Event Handler

Readme

Version 1.0

21/11/2019

Table of Contents

1. Introduction

This document contains all essential information for the usage of a configurable "Event Handler, that can be leveraged from any automation Use Case. This manual includes a description of the functions and capabilities and step-by-step procedures for setup & configuration of the Bot.

1.1 Overview

A common property of all automations is that key 'events' may occur, that require specific action. This action is often logging the information for subsequent analysis, and notification. The Event Handler Bot supports Automation Anywhere's Best Practices for Bot Development, by enabling this functionality in a way that is configurable – minimizing any need for 'hard-coding' the definition.

As we build Bots, there are a few problems that can arise, in particular as an organization scales in the volume of Use Cases, and people developing automations:

1. **Lack of standards for notification** – each person developing the Bot may adopt their own approach to tracking/sending notification
2. **Hard-coding** – actions and targeted recipients for communications may be hard-coded, requiring changes to code when these requirements change. Changes to a solution once deployed to Production introduce un-necessary risk, as changes could negatively impact the functionality.
3. **Lack of Exception Definition/Handling** – as the process for a Use Case is defined, there is often insufficient attention on the entire scope of potential process / system exceptions.

This EventHandler helps to address each of these challenges, as follows:

1. With the EventHandler, definition of notifications and required actions are defined using a standard approach across all processes/Use Cases
2. Rather than hard-coding, definition of 'events' and required actions are configurable via XML
3. Definition of 'exceptions' is embedded in an XML file, which could be defined early in the bot development lifecycle, and change to those definitions managed throughout the Bot's lifecycle.

1.1.1 What do we mean by an 'event'?

There are generally (3) different types of 'event' that might occur within an automation:

1. **Error**

   An 'error' is defined as a condition that is caught by Automation Anywhere's 'Event Handling' code block. Each of these events will have a value for both the $ErrorDescription$ and $ErrorLineNumber$ system variables.

2. **Exception**

   An 'exception' can be described as anything 'non-typical' that occurs, where specific code has been written to detect the condition. These could include:

   a) **System Exceptions** - unexpected system conditions (ex. A system expected to be available is not available)
   b) **Process Exceptions** – includes business rule failure (ex. A value is expected to be <100, but has a value of 101), or non-typical process path is required (ie. different than the processes typical "happy path".

3. **Event**

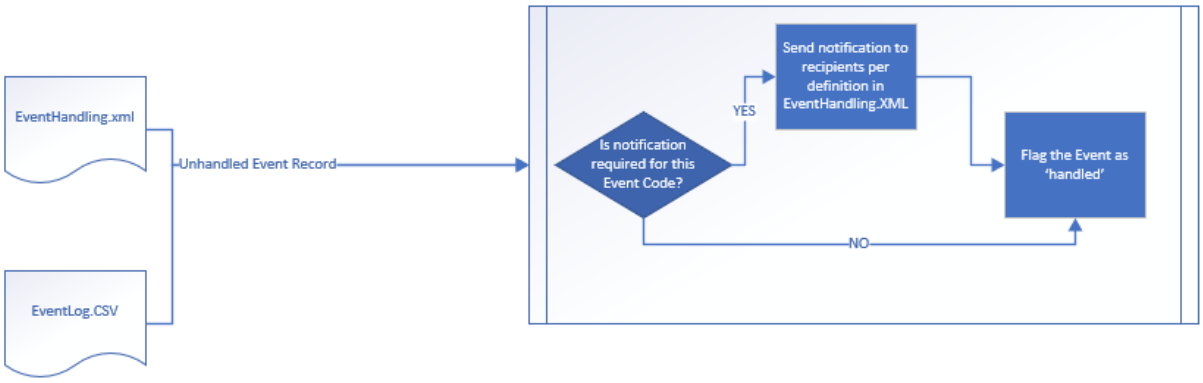
   An 'event' can be described as 'anything that does not fall into either an 'error' or an 'exception'. For example, it may be useful to log an event representing the final outcome of a process, which may require notification to stakeholders. This 'event' (the conclusion of the process) is neither an 'Error' nor an 'Exception', but nonetheless helpful to define as an event for subsequent analysis and/or notification.

