

Course Exercises Guide

Automating Tasks Using IBM Robotic Process Automation with Automation Anywhere

Course code WB502 / ZB502 ERC 3.0



March 2019 edition

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

© Copyright International Business Machines Corporation 2018, 2019.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	viii
Exercises description	ix
Exercise 1. Exploring the Control Room	1-1
1.1. Introducing the exercise case study	1-2
1.2. Exploring the Control Room	1-3
Section 1: Exploring the Dashboards	1-3
Section 2: Exploring the Administrative Features	1-9
Section 3: Create a development user	1-16
Section 4: Confirming the development user account	1-19
Exercise 2. Creating a basic bot	2-1
Section 1. Exploring the Enterprise Client	2-2
1.1. Signing in to the Enterprise Client	2-2
1.2. Exploring the Enterprise Client	2-4
1.3. Exploring the Workbench	2-6
1.4. Working with the Workbench: Adding commands	2-8
1.5. Editing commands	2-13
1.6. Deleting commands	2-16
Section 2. Automating the creation and saving of a text file	2-18
2.1. Automating the opening of Notepad	2-18
2.2. Automating the writing of text to Notepad	2-19
2.3. Automating the saving of the text file	2-20
2.4. Running the bot	2-25
Exercise 3. Writing data from a text file to an Excel spreadsheet	3-1
Section 1. Getting started with the Workbench	3-3
Section 2. Working with bot variables	3-4
2.1. Exploring the Variable Manager	3-5
2.2. Creating local variables to hold the tab-delimited text data	3-7
2.3. Defining local variables to hold transaction data	3-9
2.4. Adding a counter variable to track spreadsheet rows	3-9
2.5. Saving your bot	3-10
Section 3. Creating the beginning bot actions	3-11
3.1. Adding a comment	3-11
3.2. Automate opening an Excel spreadsheet	3-12
3.3. Resizing and moving the Excel window	3-14
Section 4. Working with loops	3-17
4.1. Defining a loop to read from a text file	3-17
4.2. Assigning the extracted data to variables	3-18
4.3. Testing the bot	3-22
Section 5. Defining loop actions: Extracting substrings and assigning them to variables	3-25
5.1. Extracting substrings	3-25
5.2. Testing the bot	3-27
Section 6. Defining loop actions: Writing data to the transactions.xlsx Excel spreadsheet	3-30
Section 7. Finishing and running the bot	3-33
Section 8. Optional: Viewing the solution file	3-37

Exercise 4. Automating data entry to a business application and a database	4-1
Section 1. Defining custom variables for customer data	4-3
1.1. Creating a task	4-3
1.2. Creating custom variables	4-4
Section 2. Opening the spreadsheet and connecting to the database	4-7
2.1. Opening the spreadsheet	4-7
2.2. Connecting to the database	4-10
Section 3. Opening the SIB CRM application	4-12
3.1. Testing the bot	4-14
Section 4. Using a loop to perform repetitive tasks	4-15
4.1. Setting up the loop	4-15
4.2. Using the Smart Recorder to automate data entry tasks	4-17
4.3. Using the Smart Recorder to submit the form and using a string operation to extract the reference number from a message window	4-23
4.4. Storing the customer reference number in the new_customers.xlsx spreadsheet	4-27
4.5. Testing your bot	4-28
4.6. Inserting customer data into a database	4-30
Section 5. Finishing and running the bot	4-31
5.1. Completing the bot	4-31
5.2. Running the bot	4-32
5.3. Confirming the database INSERT operations	4-33
Section 6. <i>Optional:</i> Viewing the solution file	4-38

Exercise 5. Creating a bot to sum check declines, query a database, and send an email	5-1
Section 1. Creating a bot and adding custom variables	5-2
1.1. Creating a task	5-2
1.2. Creating the value variables for the bot	5-2
1.3. Creating the array variables	5-5
1.4. Creating list variables	5-6
1.5. Saving your work	5-7
Section 2. Extracting values from a file and calculating totals	5-8
2.1. Automating extraction of values from a CSV file	5-8
2.2. Calculating a running total	5-9
2.3. Testing the bot	5-16
Section 3. Extracting information from a database	5-19
3.1. Automating the opening of a text editor	5-19
3.2. Connecting to a database	5-20
3.3. Running the bot	5-25
Section 4. Sending results by email	5-27
4.1. Copying text to the Clipboard	5-27
4.2. Simulating “Don’t save” in Notepad	5-27
4.3. Sending email to Margin Clerk with results	5-29
Section 5. Configuring the trigger	5-32
Section 6. Running the bot and confirming email delivery	5-35
Section 7. <i>Optional:</i> Viewing the solution file	5-38

Exercise 6. Creating a bot to evaluate data from a PDF and send an email	6-1
Section 1. Creating a bot and adding custom variables	6-3
1.1. Creating a task	6-3
1.2. Creating the variables for the bot	6-3
1.3. Saving your work	6-4
Section 2. Extracting trade receipts from email and extracting values from PDF	6-5
2.1. Adding comments to your code	6-5
2.2. Iterating through emails	6-5
2.3. Extracting information from a PDF form	6-7
Section 3. Trimming variables and determining length	6-12

3.1. Trimming variables	6-12
3.2. Determining the length of each variable	6-13
3.3. Preparing your environment to test the bot	6-15
3.4. Configuring the Workbench for debugging	6-17
Section 4. Evaluating data and sending email responses	6-20
4.1. Evaluating the length of variables	6-20
4.2. Creating an email response for missing data	6-22
4.3. Creating an email response when all the data is present	6-24
Section 5. Automating clean-up of emails and files	6-25
Section 6. Configuring a trigger	6-27
Section 7. Running the bot	6-29
7.1. Testing the email response when all data is present	6-30
7.2. Testing the email response when all data is missing	6-32
Section 8. <i>Optional.</i> Viewing the solution file	6-35

Exercise 7. Creating an interactive bot to check values in disparate systems 7-1

Section 1. Starting a new task and defining local variables	7-3
Section 2. Creating an interactive prompt and checking information from a web page	7-4
2.1. Creating an interactive prompt	7-4
2.2. Checking the customer list web page for account information	7-4
2.3. Notifying the user that the account is not found	7-12
2.4. Notifying the user that the account is found	7-14
2.5. Testing the bot	7-15
Section 3. Checking information from a Microsoft Access-based Windows application	7-17
3.1. Opening the Microsoft Access application and logging in	7-17
3.2. Searching for the account number and retrieving the Company field value	7-21
3.3. Adding vCompany2 to the Message Box	7-33
3.4. Testing the bot	7-33
Section 4. Extracting the company value from an Excel spreadsheet	7-35
4.1. Opening the Excel spreadsheet and searching for the account number	7-35
4.2. Finding the company value and assigning it to the vCompany3 variable	7-37
4.3. Adding vCompany3 to Message Box	7-40
4.4. Testing the bot	7-41
Section 5. <i>Optional.</i> Viewing the solution file	7-42

Exercise 8. Creating a login MetaBot 8-1

Section 1. Building the MetaBot	8-2
1.1. Opening the “start” file for the bot	8-2
1.2. Starting the Customer Manager Access database	8-2
1.3. Writing the MetaBot code	8-3
1.4. Defining the login logic for the MetaBot	8-6
1.5. Testing the logic behavior	8-9
Section 2. Replacing login commands with the login MetaBot	8-11
2.1. Disabling the login commands in the bot code	8-11
2.2. Replacing the logic commands with the MetaBot	8-11
2.3. Running the bot	8-13

Exercise 9. Working with web services and error handling 9-1

Section 1. Creating a copy of the Trade Booking bot and adding custom variables	9-3
1.1. Opening the “start” file for the bot	9-3
1.2. Creating user variables	9-3
Section 2. Adding error handling to the Advanced Trade Booking bot	9-6
Section 3. Working with REST web service calls	9-9
3.1. Understanding the REST Web Service command	9-9
3.2. Adding the REST Web Service command	9-11
3.3. Constructing the error message from the REST response	9-15

3.4. Extracting the Quandl error code and error message	9-17
3.5. Removing unneeded data from the stock return message	9-19
Section 4. Evaluating whether the REST response has an error	9-22
4.1. Configuring the email response to a REST error	9-24
Section 5. Working with the Quandl stock price data	9-27
Section 6. Evaluating the length of decimal values in the stock price	9-31
Section 7. Evaluating decimal values and concatenating zeros when needed	9-35
7.1. Viewing the completed bot	9-39
Section 8. Running the bot	9-40
8.1. Scenario 1: Verifying that all information is present	9-41
8.2. Scenario 2: Testing an alternative stock symbol	9-43
8.3. Scenario 3: Handling a REST error	9-44
8.4. Scenario 4: Handling a general bot error	9-46
Section 9. <i>Optional.</i> Viewing the solution file	9-48
Exercise 10. Hardening the Account Opening bot	10-1
Section 1. Preparing to code bots	10-5
1.1. Opening the “start” file for the bot	10-5
1.2. Creating a bot file	10-5
Section 2. Preparing the bot hardening code	10-7
2.1. Spreadsheet scenario 1: Renaming the spreadsheet extension	10-7
2.2. Getting the location of the Customer ID column	10-8
2.3. Moving the Customer ID column	10-10
2.4. Preparing the spreadsheet file for processing	10-11
2.5. Using the Smart Recorder to select and move the Column ID column	10-12
2.6. Copying the Excel bot code into the Account Opening with Hardening bot	10-16
2.7. <i>Optional.</i> Testing the bot	10-17
Section 3. Preparing the error handling code	10-18
Section 4. Building the error handling bot	10-21
4.1. Creating a bot and defining local variables	10-21
4.2. Opening the log file and copying log entries	10-21
4.3. Extracting the SQL error code and the bot name	10-22
Section 5. Responding to errors	10-26
5.1. Responding to the SQL 08001 error	10-26
5.2. Responding to the SQL 22001 error	10-27
5.3. Responding to all other errors	10-29
Section 6. Configuring the trigger for the Handle Errors bot	10-31
Section 1: Excel Scenarios	10-33
Section 2: Handling error scenarios	10-34
Exercise 11. Managing bots	11-1
Section 1. Preparing to code bots	11-2
1.1. Making a copy of original Data Consistency bot	11-2
1.2. Creating the client bot that passes a variable to the server bot	11-2
Section 2. Building code for client bot	11-3
2.1. Creating bot dependency	11-4
Section 3. Uploading Server bot for code comparison	11-5
3.1. Uploading the bot to Server Repository	11-5
3.2. Comparing code between the Client and Server bots	11-5
3.3. Running the new bot	11-8
Section 4. Reviewing notifications and system logs	11-10
4.1. Viewing notifications	11-10
4.2. Viewing system logs	11-10
Section 5. Creating a report in Report Designer	11-12

Exercise 12. Administering bots in the Control Room	12-1
Section 1. Uploading bots to the Control Room	12-2
1.1. Uploading bots to the Control Room repository	12-2
1.2. Preparing the environment and logging in to the Enterprise Client as a runtime user	12-4
Section 2. Running bots from the Control Room	12-8
2.1. Run bot now from the Control Room	12-8
2.2. Run bot now from the Control Room - leave running	12-11
2.3. Schedule bot to run later	12-11
2.4. Run bot with queue	12-12
Section 3. Viewing bot status	12-18
3.1. Viewing bot status	12-18

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DB™
Insight®
400®

FileNet®
Notes®

Global Business Services®
Tivoli®

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

VMware is a registered trademark or trademark of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Exercises description

This course includes the following exercises:

- [Exercise 1, "Exploring the Control Room"](#)
- [Exercise 2, "Creating a basic bot"](#)
- [Exercise 3, "Writing data from a text file to an Excel spreadsheet"](#)
- [Exercise 4, "Automating data entry to a business application and a database"](#)
- [Exercise 5, "Creating a bot to sum check declines, query a database, and send an email"](#)
- [Exercise 6, "Creating a bot to evaluate data from a PDF and send an email"](#)
- [Exercise 7, "Creating an interactive bot to check values in disparate systems"](#)
- [Exercise 8, "Creating a login MetaBot"](#)
- [Exercise , "Working with web services and error handling"](#)
- [Exercise 10, "Hardening the Account Opening bot"](#)
- [Exercise 11, "Managing bots"](#)
- [Exercise 12, "Administering bots in the Control Room"](#)

General exercise instructions

This section provides general information about the exercises in this course. Review this section before starting the exercises.



Important

The exercises in this course use a set of lab files that are installed in the following directory:

C:\labfiles

The exercises point you to the lab files as you need them.



Stop

Course updates and corrections



A course corrections document might be available for this course.

If you are taking the class with an instructor, the instructor can provide this document to you.

If you are taking the course in a self-paced environment, the course corrections document is provided with the other manuals.

To check whether a course corrections document exists for this course:

1. Go to the following URL: <http://www.ibm.biz/CloudEduCourses>
2. Find the product category for the course and click the link.
3. Find your course in the list and click the link.
4. Click the **Attachments** tab to see whether a course corrections document exists with updated instructions.
5. To save the file to your computer, click the document link and follow the prompts.

User accounts and passwords

Table 1. User accounts and passwords

Type	User name	Password
Course VMware image	Administrator	passw0rd
Microsoft Windows Server 2012	Administrator	passw0rd
IBM Robotic Process Automation with Automation Anywhere Version 10.0 Windows 64-bit English eAssembly administrative super user	rpaAdmin	rpaAdmin
IBM Robotic Process Automation with Automation Anywhere BotCreator (bot developer) user	devUser1	devUser1
hMailServer administrator account		admin
SIB IBM Robotic Process Automation with Automation Anywhere administrator email account (rpaAdmin@sib.com)	rpaAdmin	rpaAdmin
SIB bot developer email account (devUser1@sib.com)	devUser1	devUser1

Table 1. User accounts and passwords

Type	User name	Password
SIB bot developer email account for Exercise 1 (myDevUser@sib.com)	myDevUser	myDevUser
SIB bot runtime user (botUser1@sib.com)	botUser1	botUser1
SIB administrative clerk email account (adminClerk@sib.com)	adminClerk	adminClerk
SIB margin clerk email account (marginClerk@sib.com)	marginClerk	marginClerk
SIB trade reporter email account (tradeReporter@sib.com)	tradeReporter	tradeReporter
SIB trade reporter supervisor email account (tradeSupervisor@sib.com)	tradeSupervisor	tradeSupervisor
SIB trade distribution list email account (tradeList@sib.com)	tradeList	tradeList
SIB support desk email account (support@sib.com)	support	support
SIB taskbot email account (taskbot@sib.com)	taskbot	taskbot
myBank trader1 clerk email account (trader1@myBank.com)	trader1	trader1
DB2 administrator	student	Education1

Exercise structure

Each exercise is divided into sections with a series of numbered steps and lettered substeps:

- The numbered steps (1, 2, 3) represent actions to be done.
- The lettered substeps (a, b, c) provide detailed guidance on how to complete the action.



Information

If you already understand how to do the action in the numbered step, you can skip the specific guidance in the lettered substeps.

The following example comes from Exercise 1 of this course.



Example

Excerpt from Exercise 1

- 1. Edit the rule and change the debt-to-income ratio from 0.3 to 0.5.
 - a. Click **Edit** to open the rule editor.

__ b. In the **if** part of the rule, change 0.3 to: 0.5

In this example, the numbered instructions prompt you to edit a rule. The “a” and “b” substeps provide specific guidance on how to edit the rule.

Text highlighting in exercises

Different text styles indicate various elements in the exercises.

Words that are highlighted in **bold** represent GUI items that you interact with, such as:

- Menu items
- Field names
- Icons

Words that are highlighted with a `fixed font` include the following items:

- Text that you type or enter as a value
- System messages
- Directory paths
- Code

Tracking your progress

As shown in the example step, you can see that an underscore precedes each numbered step and lettered substep.

You are encouraged to use these markers to track your progress by checking off each step as you complete it. Tracking your progress in this manner might be useful if you are interrupted while working on an exercise.

Required exercise sections

Most exercises include required sections that should always be completed. It might be necessary to complete these sections before you can start subsequent exercises.

Dependencies between exercises are listed in the exercise introduction.

Optional exercise sections

Some exercises might also include optional sections that you can complete when you have sufficient time and want an extra challenge.

Exercise 1. Exploring the Control Room

Estimated time

00:30

Overview

In this exercise, you are introduced to basic features of the Control Room and learn how to start and navigate the different IBM Robotic Process Automation with Automation Anywhere components.

Objectives

After completing this exercise, you should be able to:

- Describe the features of the Control Room.
- Create an IBM Robotic Process Automation with Automation Anywhere user and allocate an appropriate license for that user.

Introduction

The Control Room is a web-based administrative dashboard where users can view information about the operational status of bots, work with centrally managed credentials, automation schedules, and bots. In addition, with the correct permissions, administrative users can create user security roles and user accounts in the Control Room and manage user licenses.

Requirements

This exercise requires the computer lab environment that was developed for this course.

