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?	ONLY if Configurable Event Handler used
3.19	Debugability	Event handling / notification	Is the XML-configurable Event Handler being called in all TaskBots to manage all Events / Exceptions /Errors raised during processing?	ONLY if Configurable Event Handler Used
3.21a	Debugability	Event handling / notification	Is the return indicator after a Sub- Task (ie. to see if an error occurred (ex. $xIsError$)?	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. $vOutput$)?	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. $vOutput$)?	Configurable 'return variable' names
5.03	Readability	Comments	Have 'Default Comments' added by Automation Anywhere been removed/replaced with meaningful comments?	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)	Any
5.05b	Readability	Comments	Source code for every TaskBot*must* have at least 15% of lines that are comments.	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	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?	Any
6.03	Security	Personal Info	All PII in automation commands and variables removed	Configurable PII Patterns
6.05	Security	Personal Info	No PII stored in logs	Configurable PII Patterns
6.06	Security	Personal Info	No PII shown in Message Boxes	Configurable PII Patterns
7.02c	Task Structure	Main Task Structure	No business exceptions thrown	ONLY if Configurable Event Handler Used
7.02e	Task Structure	Main Task Structure	At least (1) ERROR Event thrown.	ONLY if Configurable Event Handler Used
7.03b	Task Structure	Sub Task Structure	At least (1) EXCEPTION Event thrown.	ONLY if Configurable Event Handler Used
7.03c	Task Structure	Sub Task Structure	At least (1) ERROR Event thrown.	ONLY if Configurable

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..)	Any
8.05	Configurability	Path Variable	Does command having hardcode file or folder path?	Any
4.03	Testability	Task run	Does the code properly execute 'desktop clean-up' to kill applicable applications, before and after execution?	Configurable name of 'cleanup' Bot
9.97	Testability	Task run	Is the path name using?$AAApplicationPath$, but with an excessive character length?	Any
9.99	Testability	Task run	Missing TaskBot or Script referenced in Run Task/Run Script.	Must execute LTI's eCAB on Bot Runner where Bot being analysed will be Deployed

Requirements & Prerequisites

System Requirements

There is no specific hardware requirement to use LTI's eCAB.

Prerequisites

The LTI's Enhanced 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.

Security Measures

N/A

Disclaimers

N/A

Skill Matrix

N/A

Getting Started

LTI's eCAB consists of the following TaskBots:

- ECAB_SUBTASK_ExecuteCodeValidation
- ECAB_SUBTASK_LogIssueToExcel
- ECAB_SUBTASK_ReadConfigFile
- ECAB_MAIN_CodeReviewAnalysis
- ECAB_SUBTASK_CreateCharts

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

- "Perform Multiple Microsoft Excel Operations"
- "Configurable Event Handler"
- "Perform various Array Operations"

Installation Hierarchy

Bot Runner

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

1. **Deploy LTI's eCAB to all Bot Runners.** Use LTI's eCAB to scan source code deployed to that Bot Runner. One advantage of this approach is that LTI's eCAB can validate that any TaskBots (.atmx files) referenced in 'Run Task' commands exist. This option is only possible if LTI's eCAB is run on the same Bot Runner where the source code has been deployed.
2. **Deploy LTI's eCAB to a central Bot Runner.** Copy any related source code (ex. atmx/txt) to the central Bot Runner and run LTI's eCAB from there. This is one more simple deployment option. Prior to running LTI's eCAB to analyse source code, ensure that the config.XML is updated as necessary for any Use Case-specific parameters.