1.1.2 Input & Output

The Event Handler Bot works as follows:

1. Loops through 'Events' logged in the current day's "Event Log". This log would contain any Errors, Exceptions, or other Events that a specific process has logged during its execution.

2. For any records marked as ‘unhandled’, perform the following:
    a. Look-up the Event Code referenced in the event transaction (ex. ERROR_012, EVENT_099), in the EventHandling.XML file, for the current environment where the Bot Runner is connected (ex. Dev, Test, Prod)
    b. If the Event Code configuration indicates that notification is required (EmailNotify=Yes)
        i. Determine if the Outlook Metabot is available; if not, the Send Mail command will be used
        ii. Read the HTML template referenced for the Event Code configuration, which will defined a template for the email body – if no HTML template is available, a default template for email body will be used
        iii. Send the email communication based on the Event Code configuration
        iv. Mark the transaction in the Event Log as ‘handled’



1.1.2.1 Email Notification Format

An HTML template (.htm or .html file) can be associated with each Event Code; this format will then be used for any notifications when that event occurs. This file can be referenced for each event code in the EventHandling.xml file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<EventHandling>
    <DEV>

        <EVENT EventCode="ERROR_001" EventType="error">
            <Description>Error 001 occurred</Description>
            <EmailNotify>Yes</EmailNotify>
            <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
            <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
            <EmailRecipient>user@emaildomain.com</EmailRecipient>
            <EmailSubject>Error 001 occurred</EmailSubject>
            <AttachFile>Yes</AttachFile>
            <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
            <TaskStatus>Fail</TaskStatus>
        </EVENT>
```

The folder containing the HTML templates for event notifications should be passed to the EventHandler using a variable called $cNotificationHTMLTemplateFolder$. It is recommended that this path be part of the processes 'config' file.

The HTML templates associated with event notifications for the process should be stored at the location defined by variable $vEmailBodyTemplateFilePath$.

The following is a sample HTML:

```
<font face="arial">
<h2>EventDescription</h2>
<p>EventDetailedDescription</p>
<p>
<p>Task Bot: TaskBot</p>
<p>Date/Time: DateTime</p>
<p>Event Code: EventCode</p>
<table cellpadding="10">
<tbody>
<tr>
<td>Event Description</td>
<td>EventDescription</td>
<tr>
<tr>
<td>Details</td>
<td>EventDetailedDescription</td>
<tr>

<td><img src="https://www.automationanywhere.com/images/automation-anywhere-logo-corporate-two-line-lg.png" width="150" height="75" /></td>
</tr>
</tbody>
</table>
</font>
```

In this file, a set of placeholders exist, that can be replaced with dynamic values to produce a 'custom' notification.

- ProcessName – the name of the process
- TaskBot – the name of the TaskBot that raised the event
- DateTime – the date/timestamp of the event
- EventCode – the event code of the event
- EventDescription – the general description of the event (from EventHandling.XML file)
- EventDetailedDescription – the specific description of the event

In the EventHandler, the email subject will be 'ProcessName – EventDescription' – of course, this can be changed as necessary.

Here is a sample text file for the HTML notification – the highlighted text will be replaced by the dynamic values defined by variables:

```
<font face="arial">
<h2>EventDescription</h2>
<p>EventDetailedDescription</p>
<p>
<p>Process: ProcessName</p>
<p>Task Bot: TaskBot</p>
<p>Date/Time: DateTime</p>
<p>Event Code: EventCode</p>
<table cellpadding="10">
<tbody>
<tr>
<td><img src="https://domain/companylogo.png" width="150" height="150" /></td>
</tr>
</tbody>
</table>
</font>
```

1.1.2.2 Default Email Body Content

What if no HTML Template is defined/found for the Event Code? If no HTML template is provided, the following will be used for the email body.

- Task Name : <$vCallingTaskName$>
- Event Code: EventCode
- Description : EventDescription
- Detailed Description: EventDetailedDescription
- Last Run Time: <lastruntime>

1.2 Common Use cases

The Event Handler is a generic framework that can be applied to literally any automation Use Case. The framework is flexible, and supports any definition of an 'event'. Appropriate actions (ie. email notification) will be sent based on the Event Code referenced for an event in the Event Log, and configuration in EventHandling.XML.

2. Requirements & Prerequisites

2.1 System Requirements

There is no specific hardware requirement to use the Event Handler.

2.2 Prerequisites

The Event Handler 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

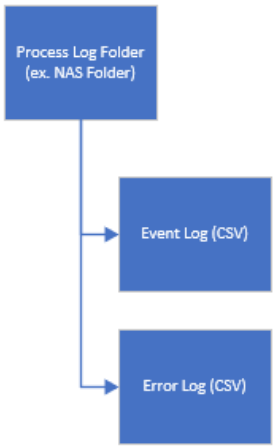
N/A

3.2 Installation Hierarchy

The Event Handler TaskBot itself will not generate any new folders. The following describe the key folders/files that are referenced by the Event Handler.

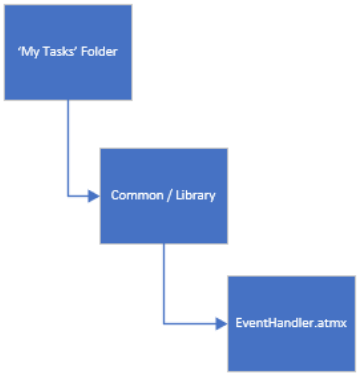
3.2.1 Process Log Folder (Shared Drive)

Per Automation Anywhere Bot Development Best Practices, the recommended location for log files (ex. audit log, error log) is generally on a shared drive (NAS). The Event Handler will access the 'Event Log' here to process any un-processed events, and the 'Error Log' here to write any errors it encounters during execution.

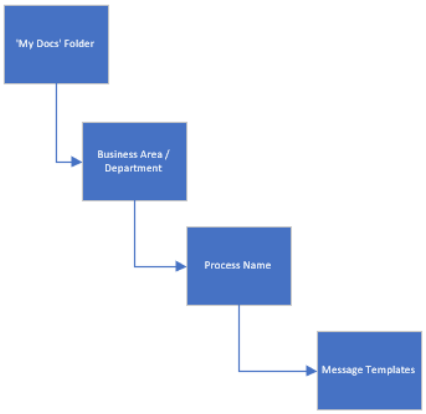


3.2.2 Bot Runner

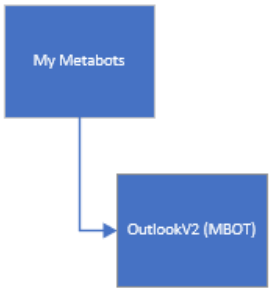
The Event Handler requires only one TaskBot (ATMX file). Generally it is recommended that these reusable TaskBots that will be used by multiple processes are stored in a folder named something like 'Library' or 'Common':



The HTML templates that are used for the body of event notification emails can be stored under My Docs in a path like the following:



Finally, if you use the Outlook Metabot for email communications and Outlook is available on the Bot Runner, then store that Metabot within the My Metabots path:



3.3 Quick Start

3.3.1 Setup

The Event Handler is designed to be reusable / shared, such that it can be used by Bots supporting any Use Case to recognize and action events (ie. send notifications).

Deploying the Event Handler to a Bot Runner involves the following steps:

- Configuring the Event Handler (EventHandler.XML file)
- Deploying the EventHandler TaskBot
- (OPTIONAL) Deploying the Outlook metabot
- Calling the EventHandler from another TaskBot

After executing these steps, the Event Handler TaskBot can be called via a 'Run Task' command, passing in the appropriate variables (see section variables)

3.3.2 Configuration

Configuration involves definition of 'event codes' in an EventHandler.xml file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <EventHandling>
  - <DEV>
      + <EVENT EventCode="ERROR_001" EventType="error">
      + <EVENT EventCode="ERROR_002" EventType="error">
      + <EVENT EventCode="ERROR_003" EventType="error">
      + <EVENT EventCode="EXCEPTION_001" EventType="exception">
      + <EVENT EventCode="EXCEPTION_002" EventType="exception">
      + <EVENT EventCode="EXCEPTION_003" EventType="exception">
      + <EVENT EventCode="EXCEPTION_004" EventType="exception">
      + <EVENT EventCode="EVENT_001" EventType="event">
      + <EVENT EventCode="EVENT_002" EventType="event">
      + <EVENT EventCode="EVENT_003" EventType="event">
    </DEV>
  + <UAT>
  + <PROD>
</EventHandling>
```