1.1. Introducing the exercise case study

The fictitious case study that underlies the various exercise scenarios is focused on a small investment bank, Smarter Investment Bank (SIB). SIB recently acquired MyBank, another small financial services firm, but their systems are not integrated yet. SIB wants to use IBM Robotic Process Automation with Automation Anywhere to automate tasks as the system integration process goes forward.

The case study starts at the point where the automation team at SIB completed the project planning phase, and the SIB IT team is now ready to start developing bots. The automation team identified the following high-priority processes, which have tasks that are suitable for automation:

2. Collecting MyBank transaction data
3. Opening an account
4. Summing declined checks and emailing a Margin Clerk
5. Booking share trades
6. Ensuring that account data is consistent across multiple systems
7. Evaluating data from a PDF and sending an email

Throughout the exercises in this course, you take on different user roles, such as administrator and bot creator (developer), to work with IBM Robotic Process Automation with Automation Anywhere to create and manage users and bots.

The exercise case study is also described at the end of Unit 4.

1.2. Exploring the Control Room

The IBM Robotic Process Automation with Automation Anywhere Control Room is a central tool for managing users and bots. In this section of this exercise, you log into the Control Room as an administrator and explore some of its features.



Important

In this exercise, you do not view all of the features of the Control Room. This exercise explores features that focus on running and managing deployed bots, and some administrative features, such as user management.

Other features, such as security roles and client management, are not explored in this exercise.

Section 1: Exploring the Dashboards

- ___ 1. Start and log in to the Control Room as an administrator.
 - ___ a. Double-click the **Automation Anywhere Enterprise Control Room** desktop shortcut. It might take a couple of minutes for the login page to load.



Automation
Anywhere
Control
Room

- ___ b. In the Control Room Login page, enter the following credentials. The password is case-sensitive.

Username: rpaAdmin

Password: rpaAdmin

- ___ c. Click **Login**.

Log in

Username
rpaAdmin

Password

Remember my username

- ___ d. If you are prompted by Internet Explorer to choose security settings, click **Use recommended security and compatibility settings** and click **OK**

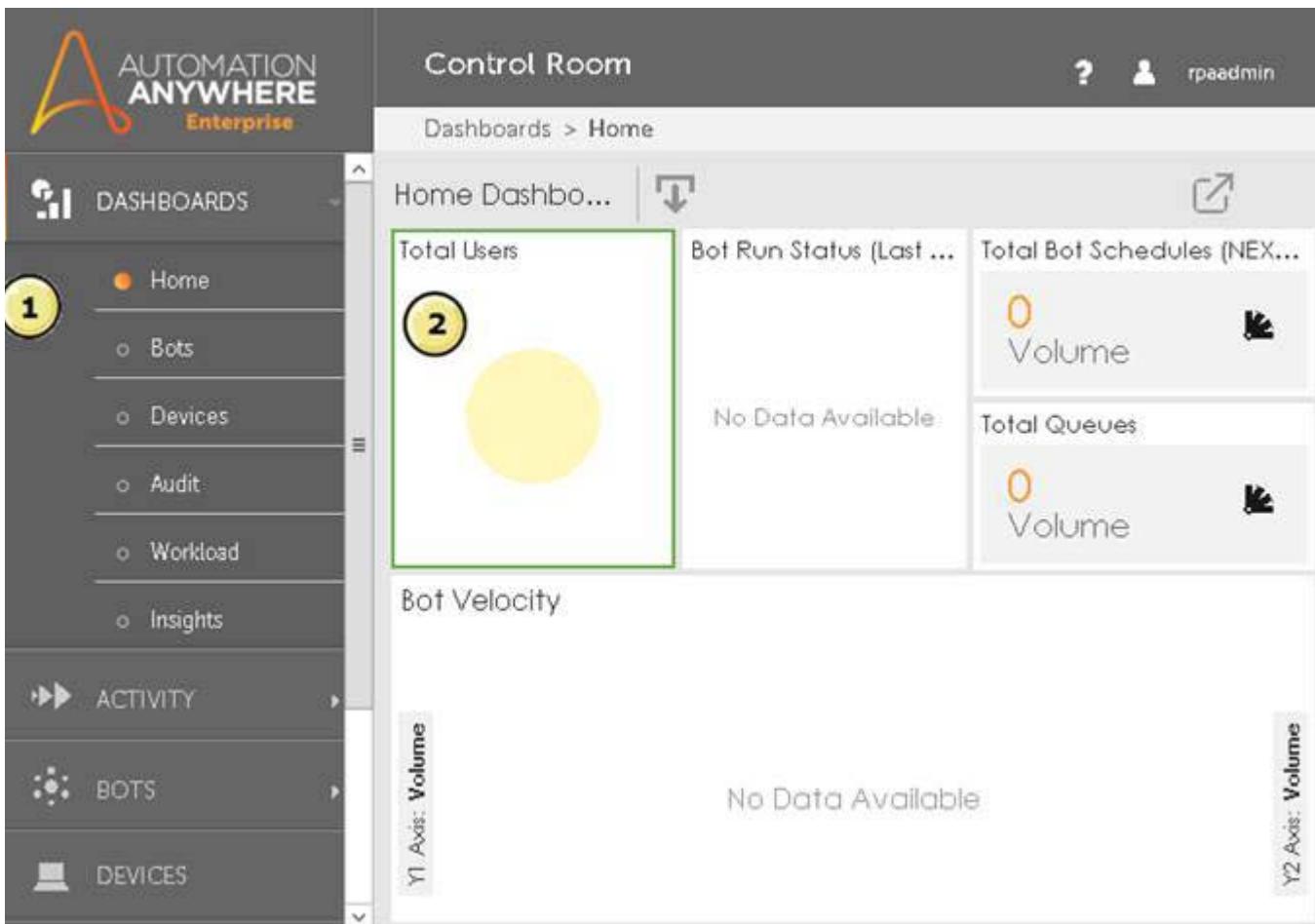


The Control Room features that a user can access depend on the permissions of the specific user role that they were assigned to by the IBM Robotic Process Automation with Automation Anywhere administrator.

Because you logged in to the Control Room as the main administrator user that was created when IBM Robotic Process Automation with Automation Anywhere was installed, you can access all the Control Room features.

You learn more about creating a user and assigning roles later in the course.

When you are logged in to the Control Room, you will see the Home Dashboard first. In the Control Room, you can use the left pane (1) to access different Control Room features. You use the main pane (2) to view information and configure settings.



Information

By default, the Control Room logs out a user after 20 minutes of inactivity.



Troubleshooting

If you have login time out problems with the Control Room, use Chrome as the web browser:

- Open a Chrome web browser window.
- Copy the URL from the Control Room window and paste it into Chrome.
 - The Control Room URL is: `vclassbase:81`.
- Log in by entering `rpaAdmin` in the **Username** and **Password** fields...

2. View the Home Dashboard.

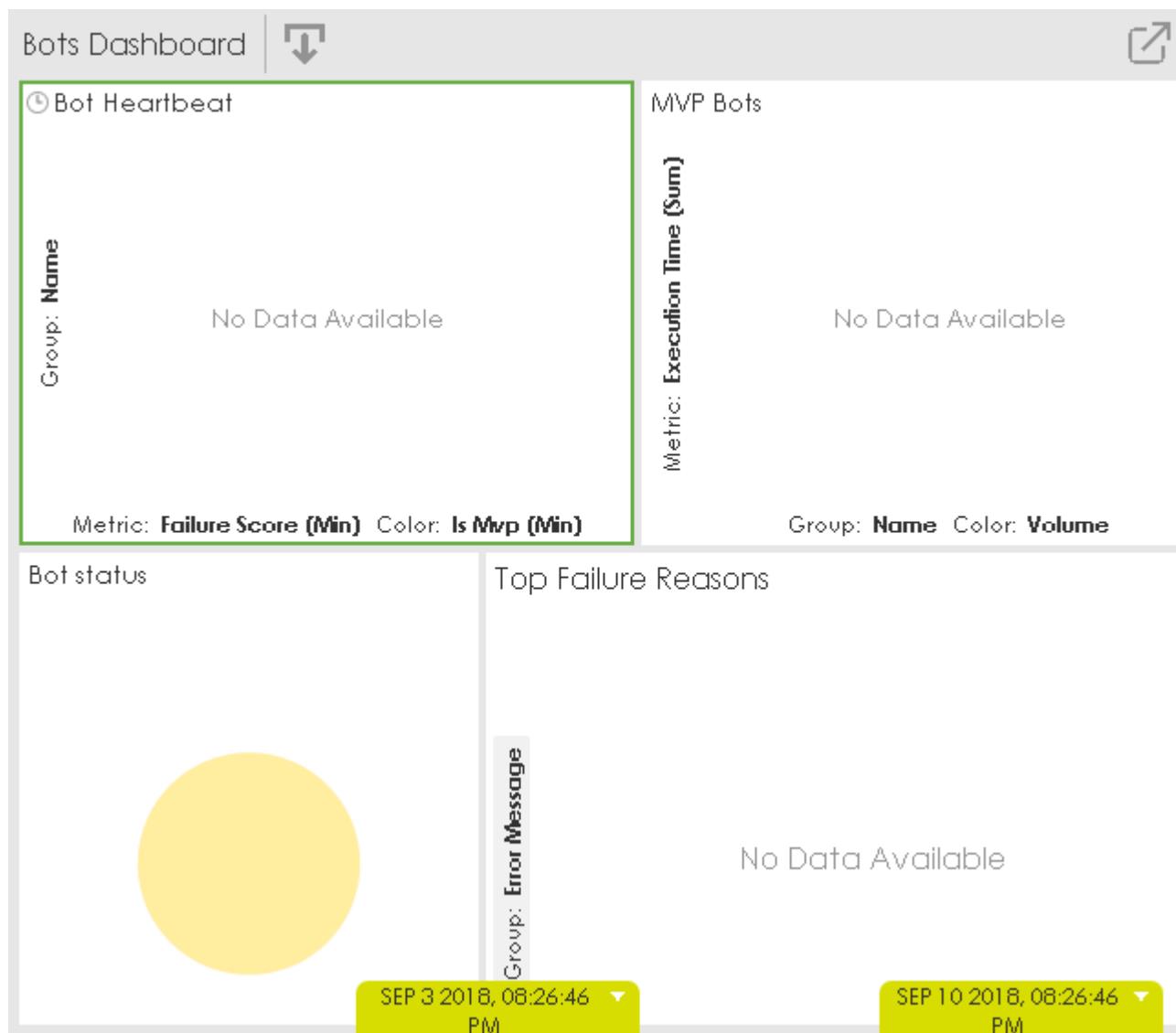
The Home Dashboard provides several high-level overviews of activities that are related to bots, and a summary of contents in the repository. You can also view information about bot run state and the number of active users.

Because you logged in as the main administrator, the Control Room opens on the Home Dashboard by default. The Home Dashboard is available only to the main Control Room administrator. The Home Dashboard provides the following information:

- Total Users
- Bot Run Status
- Total Bot Schedules
- Total Queues
- Bot Velocity
- Capacity Utilization: Bots vs. Bot Runners (may need to scroll to view)

— 3. View the Bots Dashboard.

- a. In the left pane, click **Dashboards > Bots** to open the Bots Dashboard.



The Bots Dashboard presents the following:

- Bot Heartbeat

- Displays the failure scores for bots
 - You can use this graph to determine which bot is using the most resources.
 - MVP Bots
 - Displays bots based on their maximum processing time
 - Bot Status
 - Displays the status of active bots (i.e. Deployed, Completed, Failed)
 - Top Failure Reasons
 - Displays groupings and counts of errors encountered by bots
- ___ 4. View other dashboards.
- ___ a. In the left pane, click the other dashboards to view what information is presented for each.

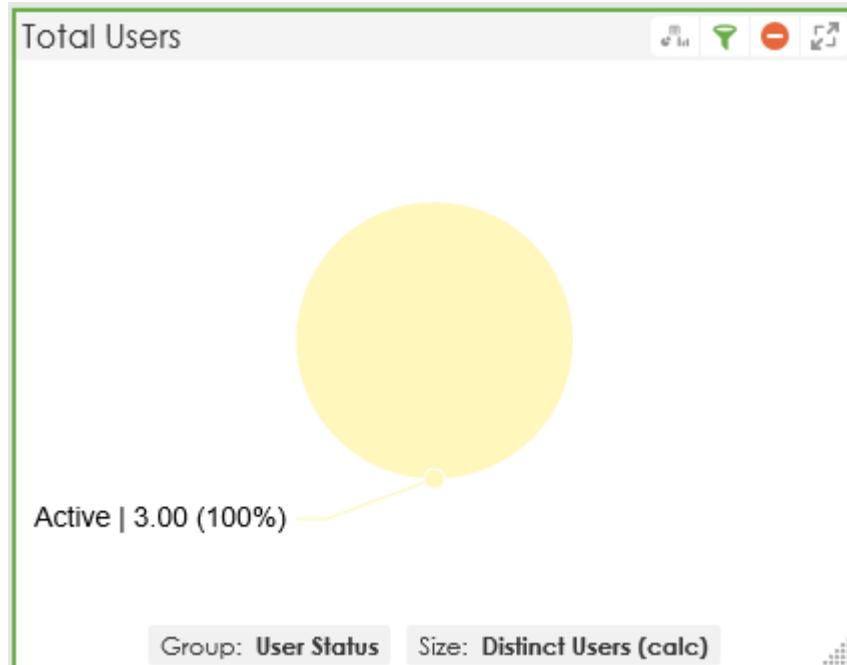
You will not see active data since you have not run any bots yet. You may see data as a result of the course build in the Audit trail dashboard. As a review, the dashboards include:

- Home Dashboard
 - Provides a high-level view including bot run status, total users, and more
- Bots Dashboard
 - Provides a high-level view of overall bot status and heartbeat
- Devices Dashboard
 - Provides a high-level view of bot runner machines and their status
- Audit Dashboard
 - Provides a snapshot of audit information along with a visual representation
- Workload Dashboard
 - Provides workload status of device pools, queues and work items
- Insights Dashboard
 - Currently not supported in the IBM product

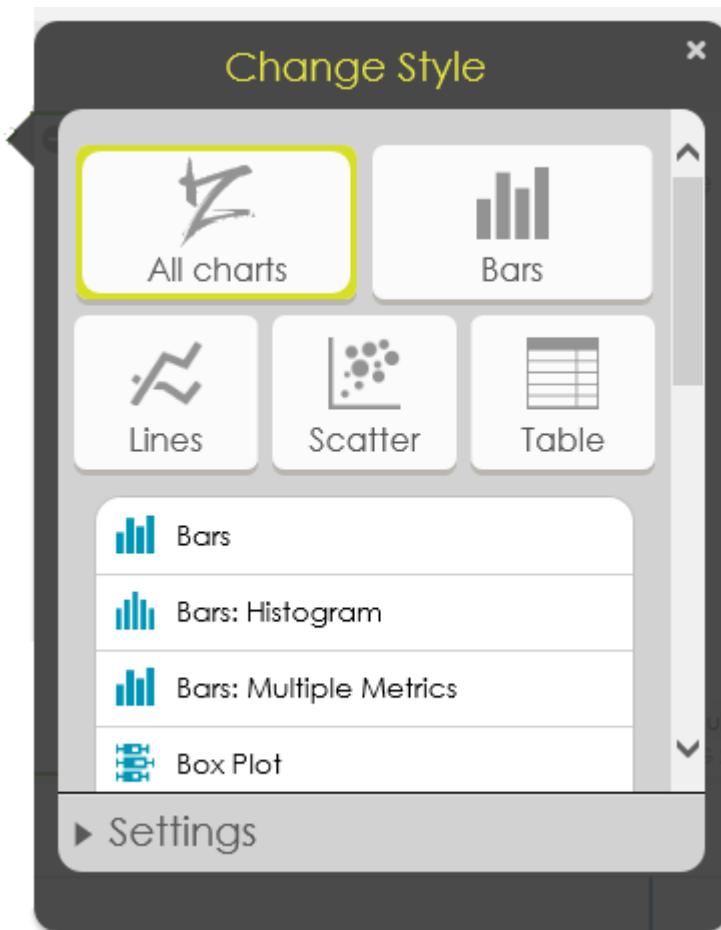
- ___ 5. Configure dashboards.

As you view the widgets, you can configure their display by changing the chart style or configuring the filter they use. To do this, highlight the widget (green box surrounds widget) and select the appropriate function in the top right corner.

- ___ a. In the left pane, click **Dashboards > Home** to open the Home Dashboard. The Total Users graph is highlighted.
- ___ b. Mouse over the **Total Users** graph to view the configuration options in the top right corner.



- ___ c. When you select the **Chart Style icon** (first icon), you will be presented with the pop up below to configure the chart style.





Information

The dashboard widgets are highly configurable, and you can view them full screen or simply enlarge them by dragging on the right bottom corner. Although these settings are the most common, other widgets may have other configurable settings such as timeline.

- ___ d. In the **Change Style** dialog box, click **Bars**. The graph is converted from a pie chart to a bar chart.
- ___ e. Select the **Pie** style to reset the graph back to the original format.