Quick Start

Setup

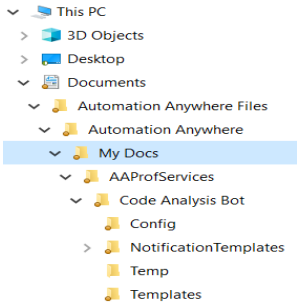
The LTI's eCode Analysis Bot is designed to be reusable / shared, such that it can be used by Bots supporting any Use Case to identify issues impacting code quality. Deploying LTI's eCAB to enable analysis of source code involves the following steps:

Setup of LTI's eCAB Folders and Files

LTI's eCAB is designed to be executed from any Automation Anywhere client. The following initial steps should be taken to setup LTI's eCAB for use in analyzing code:

1. Run the installation package from the Bot Store
2. Move the LIBRARY folder in AA My Task Folder.
3. Move the .atmx files in the My Tasks folder to the following location, under My Tasks\AAProfServices\Code Analysis Bot: A total of (5) .atmx files should be present.
    1. ECAB_MAIN_CodeReviewAnalysis.atmx
    2. ECAB_SUBTASK_CreateCharts.atmx
    3. ECAB_SUBTASK_ExecuteCodeValidation.atmx
    4. ECAB_SUBTASK_LogIssueToExcel.atmx
    5. ECAB_SUBTASK_ReadConfigFile.atmx

4. Confirm that the following Metabot files are available in the My Metabots folder: A total of (4) .mbot files should be present:
    1. ArrayMetabot
    2. Excelerate
    3. ExcelUtilities
    4. OutlookV2
5. Move the VB script "GetAllFileNames.vbs" to "My Scripts\AAProfServices\Code Analysis Bot"
6. Setup folders under 'My Docs' path.
7. Move the (4) folders from 'Input Folders' to the 'My Docs' path. After copying, your folder structure should look as follows:



8. Define or create (if necessary) an 'input' folder that will contain all TaskBots to be analyzed Note: This directly should contain only .atmx and text file only.
   NOTE: if the code you wish to analyse uses the 'Configurable Event Handler' from the Bot Store, the EventHandling.XML file will also be stored in this 'input' folder.
9. 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.

Configuration

Configuration involves updating the following (2) files:

- Config.XML – general configuration for LTI's eCAB, 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:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generally the config file for an automation is organized by environment. -->
<!-- This case is an exception, where multiple environnments for CAB may not be necessary. -->
- <Configuration>
  + <General>
  + <ReturnVariables>
  + <CodeReviewWorkbook>
  + <Events>
  + <TaskBotTypes>
  + <TypeSpecificValidationRules>
  + <DisabledValidationRules>
  + <LineCounts>
  + <Commenting>
  + <ValidVariablePrefixes>
  + <PersonallyIdentifiableInformation>
</Configuration>
```

Element	Description
LTI's eCAB Configuration	
Default Support Email Address	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 LTI's eCAB are stored.
General Coding Standards	
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	200
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\Department\ProcessName
MyScriptRootPath	For the Use Case whose source code is being analyzed, what is the root path for scripts? Ex. $AAApplicationPath$\Automation Anywhere\My Scripts\ Department\ProcessName
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

Return Variables	
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 LTI's eCAB, this can be referenced against the LTI's eCAB 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 LTI's eCAB, 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 raise

```
<?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>
```

Running LTI's eCAB to analyze Source Code team

After installing LTI's eCAB 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

Note: this step can be skipped if this is the first time the code has been analysed using CAB.

Execute the 'Main' TaskBot from the Automation Anywhere client. The following notifications may appear:

1. LTI's eCAB 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. LTI's eCAB 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 Life Cycle.

Intercepting 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.

Communicating the Results

LTI's eCAB 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 LTI's eCAB:

- 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.

```
<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>
```

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.

Ensure the <AttachFile> value is Yes to ensure the code review results are attached to the email when those events occur.

Logs

As LTI's eCAB processes, several types of logs are generated.

- **Audit Log –** a general log of the processing of LTI's eCAB as it analyses 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 (i.e. errors caught by Error Handling code blocks)

Troubleshooting & Support

Support

If you believe you have encountered a bug/defect, please provide details to the support mail id. Should you have any additional requirements for your organization that you would like to be incorporated into the LTI's eCAB, please reach out to your IAT CoE team who will support you in engaging support team. This will not be a dedicated support but CoE team will try to address the issues managing their priorities.