The EventHandling.XML file is organized by environment. In this way, the environment to which the Bot Runner is connected determines what actions/notifications are taking when that event occurs.

This enables, for example – determination that certain stakeholders should receive a notification when EVENT_001 occurs in Test, but a different stakeholder receives notification in Production.

See below for an expanded view of the EventHandling.XML. Let's look at the specific attributes within each 'environment' node, and their meaning.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <EventHandling>
  - <DEV>
    - <EVENT EventCode="ERROR_001" EventType="error">
        <Description>A general error occurred in TaskBotName1</Description>
        <EmailNotify>Yes</EmailNotify>
        <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
        <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
        <EmailRecipient>user@emaildomain.com</EmailRecipient>
        <EmailSubject>ERROR_001 - General Error in TaskBotName1</EmailSubject>
        <AttachFile>Yes</AttachFile>
        <EventHTMLTemplateFile>error_notification</EventHTMLTemplateFile>
        <TaskStatus>Fail</TaskStatus>
      </EVENT>
    + <EVENT EventCode="ERROR_002" EventType="error">
    + <EVENT EventCode="ERROR_003" EventType="error">
    - <EVENT EventCode="EXCEPTION_001" EventType="exception">
        <Description>A required folder/file could not be found</Description>
        <EmailNotify>Yes</EmailNotify>
        <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
        <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
        <EmailRecipient>user@emaildomain.com</EmailRecipient>
        <EmailSubject>EXCEPTION_001 - Missing Folder/File</EmailSubject>
        <AttachFile>Yes</AttachFile>
        <EventHTMLTemplateFile>system_exception_notification</EventHTMLTemplateFile>
        <TaskStatus>Fail</TaskStatus>
      </EVENT>
    + <EVENT EventCode="EXCEPTION_002" EventType="exception">
    + <EVENT EventCode="EXCEPTION_003" EventType="exception">
    + <EVENT EventCode="EXCEPTION_004" EventType="exception">
    - <EVENT EventCode="EVENT_001" EventType="event">
        <Description>Process Completed Successfully</Description>
        <EmailNotify>Yes</EmailNotify>
        <EmailRecipient>user@emaildomain.com</EmailRecipient>
        <EmailCCRecipient>user@emaildomain.com</EmailCCRecipient>
        <EmailBCCRecipient>user@emaildomain.com</EmailBCCRecipient>
        <EmailSubject>Process Completed Successfully</EmailSubject>