Troubleshooting

If the dashboards are not showing up in the browser, stopping and starting the Automation Anywhere Bot Insight Service may fix the issue.

The screenshot shows the Windows Services (Local) window. On the left, there is a list for the 'Automation Anywhere Bot Insight Service' with options to Stop, Pause, and Restart the service. A description below states: 'Routes service requests within the Bot Insight application.' To the right is a table listing various services:

Name	Description
Automation Anywhere Bot Insight EDC	Used to extract data from
Automation Anywhere Bot Insight PostgreSQL	The PostgreSQL database
Automation Anywhere Bot Insight Query Engine	Query Engine for Bot Ins
Automation Anywhere Bot Insight Scheduler	Periodically updates Aut
Automation Anywhere Bot Insight Service	Routes service requests v
Automation Anywhere Bot Insight Service Discover...	Provides the ability for B
Automation Anywhere Bot Insight Visualization	Renders service requests
Automation Anywhere Control Room Caching	Control Room Ignite Ser
Automation Anywhere Control Room Messaging	Allows Control Room se

Section 2: Exploring the Administrative Features

In this section you sign in to the Control Room as an administrator and explore some of its administrative features including credentials, the audit log, and server settings.

There are server configurations and settings that have been pre-built to support this course. In this section, you view the required credentials used in exercises later in the course.

- ___ 1. If not logged in to the Control Room, log in now as **rpaAdmin**.
- ___ 2. View Credentials.
 - ___ a. Go to **Bots > Credentials**

Credentials

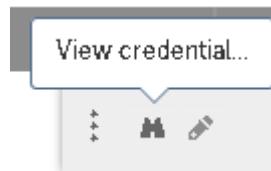
[Create credential...](#)[MY CREDENTIALS](#)[MY LOCKERS](#)[CREDENTIAL REQUESTS](#)Search name 

Credentials {2 of 2}



<input type="checkbox"/>	TYPE	NAME ↑	LOCKER NAME	MY A
<input type="checkbox"/>	Standard	ACCESS_CONNECT_STRING	IBM Business Automation Workflow Systems	
<input type="checkbox"/>	Standard	DB_CONNECT_STRING	IBM Business Automation Workflow Systems	

- ___ b. Mouse over the ellipsis for the **ACCESS_CONNECT_STRING** credential.
- ___ c. Click the **View credential** icon.



- ___ d. View details regarding the credential. The Username and Password attributes will be used in a future exercise.

ACCESS_CONNECT_STRING

[Edit](#)
[Back](#)

CREDENTIAL DETAILS

Description	Locker	My access	Credential owner
Connection credentials for Access database	IBM Business Autom...	Credential nonowner	devuser1

ATTRIBUTE NAME	DESCRIPTION	TYPE	VALUE
Username	User name for Access database	Standard	accountManager
Password	Password for Access database	Standard	accountManager

GENERAL DETAILS

Last modified 15:53:39 EDT 2018-08-15	Modified by devuser1
---	-------------------------

Object type Credential	Credential type Standard
---------------------------	-----------------------------

- ___ e. Click **Back**.
- ___ f. Mouse over the ellipsis for the **DB_CONNECT_STRING** credential.
- ___ g. Click the **View credential** icon.
- ___ h. View details regarding the credential. Notice this credential is defined differently using the IBM OLE DB driver.

DB_CONNECT_STRING

[Edit](#)
[Back](#)

CREDENTIAL DETAILS

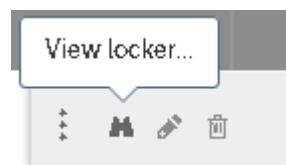
Description	Locker	My access	Credential owner
Connection credentials for DB2	IBM Business Autom...	Credential nonowner	devuser1

ATTRIBUTE NAME	DESCRIPTION	TYPE	VALUE
DB_CONNECT_STRING	DB2 connection string	Standard	Provider=IBMOLEDB.DB2CO

GENERAL DETAILS

Last modified 15:52:24 EDT 2018-08-15	Modified by devuser1
Object type Credential	Credential type Standard

- ___ i. Click **Back**.
- ___ 3. View Locker details.
 - ___ a. Click the **My Lockers** tab.
 - ___ b. Mouse over the ellipsis for the **IBM Business Automation Workflow Systems** locker and click **View locker**.



- ___ c. View details regarding the locker. Notice the two credentials are held in this locker.

 IBM Business Automation Workflow Systems [Back](#)

[!\[\]\(2fff3c9e4afb192f04e878e77fb552ed_img.jpg\) Create credential...](#) [!\[\]\(424424d14aba49c86d1c60a64247128f_img.jpg\) Edit locker...](#) [!\[\]\(c6faaa65325805483013e4324c1f0b1d_img.jpg\) Delete locker...](#)

LOCKER DETAILS

Description	Locker to hold credentials for training.
My Consumer permissions	My Additional permissions
Not a consumer	Locker owner

CREDENTIALS

OWNERS

MANAGERS

PARTICIPANTS

CONSUMERS

Locker credentials

Search name 

Credentials (2)

NAME ↑	LOCKER NAME
ACCESS_CONNECT_S...	IBM Business Automation Workf
DB_CONNECT_STRING	IBM Business Automation Workf

GENERAL DETAILS

Last modified 15:17:11 EDT 2018-08-15	Modified by devuser1	Object type Locker
---	-------------------------	-----------------------

4. View Audit Log.

- a. Click **Audit Log** in the navigation bar to the left. Recall that all Control Room activities are logged. For example: user log in, client login (user with development license), installing bots, exporting bots, running bots on bot runners, and settings changes.

__ b.

Audit log

Item name	▼	Search item name		
Actions (612 of 612)				
STATUS	TIME ↓	ACTION TYPE	ITEM NAME	
<input type="checkbox"/>	 Successful	11:28:29 EST 2018-11-29	User log in	N/A
<input type="checkbox"/>	 Successful	18:03:52 EST 2018-11-27	Connect Credential Vault	Express

- __ c. View details of an entry by performing a mouse over the ellipsis and clicking **Audit details** (binoculars icon).
- __ 5. View Server settings.
- __ a. Go to **Administration > Settings**.

Settings

General	▼
Bots	▼
Client application	▼
Credentials	▼
Email	▼

- __ b. Click the downward double arrow for the **General** section. The **General** section is expanded. Here you can view settings such as the Control Room access URL. By clicking the **Control Room Database and Software** tab, you can view settings for website security and the Control Room database configuration.

Administration > Settings

General

Configuration settings

Many of these settings were initially configured during installation.

GENERAL SETTINGS **CONTROL ROOM DATABASE & SOFTWARE**

General

Installation type
Custom

Control Room access URL
<http://vclassbase:81>

Modified by System	Last modified 11:43:43 EDT 2018-08-15
-----------------------	---

- ___ c. Expand other sections by clicking the downward double arrow for the other sections. Note you can have multiple sections open at once. If you expand the Email section, you will see the email server configuration allowing Automation Anywhere to use the hMail server as its email server.

Email

Notifications

Send email notifications

From this email address
rpaAdmin@sib.com

Email server host smtp.sib.com	Email server port 25
-----------------------------------	-------------------------

My server does not use a secure connection (SSL/TLS)

My server does not require authentication

Send an email when

No events chosen

Modified by rpaadmin	Last modified 12:25:06 EDT 2018-08-15
-------------------------	---

Section 3: Create a development user

In this section, you learn how to create an IBM Robotic Process Automation with Automation Anywhere user.

Scenario: As an IBM Robotic Process Automation with Automation Anywhere administrator, you must create a development user account for a member of the IT team.

This process consists of entering user information, assigning a user role, and allocating a development license.

Several user roles are predefined, such as the **Admin** user role that the `rpaAdmin` user belongs to, but the **Bot Creator** security role was created for this course.

- 1. In the Control Room, go to **Administration > Users**.

	USER TYPE	USERNAME	FIRST NAME	LAST NAME
<input type="checkbox"/>	Bot runner	botuser1	Bot	User1
<input type="checkbox"/>	Bot creator	devuser1	Dev	User1

- 2. Click **Create User...**
- 3. Define a user who is called `myDevUser`.
 - a. Complete the General Details section with the following information:

User Name: myDevUser

First Name: myDev

Last Name: User

Email: myDevUser@sib.com

General details

Enable user

Username

myDevUser

0\/*'[]<>+=;,?* are not allowed

Description (optional)

Max characters = 255

First name (optional)

myDev

Max characters = 50

Last name (optional)

User

Max characters = 50

Email

myDevUser@sib.com

Confirm email

myDevUser@sib.com

- __ b. In the **Select roles** section, select **BotCreator**. A yellow right arrow appears to the right.

Select roles

Select one or more roles

Search name

Available roles (8 of 8)

	NAME ↑
<input type="checkbox"/>	AAE_Basic
<input type="checkbox"/>	AAE_IQ Bot Services
<input type="checkbox"/>	AAE_IQ Bot Validator
<input type="checkbox"/>	AAE_Locker Admin
<input type="checkbox"/>	AAE_Pool Admin
<input type="checkbox"/>	AAE_Queue Admin
<input checked="" type="checkbox"/>	BotCreator

- ___ c. Click yellow right arrow to select the role. The role appears on the right.

Selected (1)

<input type="checkbox"/>	NAME ↑
<input type="checkbox"/>	BotCreator



Information

If it is required, you can assign multiple roles to a user.

- ___ d. In the Allocate license section, select **Bot creator**. You can leave the checkmark for enabling auto login.

Allocate a device license to this user?

Device licenses are only applicable if the user does not have the "Admin" or the "BotFarm admin" role.

A device, or Client UI, cannot connect to the Control Room until the user that logs into it has a device license. If you change from a Bot runner license to a Bot creator license, any schedules associated with this username will be deleted.

<input type="radio"/> Bot runner	Requires a Development license, which enables the user to create AND run Task Bots.
<input checked="" type="radio"/> Bot creator	<input checked="" type="checkbox"/> Enable auto login This allows the Client UI to remember the password and automatically log into the Control Room.
<input type="radio"/> None	

- ___ e. Scroll to the top and click **Create user**.

Create user

[Cancel](#)

[Create user](#)

All users have access to the Control Room. To give the user further permissions, allocate the appropriate license below (users with the Admin role cannot have a license).

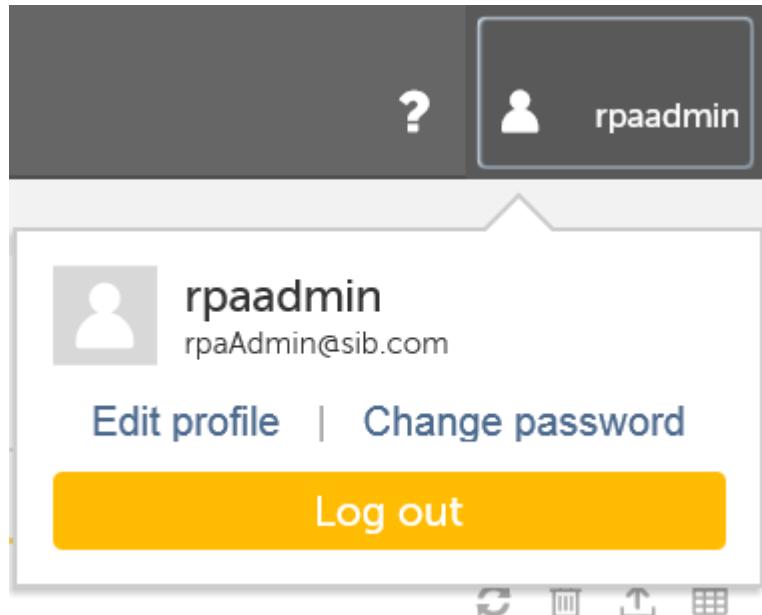
If SMTP is enabled, once you create this user, an email will be sent to them inviting them to log in.

- ___ f. A pop up message appears stating "**myDevUser** successfully created." **myDevUser** is now in the user list.

When a user is created in the Control Room, and the Control Room is configured to send email, the Control Room automatically sends a notification to the email address that was entered.

The user must then open the email to confirm the account and create a password. You complete these steps in the next section.

- ___ 4. Log out of the Control Room as the administrator.
- ___ a. On the top right, click **rpaAdmin**.



- ___ b. Click **Logout**.
- ___ c. Close the Control Room browser window.

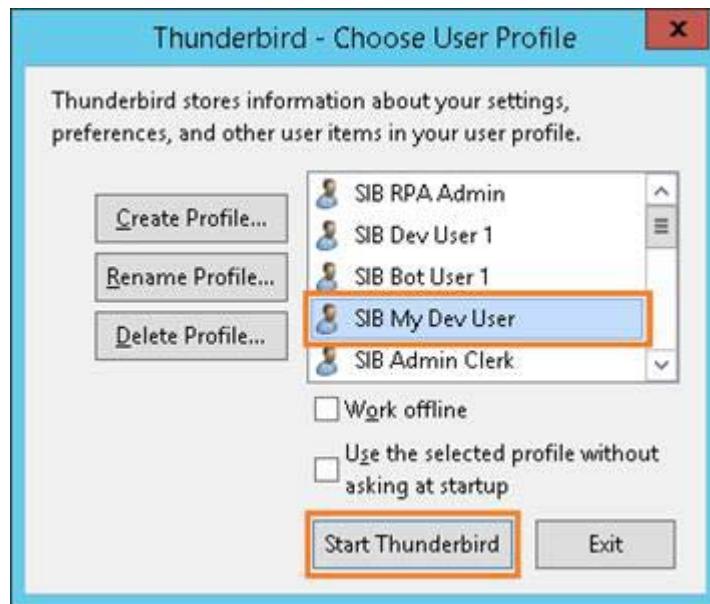
Section 4: Confirming the development user account

In this section, you take on the role of the development user `myDevUser` to confirm the development user account and create a password. You then start the Enterprise Client to confirm that your user account works.

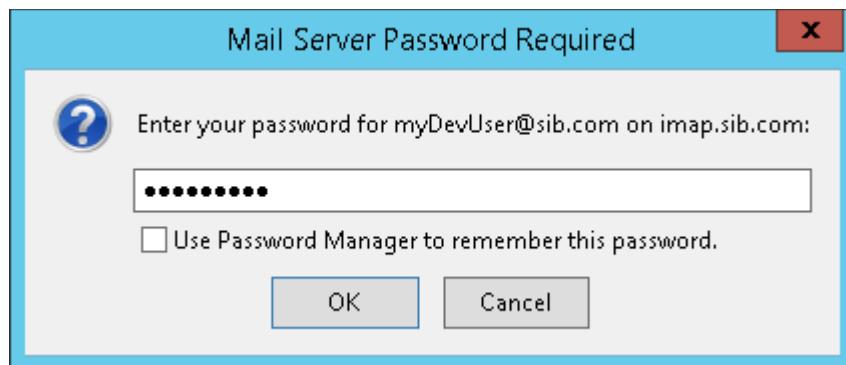
- ___ 1. Open Mozilla Thunderbird and log in as `myDevUser`.
- ___ a. On the desktop, double-click the **Mozilla Thunderbird** shortcut.



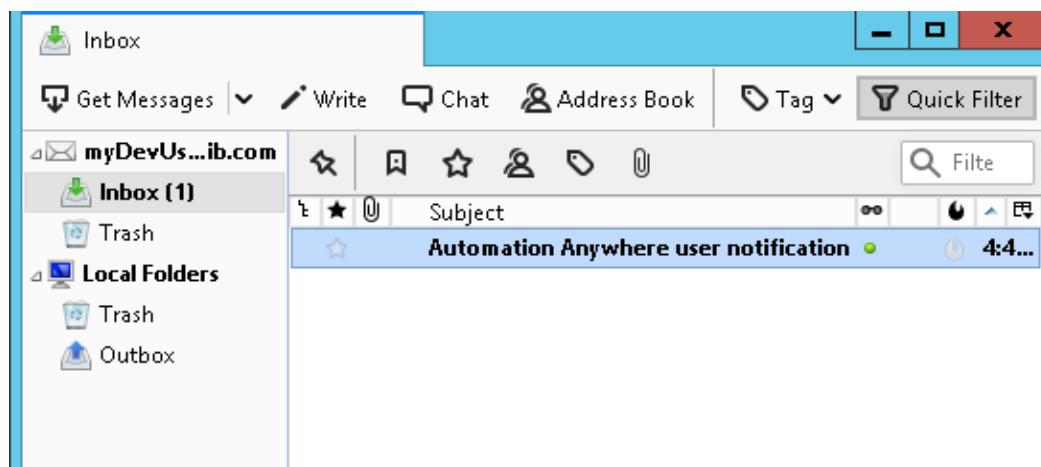
- ___ b. In the Thunderbird - Choose User Profile window, select **SIB My Dev User**, and click **Start Thunderbird**.



- ___ c. In the Mail Server Password Required window, enter `myDevUser` and click **OK**.



- ___ 2. Open the `myDevUser` mail inbox and confirm the Automation Anywhere account.
- ___ a. In Thunderbird, in the left pane, click **myDevUser@sib.com > Inbox**.
- ___ b. In the Inbox pane, click the **Automation Anywhere credentials notification** email to open it.





Information

Because of the way the IBM Business Automation Workflow Systems Locker is configured, the account will be given “Consumer” permissions to the credentials. You may see an email in Thunderbird notifying the user of this.

- ___ c. In the email pane, review the message.

In addition to the link that you use to confirm your account and set your password, the email message lists the web address for the Control Room.

- ___ d. In the email message, click **go to Automation Anywhere’s Control Room now** to confirm the account. The Control Room starts and opens.

From rpaAdmin@sib.com☆ More ▾

Subject **Automation Anywhere user notification** 4:44 PM

To Me ☆

Hello mydevuser;

Your user account has been activated. To get started, [go to Automation Anywhere's Control Room now.](#)

For future reference, the URL for the Control Room is: <http://vclassbase:81#login>

Thanks and go be great,


Automation Anywhere team
www.AutomationAnywhere.com

- ___ e. Enter `myDevUser` for the **Password** and **Confirm password** fields.

Welcome! First things first...

[Save and log in](#)

PASSWORD

Username: mydevuser

Password: *****

SECURITY QUESTIONS

Create a password

Please create a password and note down your username.

Password
***** ∅

8-15 characters; a-z, A-Z, 0-9, @, ., _, !, #, \$, %, &, and allowed.

Confirm password
***** ∅

[Next >](#)

- ___ f. Click **Next**. The Create security questions window appears.
- ___ g. For the purposes of this class, you can enter the following:
 - Question 1: 111 Answer 1: 111
 - Question 2: 222 Answer 2: 222
 - Question 3: 333 Answer 3: 333

Welcome! First things first...

[Save and log in](#)

PASSWORD

Username: mydevuser

Password: *****

SECURITY QUESTIONS

- Question 1
 - 111
 - Answer: *****
- Question 2
 - 222
 - Answer: *****
- Question 3
 - 333
 - Answer: *****

Create security questions

Provide three security questions and the answers of your choice to be asked in case you lose your password. Questions and answers are not case sensitive.

Question 1
 Min characters = 3

Answer
*** ∅

Min characters = 3

[< Back](#)

- ___ h. Click **Save and log in**. The Control Room appears logged in as myDevUser. The Home dashboard is displayed.
 - ___ 3. Log out of the Control Room as the administrator.
 - ___ a. On the top right, click **myDevUser**.
 - ___ b. Click **Logout**.
 - ___ c. Close the Control Room browser window.
 - ___ 4. Confirm that the `myDevUser` account can access the Enterprise Client.
-



Note

Though you confirm that the development user can open and log in to the Enterprise Client, you do not work with the Enterprise Client as `myDevUser` in the bot development exercises.

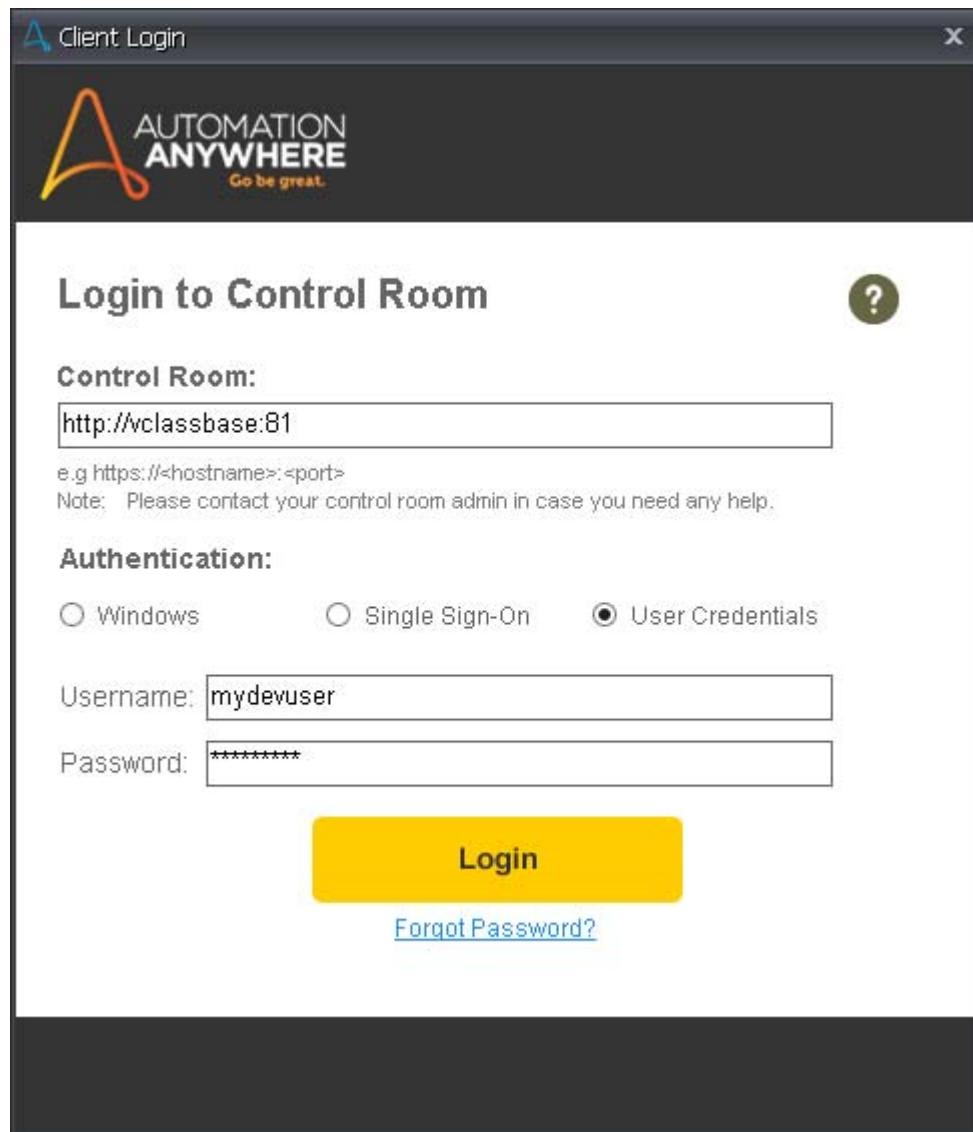
- ___ a. On the Desktop, double-click the **AA Enterprise Client** shortcut.



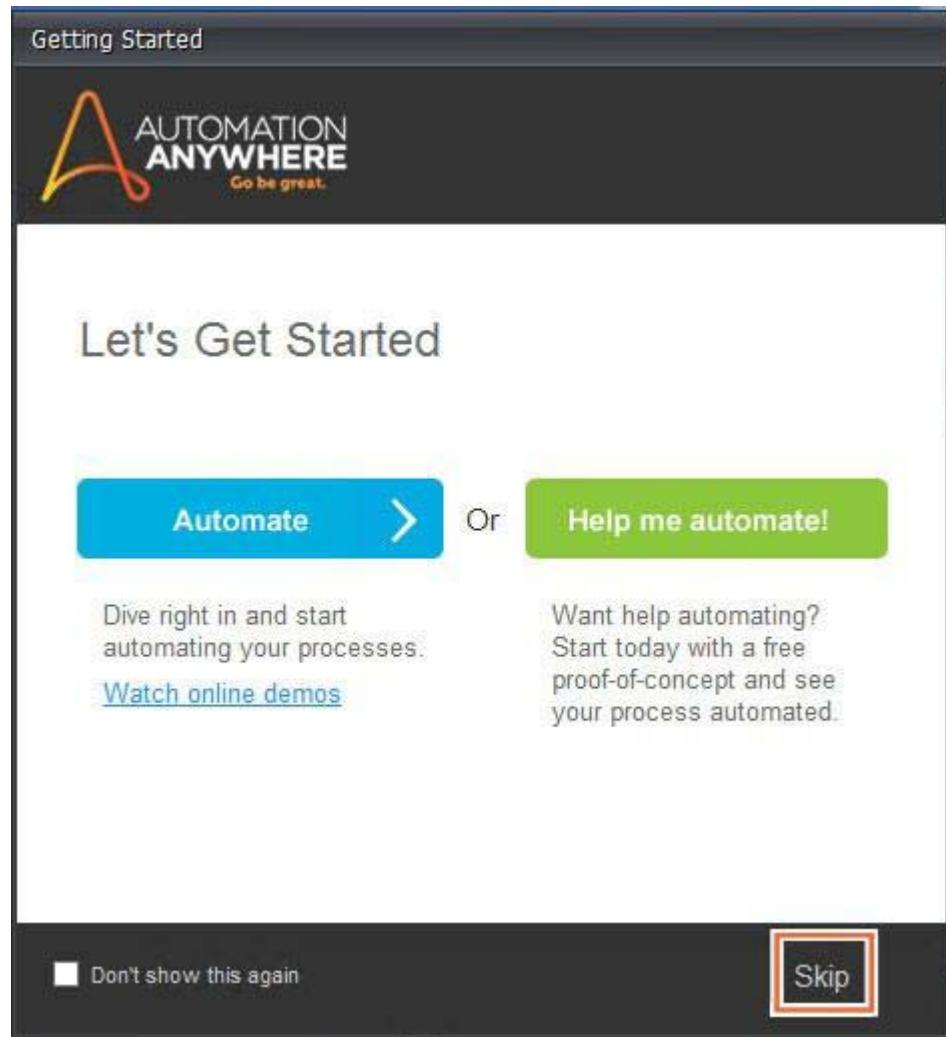
- ___ b. In the Client Login window, make sure that the **Control Room** has the following URL:
`http://vclassbase:81`

If the Control Room URL is not already in the field, you can copy (Ctrl+C) and paste (Ctrl+V) the Control Room web address from the Control Room browser window to this field.

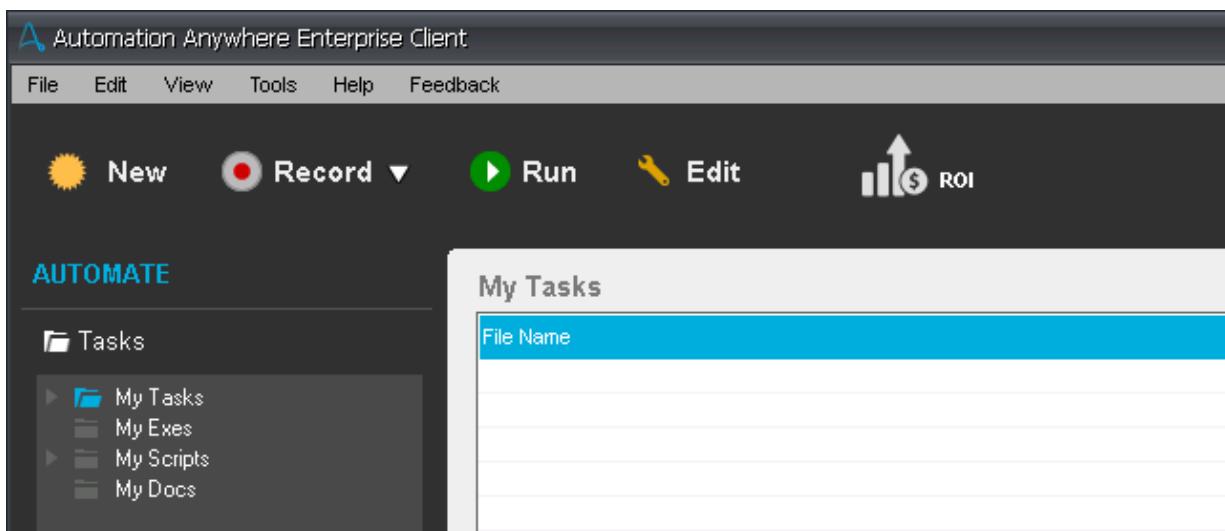
- ___ c. In the **Username** field, enter: `myDevUser`
- ___ d. In the **Password** field, enter: `myDevUser`



- ___ e. Click **Login**.
- ___ f. In the Getting Started window, click **Skip**.



Wait for the Enterprise Client to load. It might take a couple of minutes. You are now logged in to the Enterprise Client.



You are introduced to the Enterprise Client interface in the next exercise.

- g. Close the Enterprise Client.
- h. Close Mozilla Thunderbird.
- i. Log out of the Control Room as `myDevUser` and close the Control Room window.

End of exercise

Exercise review and wrap-up

In the first part of the exercise, you logged in to the Control Room as the main administrative user and explored some of the features of the Control Room. Next, you continued to work as the administrator and created a development user. You then assigned this user the `BotCreator` role and a development license. In the last part of the exercise, you worked as the development user to confirm the Automation Anywhere account. As the development user, you logged in to the Control Room and opened the Enterprise Client.

Exercise 2. Creating a basic bot

Estimated time

00:45

Overview

This exercise introduces you to the IBM Robotic Process Automation with Automation Anywhere Enterprise Client interface. You also learn how to create a basic bot.

Objectives

After completing this exercise, you should be able to:

- Explore the Enterprise Client interface
- Work with the Workbench to add and edit bot commands
- Create a bot that opens a text editor, creates a text file, and saves the file

Introduction

In this exercise, you sign in to the Enterprise Client, which requires the Control Room URL, and become familiar with the bot commands. Next, you create a bot that opens Notepad, writes text, and saves the text file to your desktop.

The exercise includes these sections:

- [Section 1, "Exploring the Enterprise Client"](#)
- [Section 2, "Automating the creation and saving of a text file"](#)

Requirements

This exercise requires the computer lab environment that was developed for this course.

Section 1. Exploring the Enterprise Client

In this section, you log in to the Enterprise Client as the development user `devUser1`. You also explore some of the Enterprise Client features.

1.1. Signing in to the Enterprise Client

- ___ 1. Log in to the Web Control Room as the administrator.
 - ___ a. Double-click the **Automation Anywhere Enterprise Control Room** desktop shortcut.



- ___ b. In the Control Room Login page, in the **Username** and **Password** fields, enter: `rpaAdmin`
- ___ c. Click **Login**.

- ___ 2. Minimize the Control Room window, but keep it open.
-



Troubleshooting

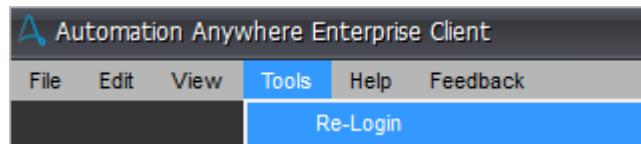
If you have login time out problems with the Control Room, use Chrome as the web browser:

- Open a Chrome web browser window.
 - Copy the URL from the Control Room window, and paste it into Chrome.
 - The Control Room URL is: `http://vclassbase:81`
 - Log in by entering `rpaAdmin` in the **Username** and **Password** fields.
-

- ___ 3. If the IBM Robotic Process Automation with Automation Anywhere Enterprise Client is not running, start the Enterprise Client and log in as the bot developer user `devUser1`.
 - ___ a. Double-click the **AA Enterprise Client** desktop shortcut.



- ___ b. If Enterprise Client opens directly, click **Tools > Re-Login**.





Information

The user logged in to the Enterprise client is displayed at the bottom left.

- ___ c. In the login window, make sure that the Control Room URL is correct.

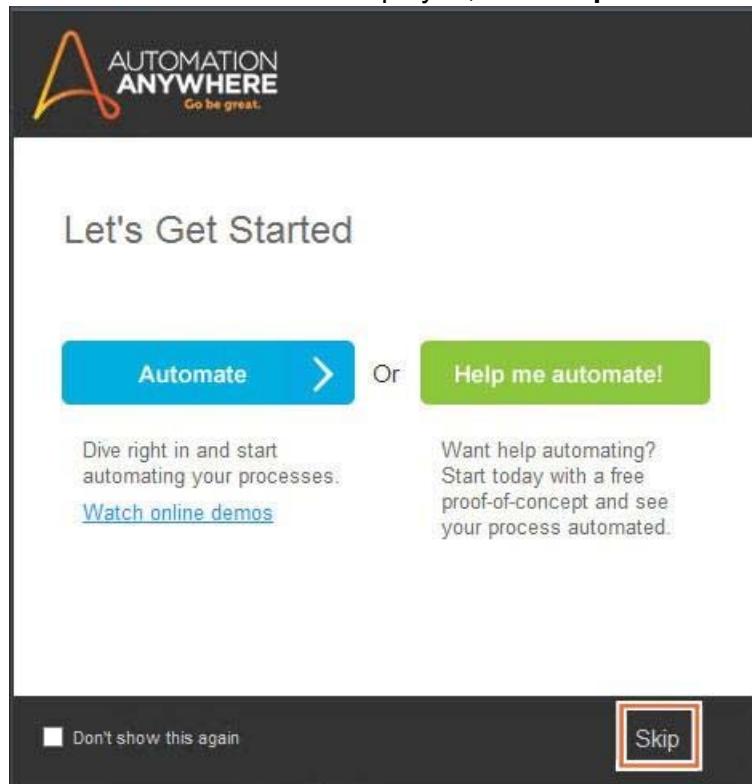
If needed, you can copy the URL from the Control Room window and paste it into this field.