        <AttachFile>Yes</AttachFile>
        <EventHTMLTemplateFile>standard_event_notification</EventHTMLTemplateFile>
        <TaskStatus>Pass</TaskStatus>
      </EVENT>
    + <EVENT EventCode="EVENT_002" EventType="event">
    + <EVENT EventCode="EVENT_003" EventType="event">
  </DEV>
  + <UAT>
  + <PROD>
</EventHandling>
```

Each EVENT element has two attributes:

1. **Event Type** = error, exception, event
2. **EventCode** = a unique identifier for the event. Recommended naming is a prefix for the type – ex. ERROR_001, EXCEPTION_001, EVENT_001, and that the event code be unique across all **TaskBots** in the automation.

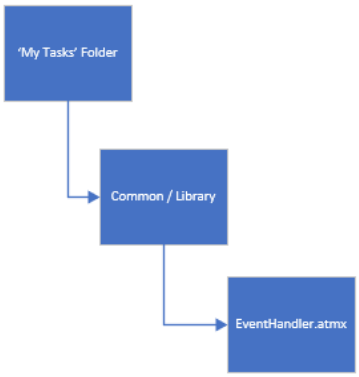
Note: that this naming standard for event codes is useful to enable subsequent analysis on the specific events that are being generated across the Bots being executed.

The following are the child elements contained within each child 'EVENT' element:

Element	Description	Example
Description	A general description for this event.	A require folder
EmailNotify	Should an email notification be sent if this event occurs?	Yes No
EmailCCRecipient	The CC recipient of email communication	user@domain.com
EmailBCCRecipient	The BCC recipient of email communication	user@domain.com
EmailSubject	Subject of email notification of event	"Event Notification"
AttachFile	Is there a file that should be included as an attachment to the email communication?	Yes No The path to the specific file to be attached is passed via $xEventAttachmentFile$
EventHTMLTemplateFile	An .html or .htm file that defines the HTML template for the event notification. The file has special placeholders that will be replaced with dynamic information (See Use of Outlook Metabot). The folder $xNotificationHTMLTemplateFolder$ should house these files.	

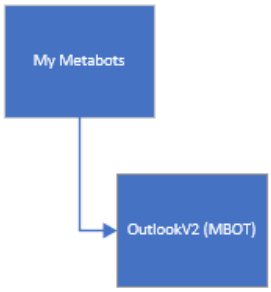
3.3.3 Deploying the Event Handler

The Event Handler requires only one TaskBot (ATMX file). Generally it is recommended that these reusable TaskBots that will be used by multiple processes are stored in a folder named something like 'Library' or 'Common':

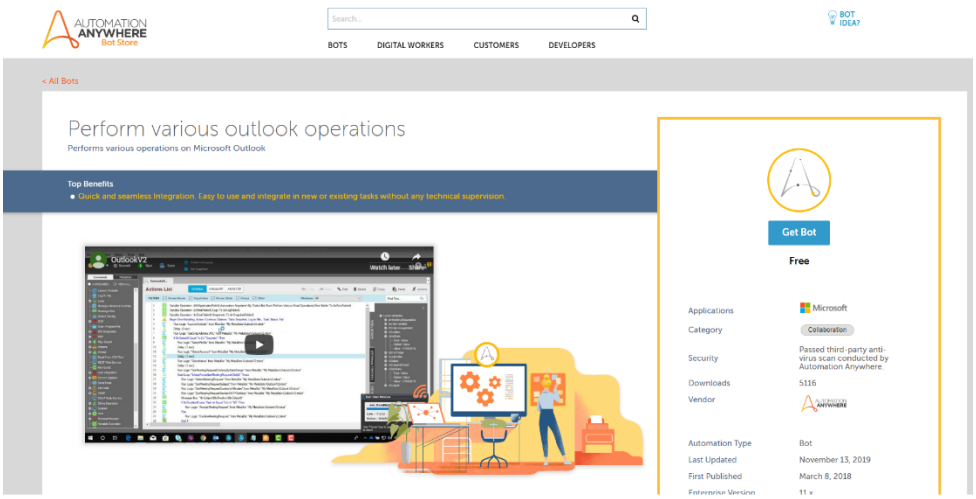


3.3.4 (OPTIONAL) Deploying the Outlook metabot

If you want to use the Outlook metabot ("Perform Various Outlook Operations") to send email communications, ensure the appropriate Outlookv2.mbot file is deployed to this directory:



For more information about this Metabot, consult the documentation on the Bot Store.



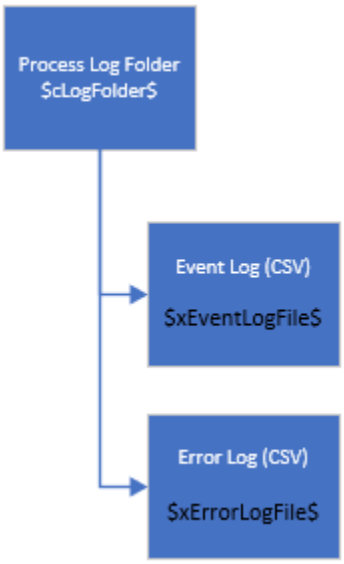
3.3.5 Calling the Event Handler From Another TaskBot