- ___ d. In the **Username** and **Password** fields, enter: devUser1

- ___ e. Click **Login** and in the login successful message window, click **OK**.

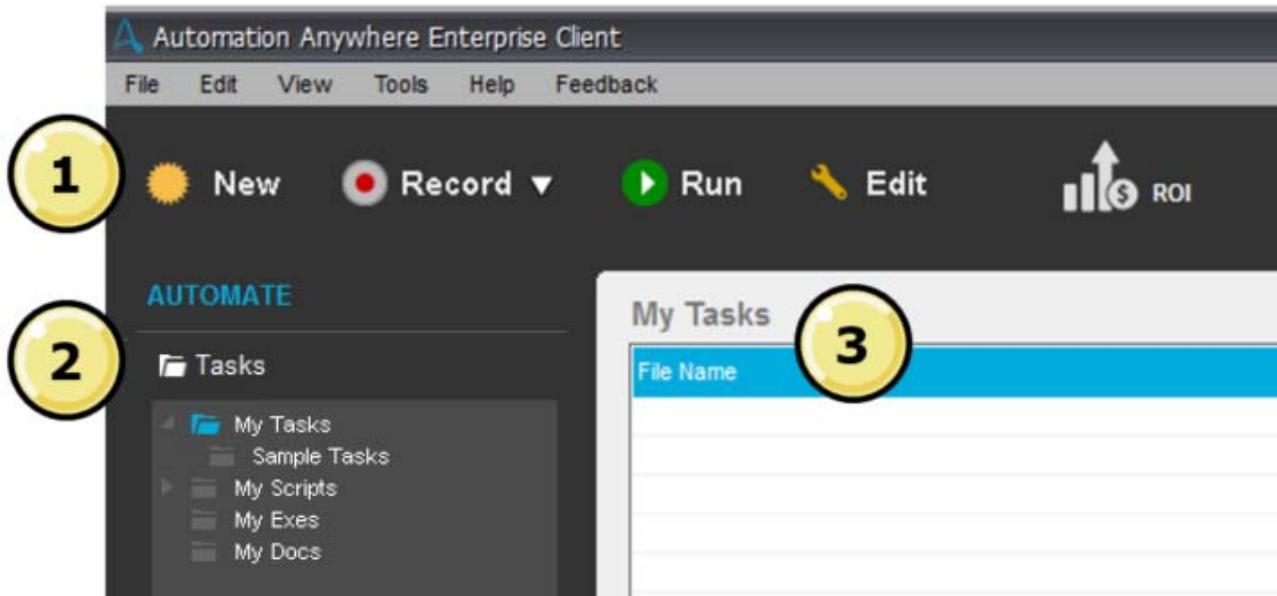
The screenshot shows the 'Client Login' window for Automation Anywhere. At the top, there's a logo for 'AUTOMATION ANYWHERE Go be great.' Below it, the title 'Login to Control Room' is centered. To the right of the title is a question mark icon. Underneath the title, the text 'Control Room:' is followed by a text input field containing 'http://vclassbase:81'. Below this field, there's a note: 'e.g https://<hostname>:<port>' and 'Note: Please contact your control room admin in case you need any help.' Under 'Authentication:', there are three radio buttons: 'Windows' (unchecked), 'Single Sign-On' (unchecked), and 'User Credentials' (checked). Below these buttons are two text input fields: 'Username' with 'devUser1' and 'Password' with '*****'. A large yellow button labeled 'Login' is positioned below the password field. At the bottom of the window, there's a link 'Forgot Password?'

- ___ f. If the “Let’s Get Started” window is displayed, click **Skip**.



1.2. Exploring the Enterprise Client

The Enterprise Client window has the following main features: (1) the toolbar, (2) the Automate pane, and (3) the Task pane.



- The three main parts of the Enterprise Client are: (1) the toolbar, (2) the Automate pane, and (3) the Bot list. As you access folders on the left, the bots in the folder are displayed on the right.
- **1. Toolbar:** Access Enterprise Client tools:

- **New:** Create a task through one of the recorders (Smart Recorder, Screen Recorder, or Web Recorder), or open the Workbench window
 - **Record:** Open the Smart Recorder to start recording task actions
 - **Run:** Run the selected task (bot)
 - **Edit:** Open the selected task in the Workbench
 - **2. Automate** pane: Access tasks (bots) and other RPA artifacts
 - **3. Tasks** pane: Lists the RPA artifacts that are in the various **Automate** pane folders
- 1. In the **Automate** pane, click **Sample Tasks**.

The **Sample Tasks** folder contains some simple example tasks (bots). When a folder in the Automate pane is active and contains RPA artifacts, the list of artifacts loads in the Task pane.

The screenshot shows the IBM Worklight Studio interface. On the left, the 'AUTOMATE' pane is open, displaying a tree view of 'Tasks'. Under 'Tasks', there are several categories: 'My Tasks', 'IBM', 'TaskBot Import (Tech Preview)', 'Sample Tasks' (which is currently selected and highlighted in blue), 'My Scripts', 'My Exes', and 'My Docs'. To the right of the Automate pane is the 'Sample Tasks' list. The list has a header row with columns for 'File Name' and 'Examples Online'. Below this, there is a list of RPA artifacts, with 'Files-Folders.atmx' being the selected item (highlighted in blue). Other items in the list include 'Analytics_ATM Reconciliation.atmx', 'Analytics_MortgageProcessing.atmx', 'Analytics_TelecomOrderEntry.atmx', 'List-Variable.atmx', 'Loops.atmx', 'Prompt.atmx', and 'Variables.atmx'.

The bots under the IBM folder are used to integrate with IBM Business Automation Workflow. RPA - BAW integration is not covered in this course.

- 2. When a bot is highlighted on the top, the bottom pane displays Properties, Schedule, and Triggers for that bot.

The screenshot shows the 'PROPERTIES' tab of the properties pane for the 'Files-Folders.atmx' bot. The pane is divided into sections: 'General', 'Schedule', and 'Trigger'. The 'General' section contains the following details:

	Value		Value
Repeat	File Name :	Files-Folders.atmx	Task Report
Speed	Created at :	09-25-2014 21:25:29	Mouse Clicks 0
Notification	Status :	N/A	Keystrokes 1339
Hotkey	Last Run Time :	N/A	Total Clicks 1339
Security	Priority (for queuing) :	Low	
	Timeout (in minutes) :	0	
	<input type="checkbox"/> Enable this task to run with other similar files or window titles ?		



Information

The RPA artifacts that you see listed in the Automate pane are located in the following directory:

C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere

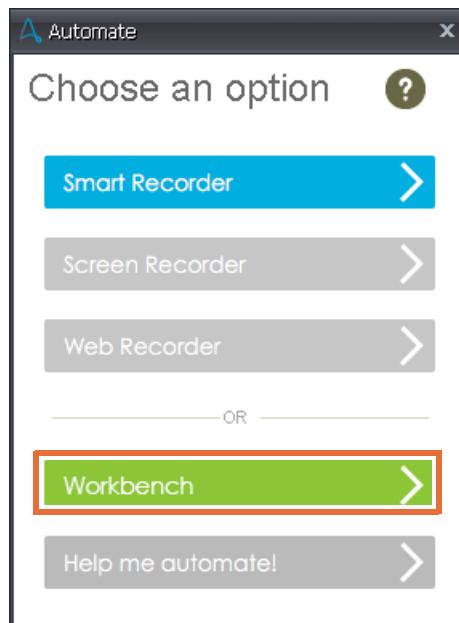
1.3. Exploring the Workbench

In this section, you create a task and explore the Workbench interface. You also learn how to add, edit, and delete commands from the Workbench.

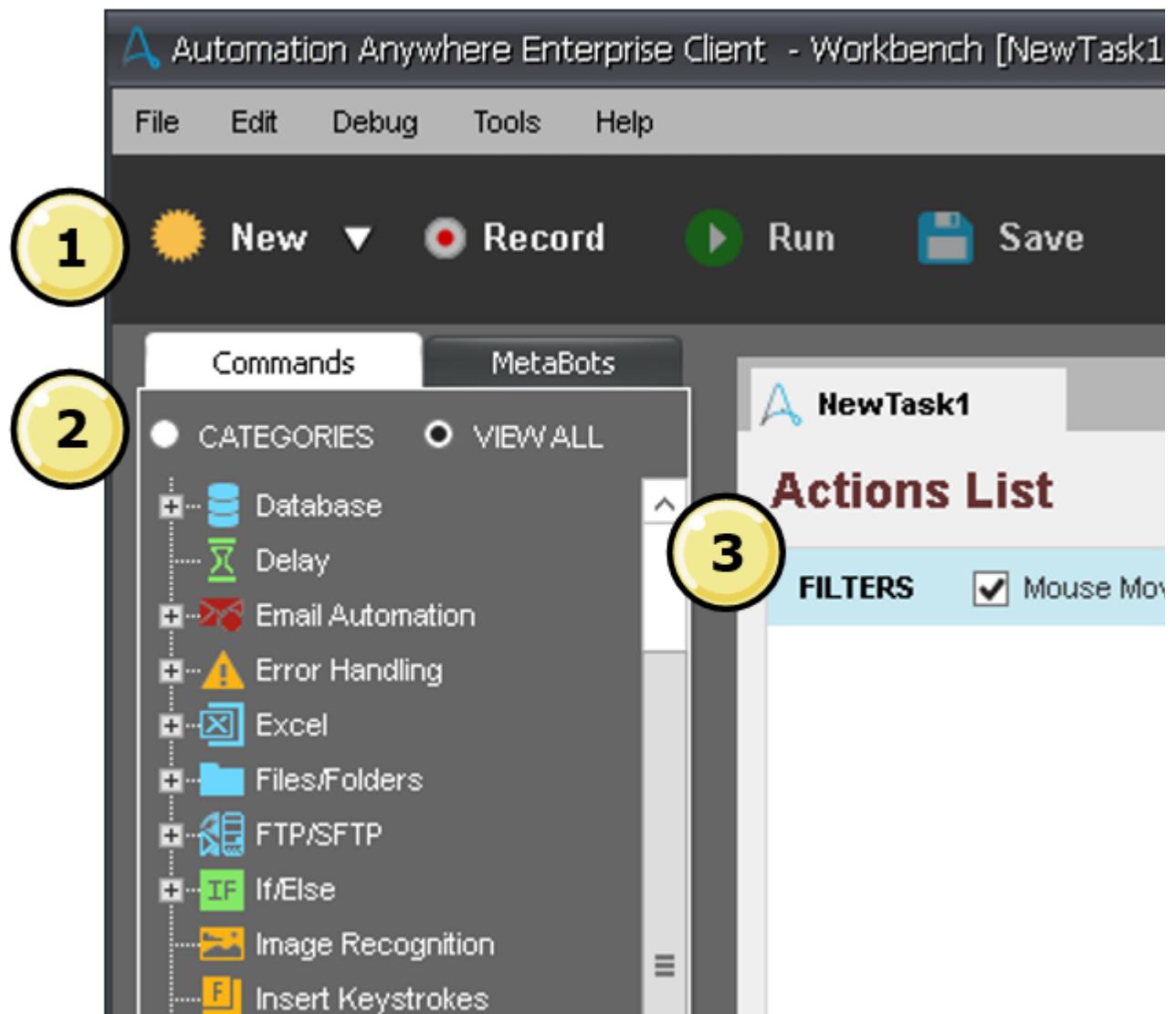
- 1. Create a bot by going to the Enterprise Client toolbar, click **New**.



- 2. In the Automate window, click **Workbench**.



The Workbench opens in a new window.



The Workbench is the main tool for creating and editing the list of commands (actions) that make up a bot.

The three main parts of the Workbench window are: (1) the toolbar, (2) the Commands list, and (3) the Actions List.

1. The toolbar provides quick access to the following features:
 - **New:** Create a task.
 - **Record:** Open the Smart Recorder for recording task actions.
 - **Run:** Run the open task.
 - **Save:** Save changes to the open task. You must save the task before you can run it. You cannot run a task with unsaved changes.
 - **Enable Debugging:** Opens Debugging mode, which you can use to troubleshoot bots and view variable values that are assigned during the bot run.

- **Set SnapPoint:** The SnapPoint feature is used to add screen captures to a task. Not all commands support the use of SnapPoints.

2. The **Commands** list has all the automation commands that you use to build bots.

- The list of available commands encompass various tasks, including:
 - o Application-specific tasks, such as working with Excel spreadsheets or SQL databases.
 - o System tasks, such as restarting the computer or working with the internet connection.
 - o Data handling tasks, such as working with local task variables and strings.
 - o Logic tasks, such as working with If/Else statements and error handling.
 - o Interface tasks, such as inserting keystrokes or mouse clicks.
 - o Interactive tasks, such as including message windows or prompt windows for user input.

3. The Actions List is the main workspace for building bots.

- It shows the commands that make up the bot.
- You can add and edit the commands in the list.
- Commands are run in sequential order.

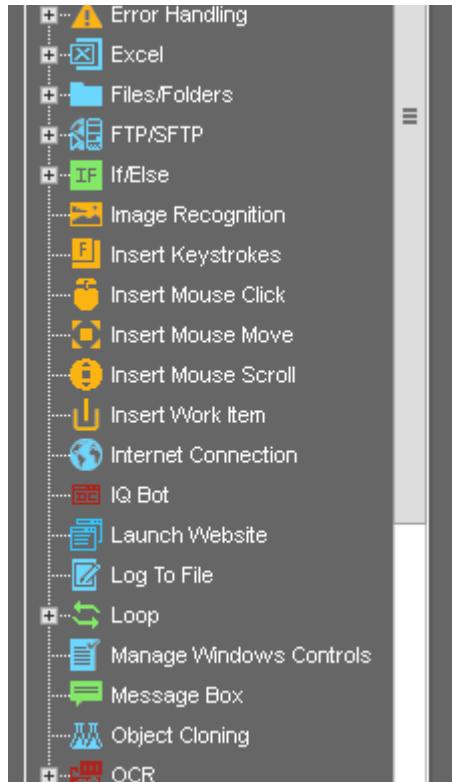
1.4. Working with the Workbench: Adding commands

You build bots by adding and editing commands and actions to the Actions List.

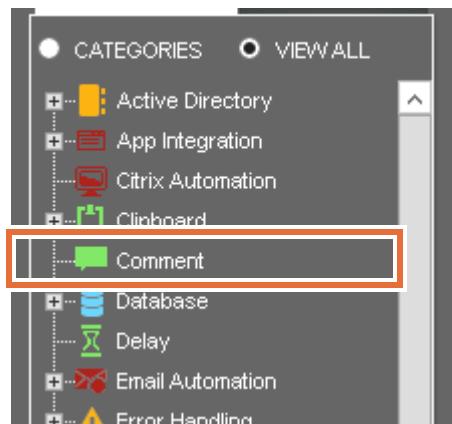
You can add tasks to the Tasks Action List by either double-clicking the command in the Commands list, or by dragging the command from the Commands list to the Actions List.

After you double-click or drag the command, you configure the command so that it performs the task that is required. Finally, you save the command to add it to the Actions List.

- 1. In the left side of the Workbench, scroll through the list of commands in the **Commands** list to familiarize yourself with this list.



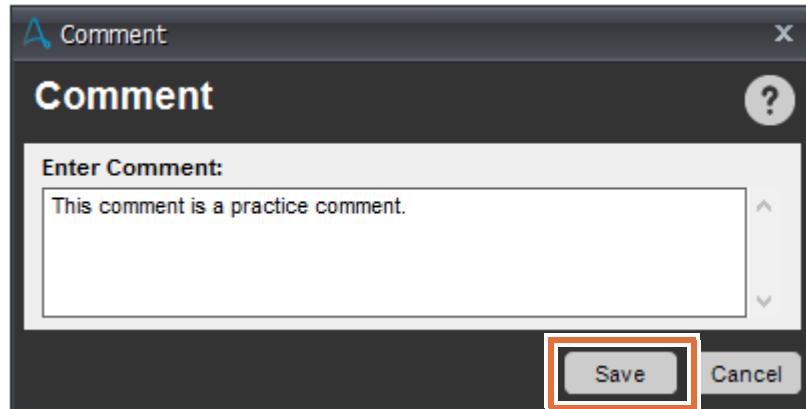
- 2. In the **Commands** list, double-click **Comment**.



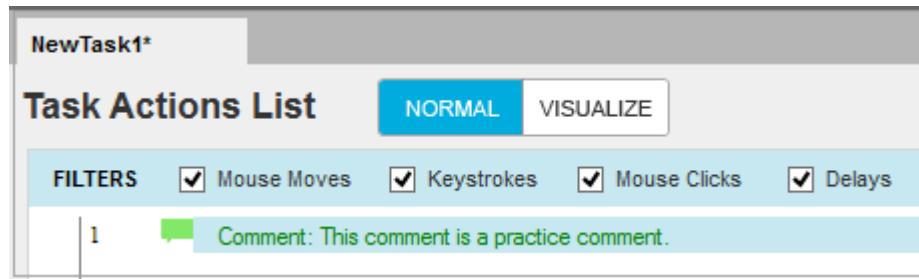
- 3. In the Comment window, enter some text in the **Enter Comment** field, such as:

This comment is a practice comment

- ___ 4. Click **Save**.

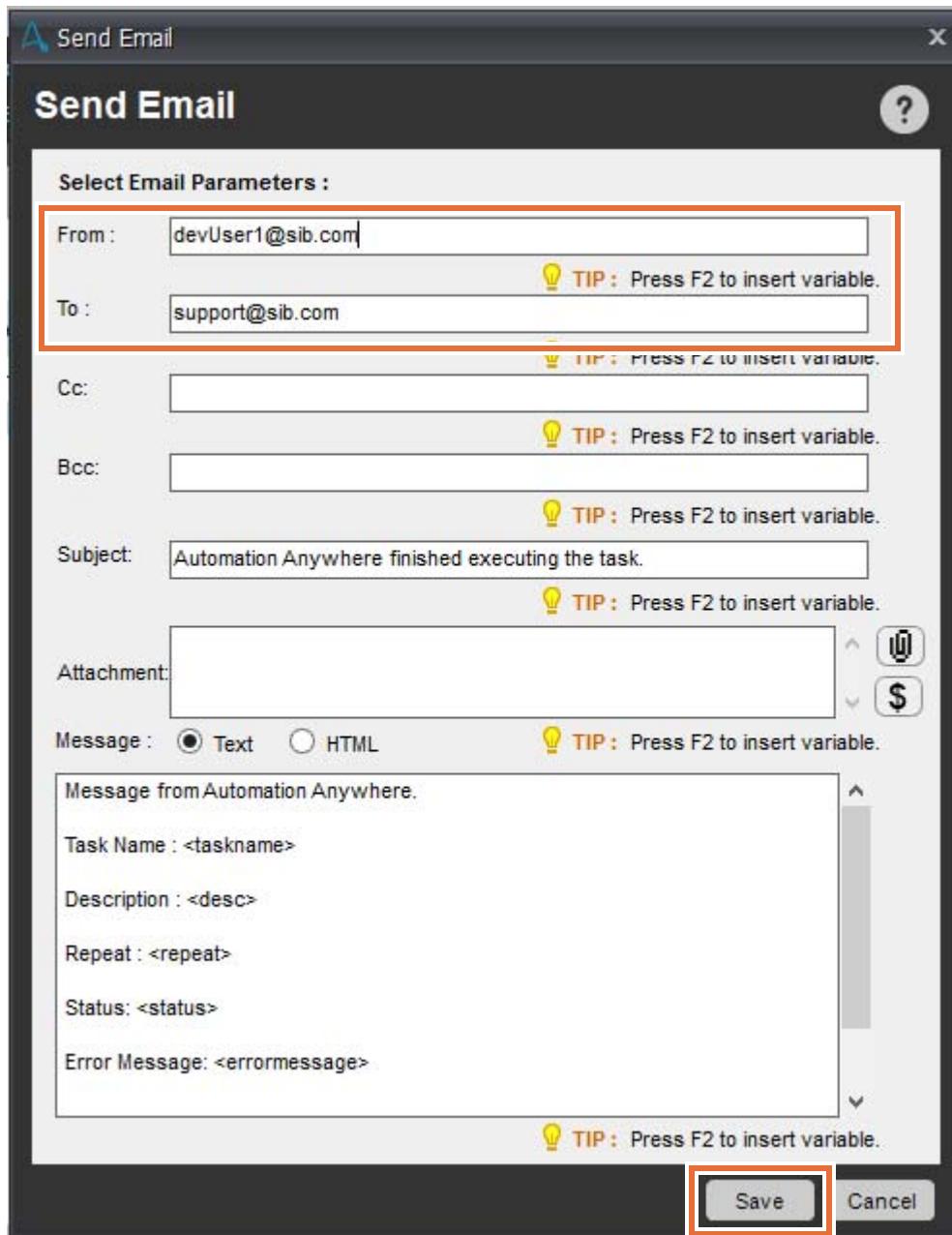


The comment is added to the Actions List. Notice that it is highlighted in blue. When you add a command to the Actions List, it is placed after whichever command is highlighted.

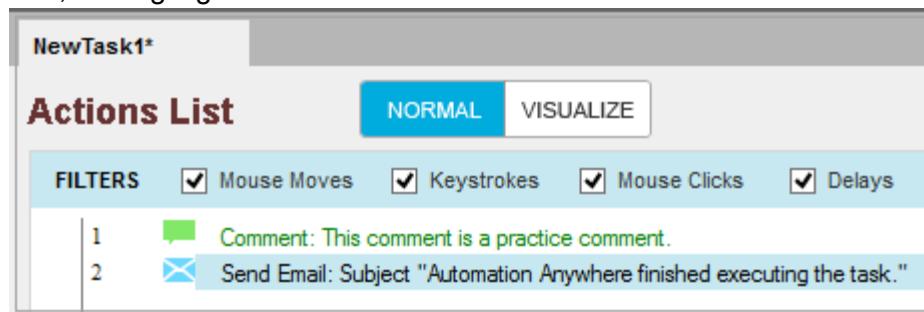


- ___ 5. In the **Commands** list, double-click the `Send Email` command.
 ___ 6. In the `Send Email` window, enter the following values:
- **From:** devUser1@sib.com
 - **To:** support@sib.com
 - Leave the rest of the options at their default settings.

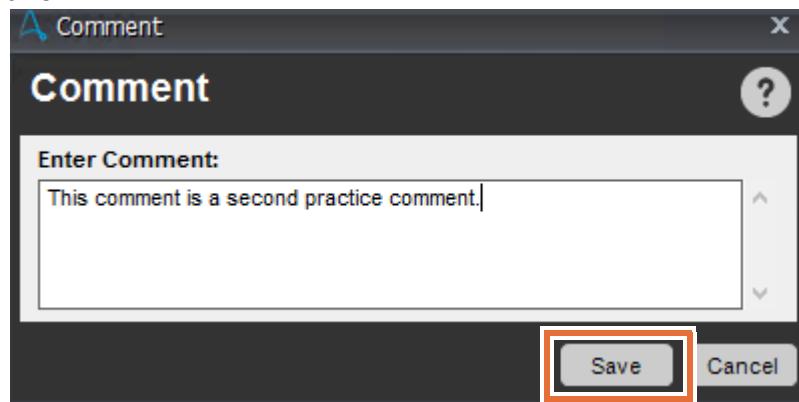
7. Click **Save**.



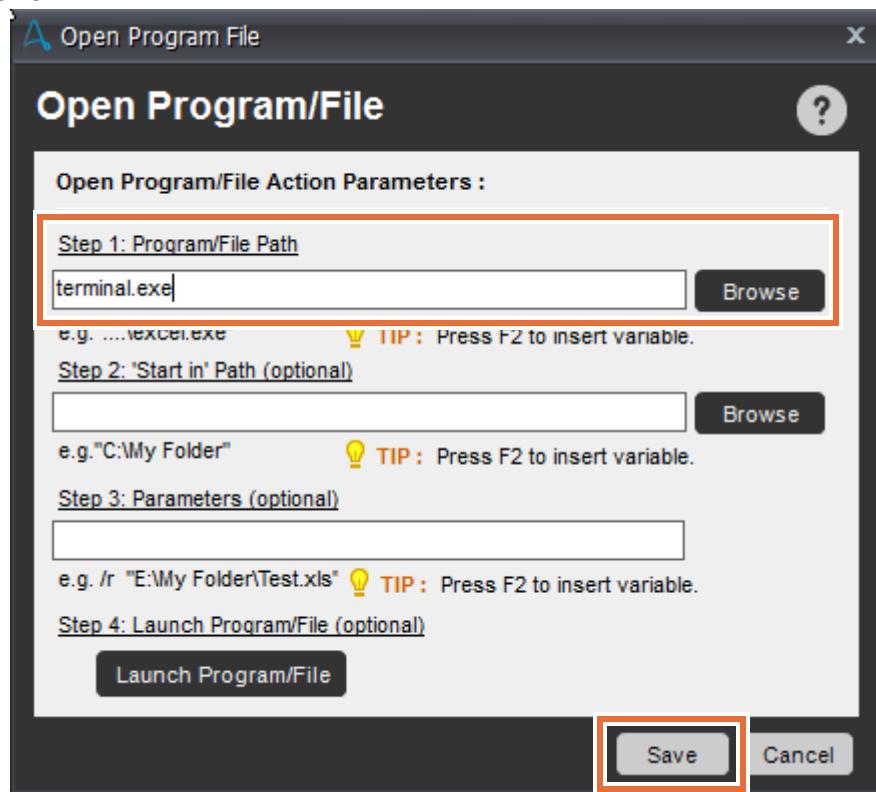
The Send Email command is now in the Actions List. Because it is the most recently added command, it is highlighted.



- ___ 8. Add another comment to the list by dragging the **Comment** command from the **Commands** list to the Actions List.
- Go to the **Commands** list, and drag **Comment** to the white space in the Actions List.
 - Enter some text in the **Enter Comment** field, such as:
This comment is a second practice comment.
 - Click **Save**.



- Drag the Open Program/File command from the **Commands** list to the Actions List.
- In the Open Program/File window, in the **Step 1: Program/File Path** field, enter:
terminal.exe
- Click **Save**.



The Actions List now has four items: two comments, a Send Email command, and an Open Program command.

The Actions List window displays the following commands:

- 1 Comment: This comment is a practice comment.
- 2 Send Email: Subject "Automation Anywhere finished executing the task."
- 3 Comment: This comment is a second practice comment.
- 4 Open: "terminal.exe"

Filters at the top: Mouse Moves, Keystrokes, Mouse Clicks, Delays (all checked).

1.5. Editing commands

You can change commands in the Workbench in two different ways:

- You can change the order of the commands by dragging a command to a different location.
- You can change the configuration of the command by editing it.

In the next step, you practice moving a command.

- 1. Move the Open: "terminal.exe" command so that it is at line 3 of the Actions List.
- Click Open: "terminal.exe", which is at line 4 of the Actions List, and keep the mouse button pressed.
 - Drag the Open: "terminal.exe" command to line 2 so that the Send Email command is highlighted.

The Actions List window shows the Open command being moved:

- 1 Comment: This comment is a practice comment.
- 2 Send Email: Subject "Automation Anywhere finished executing the task."
- 3 Open: "terminal.exe" (highlighted with a blue selection bar)
- 4 Comment: This comment is a second practice comment.

Filters at the top: Mouse Moves, Keystrokes, Mouse Clicks, Delays (all checked).

- Release the mouse pointer to move the Open: "terminal.exe" command to line 3 of the Actions List.

The Actions List window shows the Open command moved to line 3:

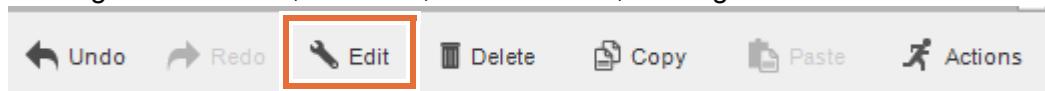
- 1 Comment: This comment is a practice comment.
- 2 Send Email: Subject "Automation Anywhere finished executing the task."
- 3 Open: "terminal.exe"
- 4 Comment: This comment is a second practice comment.

Filters at the top: Mouse Moves, Keystrokes, Mouse Clicks, Delays (all checked).

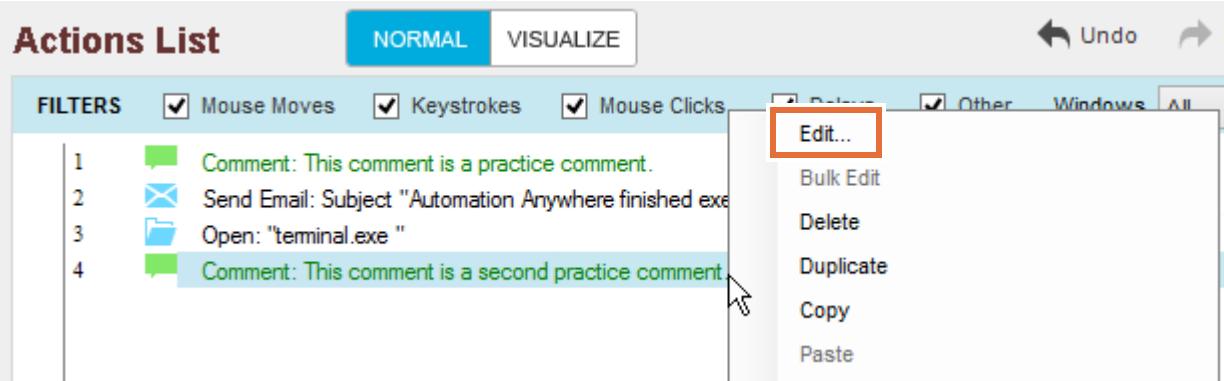
You can open the configuration window for a command by:

- Double-clicking the command

- Selecting the command, and then, on the toolbar, clicking **Edit**



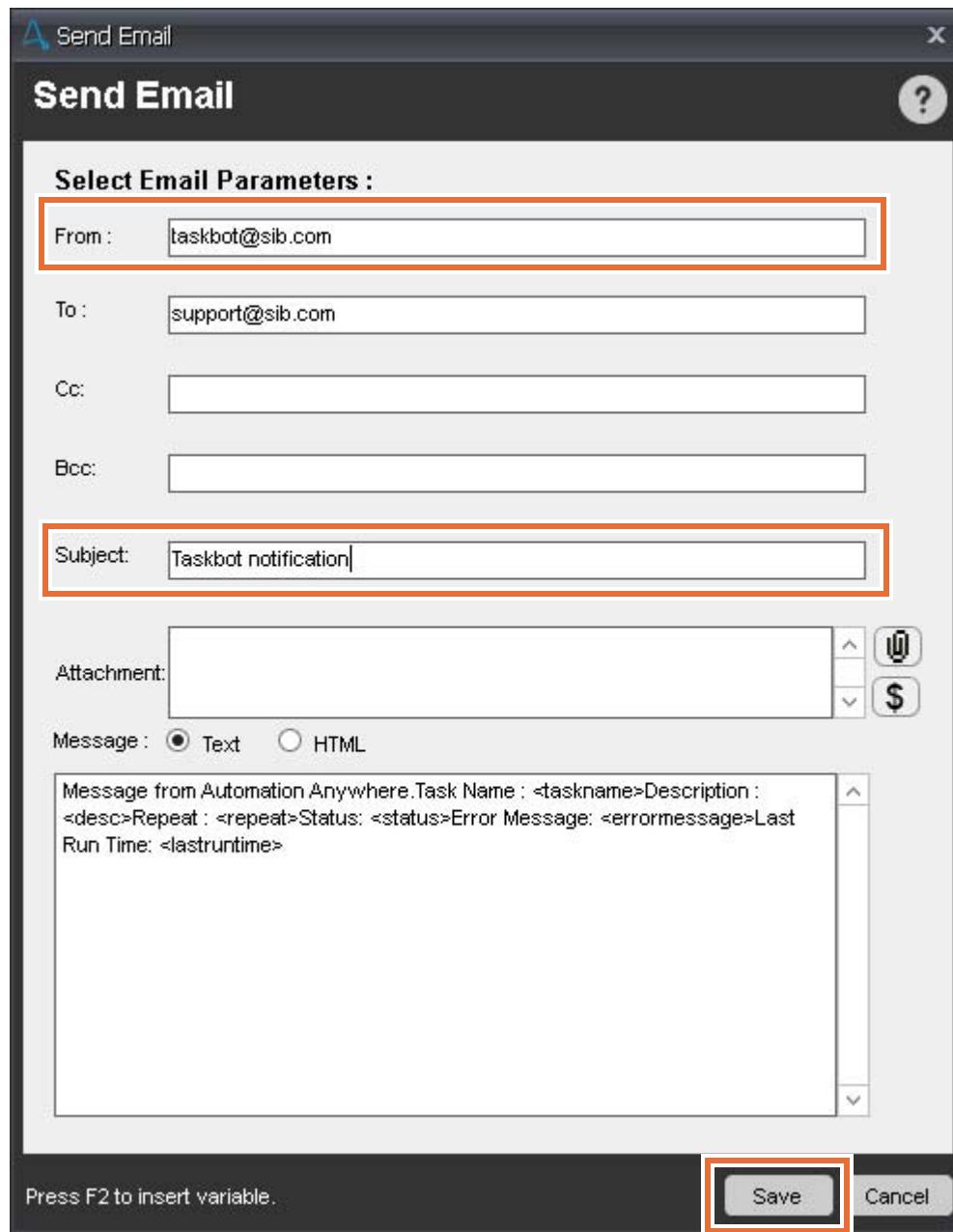
- Right-clicking the command, and clicking **Edit**



In the next steps, you practice editing two of the commands.