To raise an event, these steps can be implemented in a TaskBot that is part of a process automation:

1. Log the occurrence of the event to an EventLog.csv file, referencing the appropriate Event Code.
2. Call the Event Handler ATMX file, with the required input variables

Note: the EventHandler expects both the Error Log and Event Log files to be located in $cLogFolder$ (passed to the EventHandler as a variable), using the following variable names:

- $xErrorLogFile$ - ex. $cLogFolder$\ErrorLog_$Year$_$Month$_Day.csv
- $xEventLogFile$ - ex. $cLogFolder$\EventLog_$Year$_$Month$_Day.csv



The Event Handler will read the EventLog.csv, lookup the Event Code in the EventHandler.XML file, and issue an appropriate notification based on the Event Code configuration, and the environment to which the Bot Runner is connected.

The following shows sample content from the Event Log.CSV file:

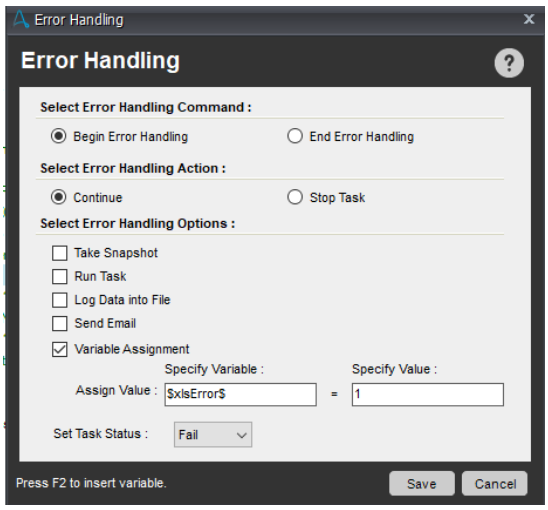
BOT RUNNER	CALLING TASK BOT	EVENT CODE	(ERROR) LINE NUMBER	DESCRIPTION	EVENT TYPE	TIMESTAMP	EVENT HANDLED?
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	EVENT_004		Process Completed Successfully	Event	10/15/2019 8:27	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	EVENT_004		Process Completed Successfully	Event	10/15/2019 8:41	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	ERROR_012	52	The process cannot access the file 'c:\temp\file.xml'	Error	10/15/2019 8:57	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	EVENT_004		Process Completed Successfully	Event	10/15/2019 9:37	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	EVENT_001		Process Completed Successfully	Event	10/15/2019 13:04	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_ExecuteCodeValidation	EVENT_001		Process Completed Successfully	Event	10/15/2019 13:23	Y
AA-TX-BotRunner1 - bot_dev	SUBTASK_LogIssueToExcel	EXCEPTION_012		Unsuccessful logging in to the system.	Exception	10/15/2019 13:45	Y

***Note**: The 'header' row here is normally not present, but was added as an aide to explain the content of this file.*

3.3.5.1 Example 1 – Raising an 'Error'

The following example demonstrates raising an Event of the 'Error' type, from a calling Task Bot. Note that errors are the result of 'Error Handling' blocks. In those Error Handling blocks, we can set a variable to indicate that the error occurred; after the Error Handling block, the 'Error' event can be raised accordingly.

Of the (3) Event Types, 'error' is unique in that it will have an event description ($ErrorDescription$) and line number ($ErrorLineNumber$).