2. Edit the **Send Email** command.
 - In the Actions List, double-click the **Send Email** command to open the Send Email window.
 - In the **From** field, delete devUser1@sib.com and enter: taskbot@sib.com
 - In the **Subject** field, delete the default text, and enter: Taskbot notification

- d. Click Save.



You can see the changes to the Send Email command subject line.

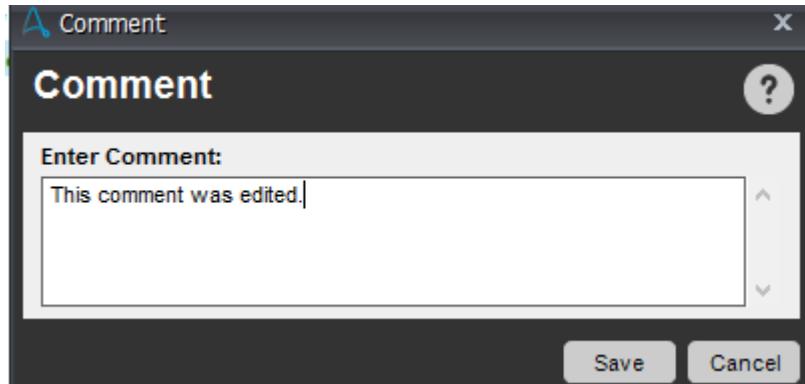
The Actions List interface shows a list of actions:

- 1 Comment: This comment is a practice comment.
- 2 Send Email: Subject "Taskbot notification"
- 3 Open: "terminal.exe"
- 4 Comment: This comment is a second practice comment.

- 3. Edit the comment in line 4.

- a. In the Actions List, select Comment: This comment is a second practice comment.

- ___ b. In the Workbench toolbar, click **Edit**.
- ___ c. In the Comment window, delete the existing comment text and type:
This comment was edited.
- ___ d. Click **Save**.



The edited comment is now in the Actions List.

1.6. Deleting commands

You can delete comments by:

- Selecting the command and then in the Workbench toolbar, clicking **Delete**

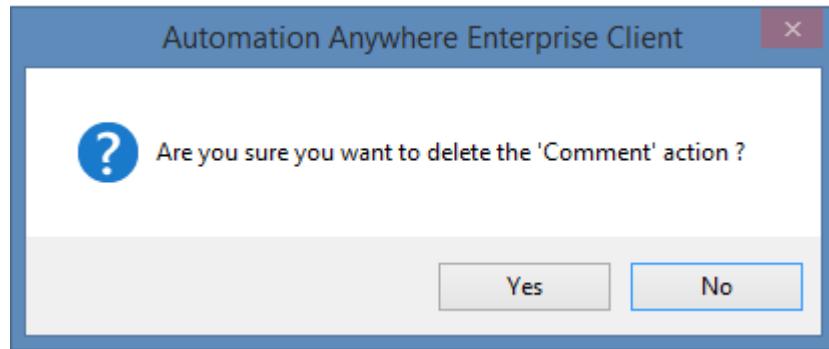


- Right-clicking the command, and then clicking **Delete**

In the next steps, you practice deleting comments from the Actions List.

- ___ 1. Select Comment: This comment is a practice comment, and on the toolbar, click **Delete**.

- __ 2. When asked to confirm the deletion, click **Yes**.



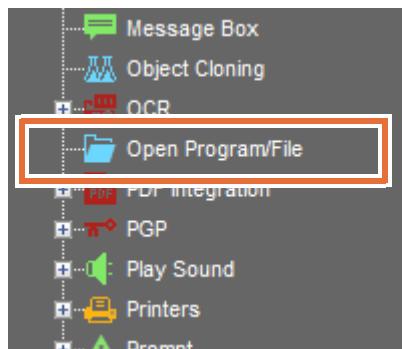
- __ 3. Right-click the **Send Email** command, click **Delete**, and confirm the deletion by clicking **Yes**.
- __ 4. Delete the remaining two commands from the Actions List. The Actions List should now be empty.

Section 2. Automating the creation and saving of a text file

In this section, you create a simple bot that creates a text file, writes some text to it, and saves the file.

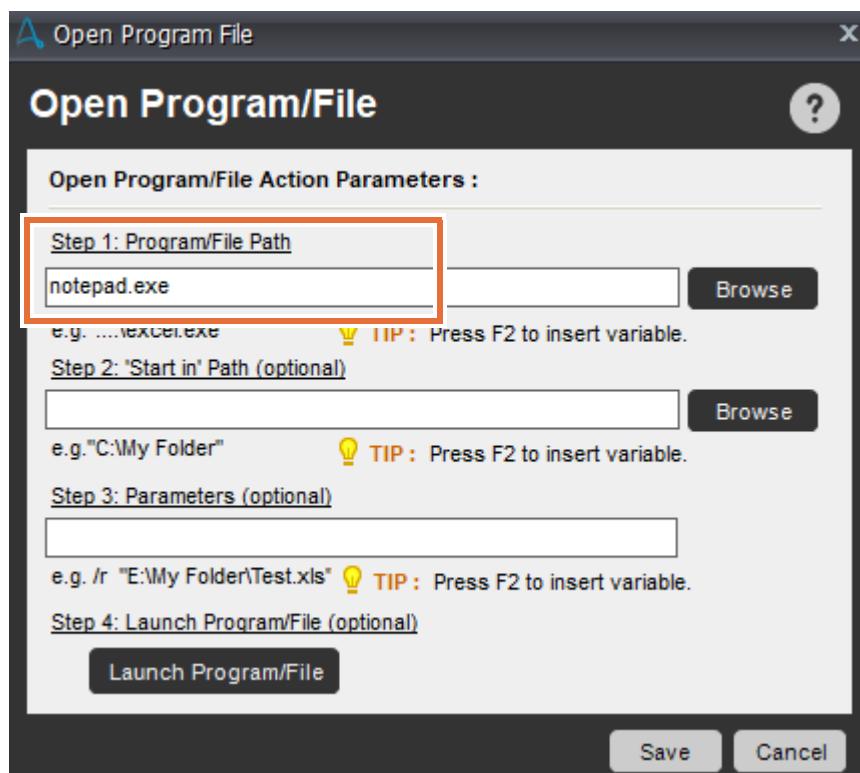
2.1. Automating the opening of Notepad

- 1. In the **Commands** list, double-click **Open Program/File**.



The Open Program/File dialog box opens.

- 2. In the **Step 1: Program/File Path** field, type: notepad.exe



- 3. In the **Step 4: Launch Program/File** section, click **Launch Program/File**, and minimize the Notepad window.

This step verifies that the path in Step 1 is correct. You use the open Notepad window in the next step.

- ___ 4. Click **Save**.

2.2. Automating the writing of text to Notepad

To automate writing text, you use the `Insert Keystrokes` command, which provides a keyboard for you to choose what to type.

- ___ 1. From the **Commands** list, add the `Insert Keystrokes` command to the Actions List.
The Insert Keystrokes dialog box opens.
 - ___ 2. In the **Select Window** list, choose **Untitled - Notepad**.
-

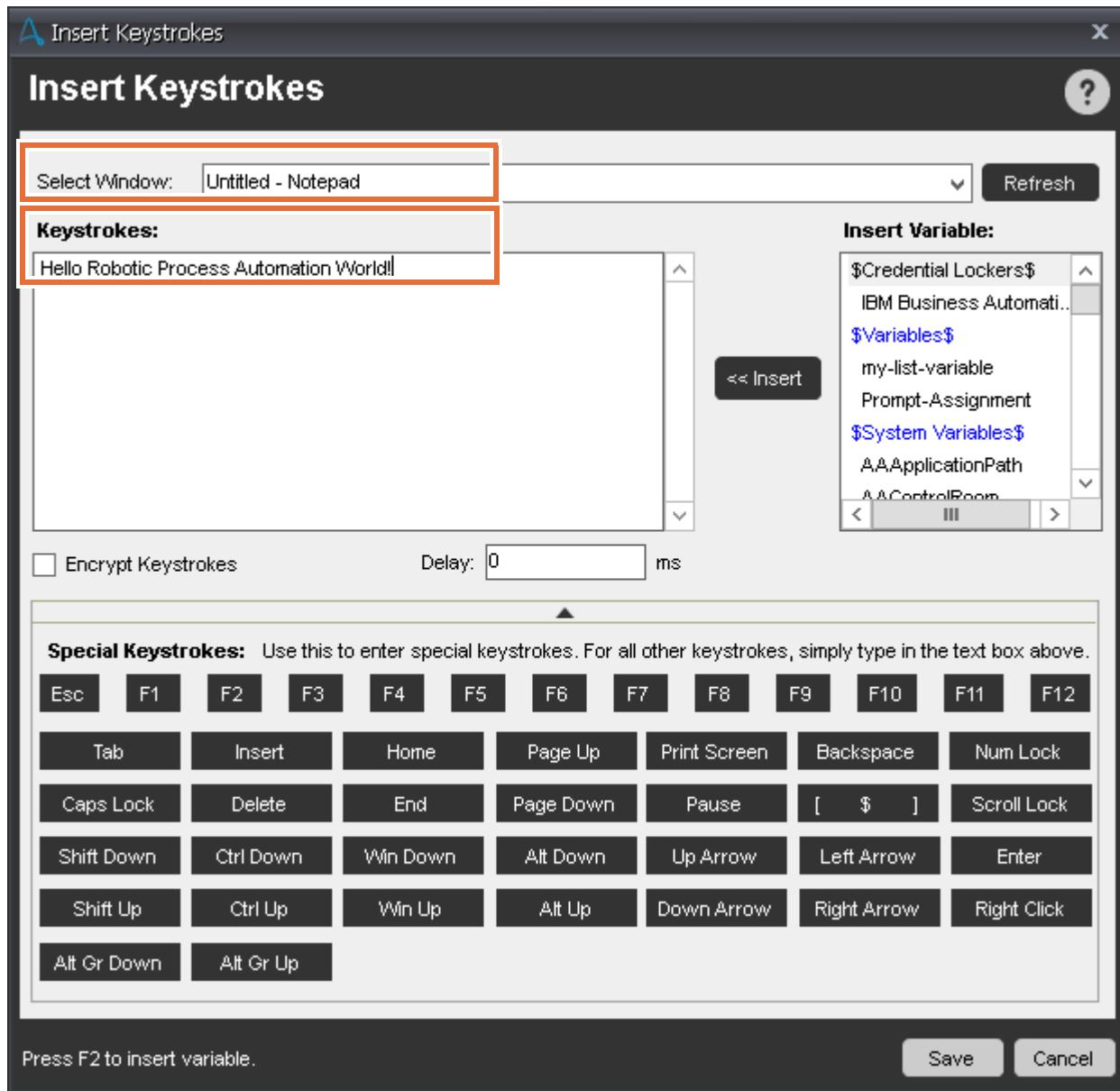


Note

If you do not see **Untitled - Notepad** in the Select Window list, you can click **Refresh**.

- ___ 3. In the **Keystrokes** field, type:

Hello Robotic Process Automation World!



- ___ 4. Click **Save**.

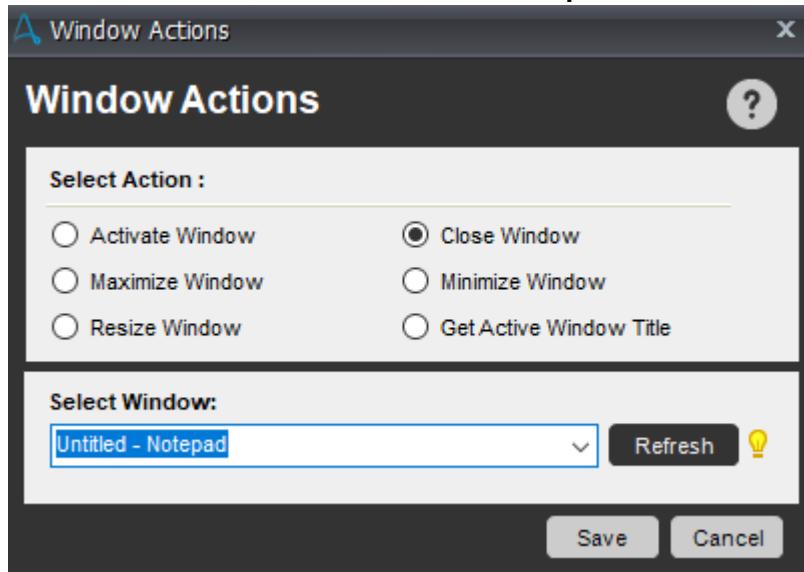
2.3. Automating the saving of the text file

In this section, you use a **Window Action** command to close the Notepad window, which triggers a save window. You use an **Insert Keystrokes** command to simulate pressing S, which opens a **Save As** window where you must first enter a file name. You then add another **Insert Keystrokes** command to give the file a name and to save the file.

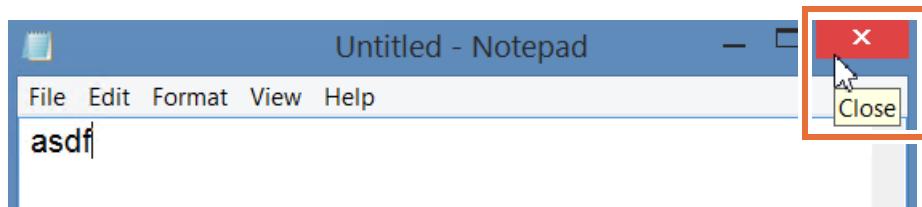
- ___ 1. Automate closing the Notepad window.

- ___ a. In the **Commands** list, double-click **Window Actions** to expand the section, then double-click the **Close Window** subcommand.

- ___ b. In the **Select Action** options of the Window Actions dialog box, make sure that **Close Window** is selected.
- ___ c. In the **Select Window** list, choose **Untitled - Notepad**.



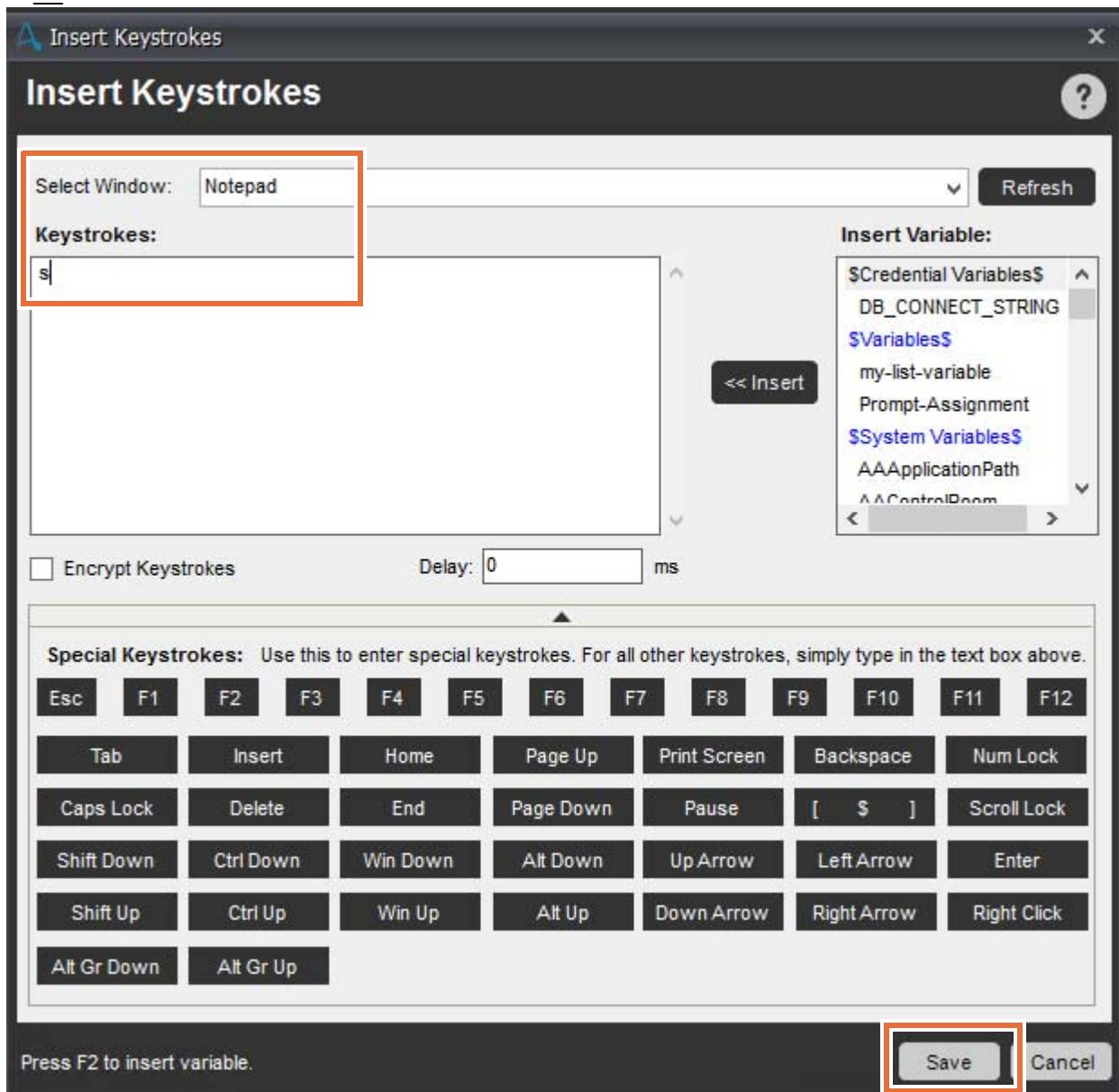
- ___ d. Click **Save**.
- ___ 2. Close the Notepad window to open the “Save as” dialog box for Notepad.
 - ___ a. Go to the Notepad window, and type some text.
 - ___ b. Click the **X** to close the Notepad window.