```
Comment: **********
Comment: ********** HANDLE (UNCAUGHT) 'ERRORS' FROM ERROR HANDLING BLOCKS **********
Comment: **********
If $xlsError$ Not Equal To (<>) "0" Then
    Variable Operation: $Error Description$ To $xErrorDescription$
    Variable Operation: $Error Line Number$ To $xErrorLineNumber$
    Log to File: $Date$,$Machine$,$System(USERNAME)$,$vTaskName$,$Error Line Number$,"$Error Description$" in "$xErrorLogFile$"
    Log to File: $Machine$ - $AATaskExecutor(Executor_UserName)$,$vTaskName$,ERROR_003, $Error Line Number$,$Error Description$, Error,$Date$, ,N in "$xEventLogFile$"
    Run Task "$AAApplicationPath$\Automation Anywhere\My Tasks\Common\EventHandler.atmx" @Repeat: Do Not Repeat @Speed: High Speed @Pass Variable as argument: $cDefaultRPASe
    Stop The Current Task
End If
```

NOTE: in the above example, we write to the Event Log so that the Event Handler can process the event. Additionally, we (optionally) write to the Error Log.

3.3.5.2 Example 2 – Raising an 'Exception'

The following example demonstrates raising an Event of the 'Exception' type, from a calling Task Bot.

Note that for exceptions, we have code that explicitly checks for a condition to identify the exception. In the below case, the 'exception' is that a window that we expected to exist, does not exist:

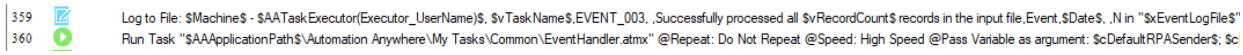
```
110 If Window Does Not Exist ("*Bank Reconciliation System*") Then  (Wait up to 1 seconds - for Window not to exist)
111     Comment: Assume a performance issue - may have been slow to open screen after logging in.  Wait, then check again.
112     Log to File: $Machine$ - $AATaskExecutor(Executor_UserName)$,$vTaskName$, EXCEPTION_009, ,Bank Rec system not available during processing - trying again in a few seconds,Exception,$Date$, ,N in "$xEventLogFile$"
113     Run Task "$AAApplicationPath$\Automation Anywhere\My Tasks\Common\EventHandler.atmx" @Repeat: Do Not Repeat @Speed: High Speed @Pass Variable as argument: $cDefaultRPASender$; $cDefaultRPASupportEma
```

For ‘Exception’ events, line number in the Event Log is generally blank, and the ‘event description’ written to the Event Log can be considered optional if further description beyond the ‘Event Code’ description is helpful.

3.3.5.3 Example 3 – Raising an ‘Event’

The following example demonstrates raising an Event of the ‘Event’ type, from a calling Task Bot:

Note that events, like exceptions, are embedded within code at the appropriate stage where the ‘event’ occurs – for example, at the end of the code to raise an event representing the over-all outcome of the execution.



For ‘Events’, line number in the Event Log is generally blank, and the ‘event description’ written to the Event Log can be considered optional if further description beyond the ‘Event Code’ description is helpful.

3.3.6 Input Variables

The 'calling' TaskBot should pass in the following variable values:

Variable	Description	Example
xEventLogFile	Path to the Event Log CSV file.	<NASPATH>\Business Area\Process Name\Logs\EventLog_DATE.csv
xErrorLogFile	Path to the Error Log CSV file. Note if the EventHandler does not work, we will write *only* to this Error Log, and attempt to send an email using the default 'Send Mail' command.	<NASPATH>\Business Area\Process Name\Logs\ErrorLog_DATE.csv
xBusinessFunctionalArea	The functional area/grouping for this process name (ex. department)	• Finance • HumanResources • IT
xBusinessProcess	Name of the business process for this automation.	AccountsPayable
xEnvironment	The Automation Anywhere/Control Room environment to which the Bot Runner is connected. Used to reference the appropriate 'Event' description in Event Handling.XML file.	Dev UAT Prod
xEventAttachmentFile	Path to an attachment that will be included with the event notification.	$AAApplicationPath$\Automation Anywhere\My Docs\$xBusinessFunctionalArea$\$xBusinessProcess$\theFile.jpg
$xNotificationHTMLTemplateFolder$		$AAApplicationPath$\Automation Anywhere\My Docs\$xBusinessFunctionalArea$\$xBusinessProcess$\EmailMessageTemplates
cLogFolder	The folder where log files (ex. the event log CSV file) are stored	<NASPATH>\Business Area\Process Name\Logs
cDefaultRPASender	From what email address should email notifications be sent?	user@domain.com
cDefaultRPASupportEmail	To whom should email notifications be sent, if not found by looking up the event code? Ie. Default if the event code referenced in the Event Log. CSV file cannot be located in EventHandling.xml.	user@domain.com

Note: In the above variables, a ‘c’ prefix indicates that their value is sourced from the process ‘config’ file; an ‘x’ prefix indicates they are defined and passed from a calling TaskBot.

3.3.7 Output Variables

The only variable intended to be passed to the calling Task is $xIsError$, a flag for which a value of ‘1’ indicates that an error has occurred.

Note: Great care should be taken in the calling Task in handling situations where the Event Handler returns a value of ‘1’ for $xIsError$. For example, it may not be appropriate for the calling Task to stop processing just because the Event Handler failed to issue event notifications.

4. Reports

The 'Event Log' can be used in powerful ways to drive insight around the errors, exceptions, and other events that are logged by a process.

The Event Log for a specific process/Use Case could be analyzed, or Event Logs across Use Cases could be consolidated and analyzed holistically.

In all cases, the following are key data elements on which analysis can be performed:

- **Event Type** – was the event an Error, Exception, or (other) Event?
- **Event (Code) Description** – what event occurred? Ex. "System not available after login", "A required folder did not exist", or "Business rule for ABC failed"
- **Event Description** – what are the details for what occurred? Ex. "Folder 'input' was not found at location <path>"
- **Timestamp** – when did the event occur?

It may be noted that there are actually multiple 'descriptions' associated with an event. This is 'by design' to provide multiple 'levels' of event description, for maximum utilization in both event communication, and subsequent analysis:

Event (Code) Description – in EventHandling.xml, each Event Code has a description in the <Description> tag. This should be a high-level/generic description for the event code.

Event Description – when the Calling Task logs an event in the Event Log, a more specific/detailed description can be provided.

Examples of events logged in an Event Log:

Event Code	Event Type	Event (Code) Description (ie. Description of Event Code in EventHandling.xml)	Event (Detailed) Description (ie. Description entered in EventLog)
EXCEPTION_032	Exception	"A required folder/file was not found"	"Could not find Config.XML in <path>"
			"Could not find Input Folder <path>"
ERROR_001	Error	"A general error occurred in TaskBot <TaskBotName>"	$ErrorDescription$ occurred at line $ErrorLineNumber$
EVENT_001	Event	"Process completed successfully"	"15 Records successfully processed"
			"10 Records successfully processed, 5 records were skipped as were previously processed"
EVENT_002	Event	"Process completed with failures"	"5 Records processed; 3 additional records failed"
			"5 Records processed; 3 additional records failed"

5. Logs

(2) logs are used by the Event Handler – in both cases, information is initially inserted to the logs by the process that calls the Event Handler:

1. Event Log – this log should be appended by a calling Task as 'events' occur during execution. The Event Handler will 'read' this log, and action the events (ie. send notifications) as defined bv the configuration.
2. Error Log – this log, updated by the process using the Event Handler to log errors, will also be used by the Event Handler should errors occur during execution of the Event Handler.

6. Troubleshooting & Support

6.1 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*	*Nov 21st, 2019*	*AA Professional Services*	*Initial Submission to Bot Store*

Appendix B: Acronyms

No.	Acronym	Description

Appendix C: References

No.	Topic	Reference Link
1	"Perform Various Outlook Operations"	Click here