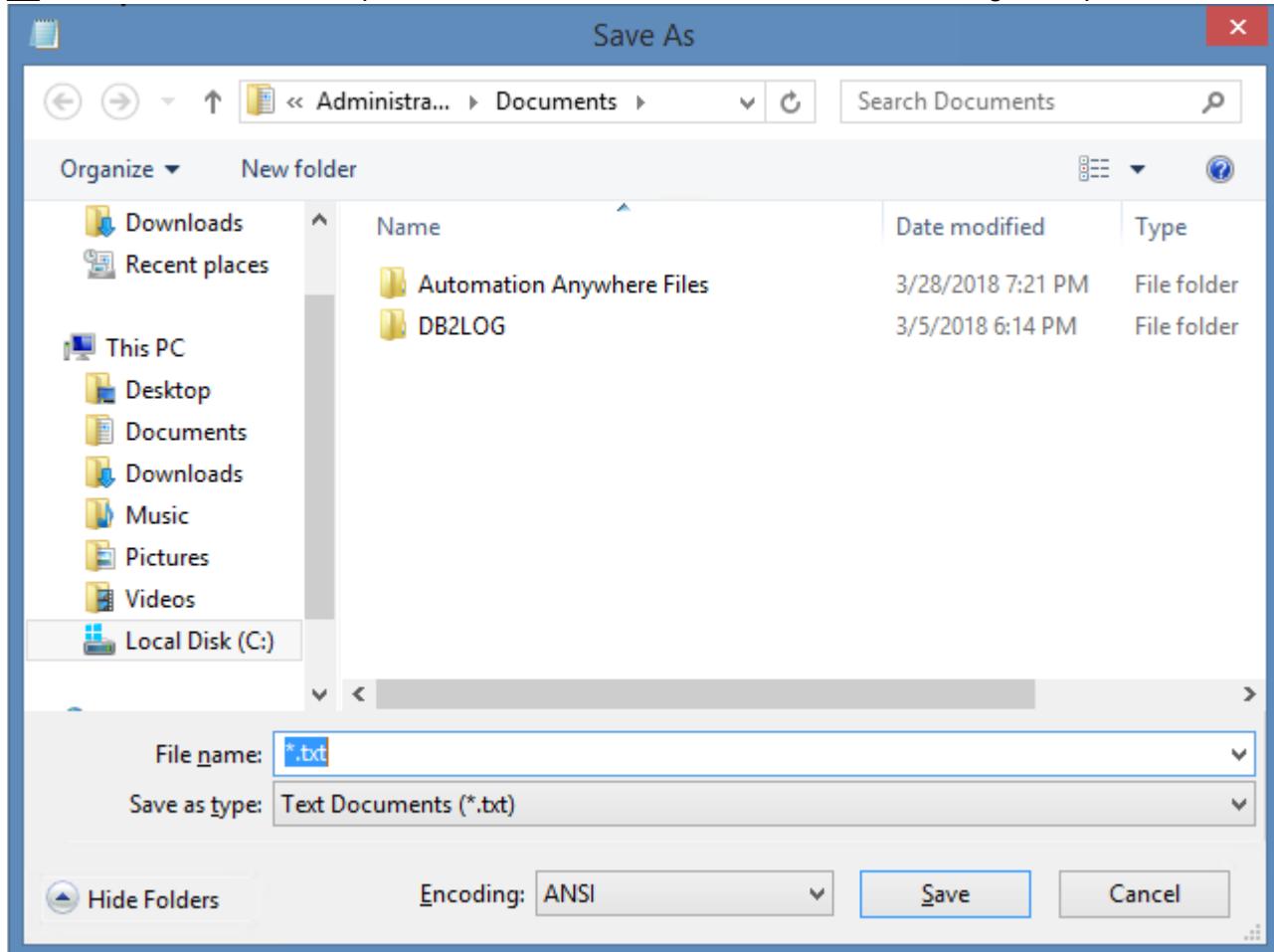
The Notepad Save dialog box opens.

- ___ c. Leave the Notepad window open, and return to the Actions List.
- ___ 3. Configure the **Insert Keystrokes** command to save the text file.
 - ___ a. In the Commands list, double-click **Insert Keystrokes**.
 - ___ b. In the **Select Window** field, choose **Notepad**.
 - ___ c. In the **Keystrokes** field, type: s

__ d. Click Save.

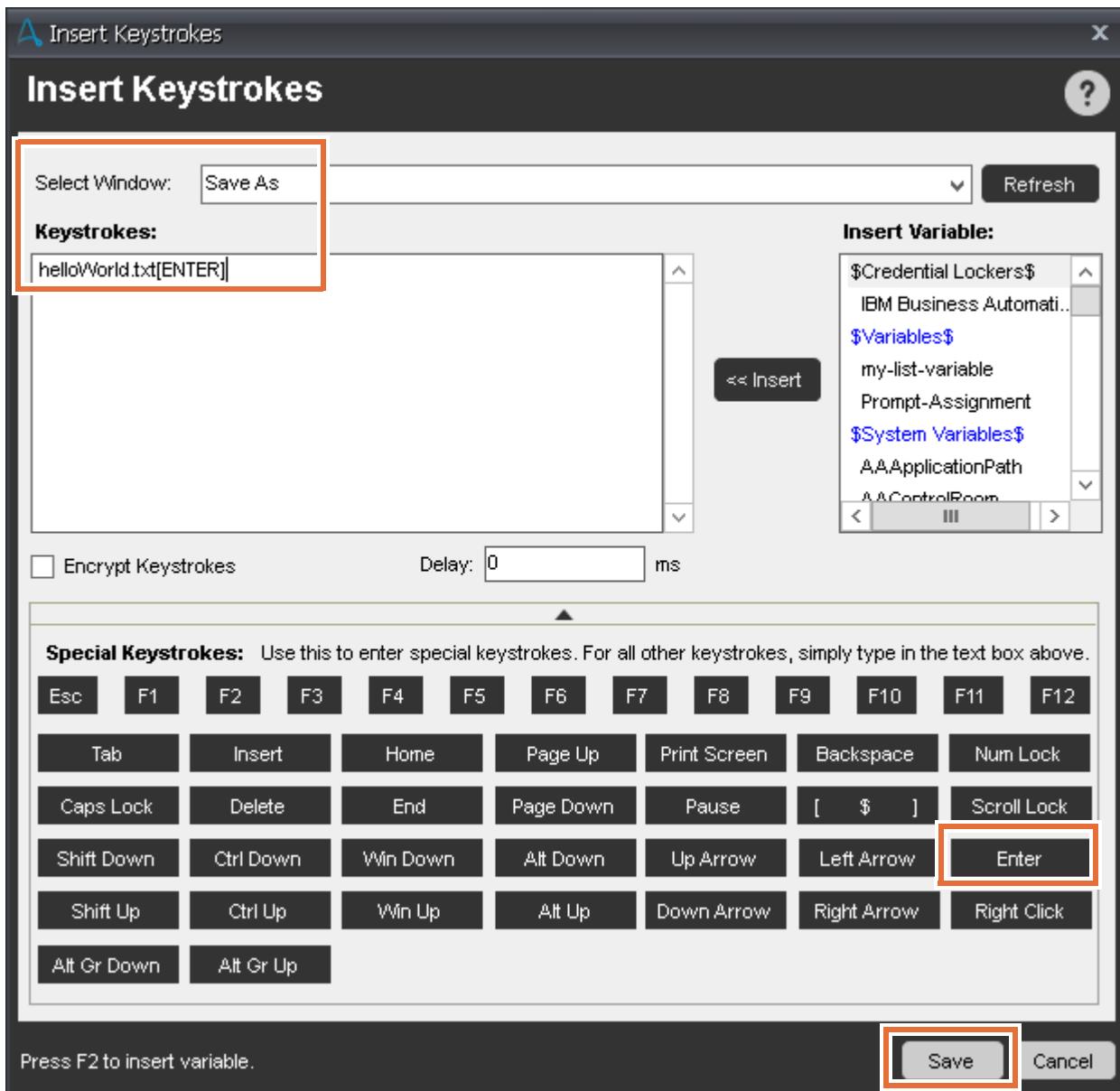


- ___ 4. Go back to the Notepad window, and click **Save**. The “Save as” dialog box opens.

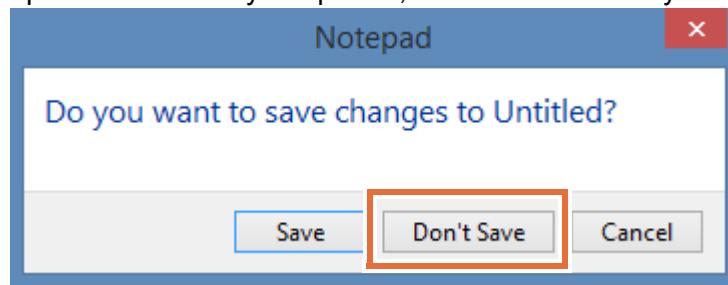


- ___ 5. Use the **Insert Keystrokes** command to enter a name for the Notepad text file so that you can save it.
- ___ a. Returning to the Workbench, in the **Commands** list, double-click **Insert Keystrokes**.
 - ___ b. In the **Select Window** field, choose **Save As**.
 - ___ c. In the **Keystrokes** field, type: `helloWorld.txt`
 - ___ d. In the **Special Keystrokes** section, click **Enter**.

__ e. Click **Save**.



- __ 6. Return to the Notepad Save As window, and click **Cancel**.
- __ 7. Close the Notepad window that you opened, but do not save it by clicking **Don't Save**.

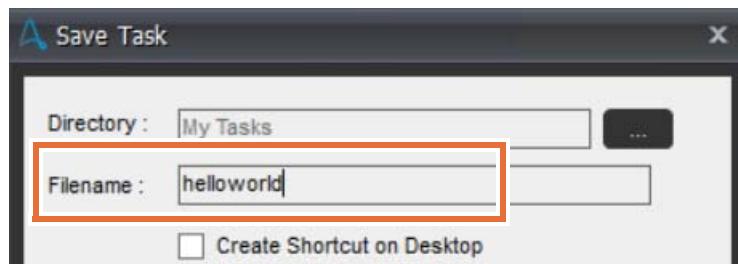


___ 8. Save your work.

___ a. On the toolbar, click **Save**.



___ b. In the **Filename** field, enter: helloworld



___ c. Click **Save**.

This screen capture shows the completed Hello World bot.

Actions List	NORMAL	VISUALIZE			
FILTERS	<input checked="" type="checkbox"/> Mouse Moves	<input checked="" type="checkbox"/> Keystrokes	<input checked="" type="checkbox"/> Mouse Clicks	<input checked="" type="checkbox"/> Delays	<input checked="" type="checkbox"/> Other
1	Open: "notepad.exe "				
2	Keystrokes: Hello Robotic Process Automation World! in "Untitled - Notepad"				
3	Close Window: "Untitled - Notepad"				
4	Keystrokes: s in "Notepad"				
5	Keystrokes: helloWorld.txt[ENTER] in "Save As"				

2.4. Running the bot

___ 1. On the toolbar, click **Run**.



When you run the bot, you should see these actions:

- Notepad opens
- Text is written in Notepad
- Notepad windows open to save the file and provide a file name
- The `helloWorld.txt` file is saved to the Desktop

___ 2. Go to the desktop and open the `helloWorld.txt` file to confirm the contents.

___ 3. Close the notepad file.

___ 4. Close the Workbench to close the helloworld bot.

End of exercise

Exercise review and wrap-up

In this exercise, you signed in to IBM Robotic Process Automation with Automation Anywhere Enterprise Client and explored the user interface. You then learned how to add, edit, and delete commands in the Workbench. You also learned how to create a simple bot.

Exercise 3. Writing data from a text file to an Excel spreadsheet

Estimated time

01:45

Overview

In this exercise, you learn how to define custom variables, set up a loop, and work with strings to transfer data from one file to another.

Objectives

After completing this exercise, you should be able to:

- Work with the Variable Manager to add custom variables
- Configure a basic loop for iterating through a text file and a spreadsheet
- Use String Operation commands to work with text string data

Introduction

In this exercise, you work as a bot developer to retrieve the transaction data from a text file, and transfer the data to a Microsoft Excel spreadsheet.

Scenario: MyBank sends a list of transactions in a text file to SIB for analysis. The current process at SIB is for a clerk to manually transfer the transaction data to a spreadsheet. The source text file holds a list of transactions. Each transaction is on a separate line. The individual data for each transaction is tab-delimited, and consists of a header (such as `TRANSACTION:`) and information (such as `DEBIT`).

As a bot developer, you must develop a bot that automates these tasks:

- Open and iterate through a text file
- Extract tab-delimited data from the text file to variables that are used as a source string
- Transfer the extracted information to specific columns and rows in the spreadsheet

The exercise includes these sections:

- [Section 1, "Getting started with the Workbench"](#)
- [Section 2, "Working with bot variables"](#)
- [Section 3, "Creating the beginning bot actions"](#)

- [Section 4, "Working with loops"](#)
- [Section 5, "Defining loop actions: Extracting substrings and assigning them to variables"](#)
- [Section 6, "Defining loop actions: Writing data to the transactions.xlsx Excel spreadsheet"](#)
- [Section 7, "Finishing and running the bot"](#)
- [Section 8, "Optional: Viewing the solution file"](#)

Requirements

This exercise requires the computer lab environment that was developed for this course.

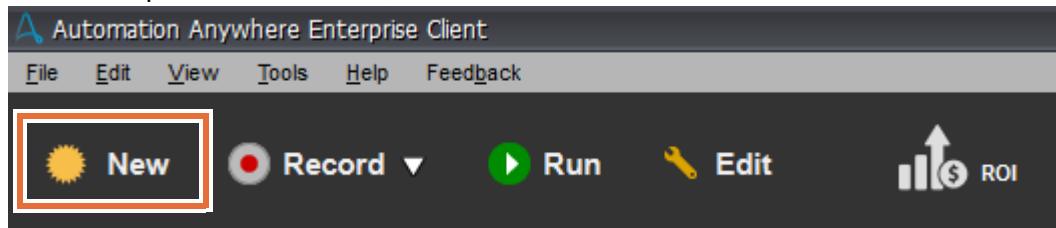
Section 1. Getting started with the Workbench

In this section, you start a new task in the Enterprise Client.

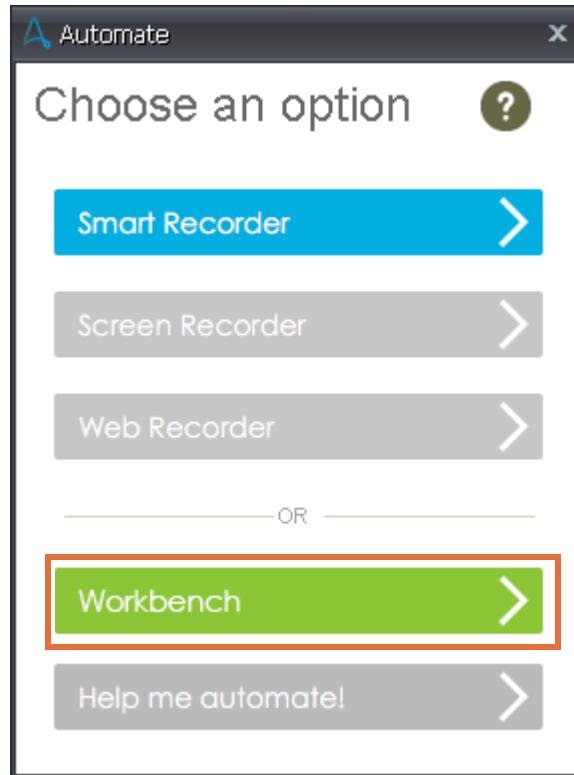
- 1. Go to the Enterprise Client.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1, "Exploring the Enterprise Client"](#) on page 2-2.

- 2. In the Enterprise Client toolbar, click **New**.



- 3. In the Automate window, click **Workbench**.



The Workbench opens to a new, empty task.

Section 2. Working with bot variables

In this section, you learn how to work with bot variables. You review the Enterprise Client Variable Manager and the two different types of variables: user (or local) variables and system variables. You also define the user variables (or local variables) that the bot uses to complete its tasks. The variables that you define in this section hold different data from the text file.

You define the following custom variables:

Variable name	Purpose
vDateString	Holds the DATE: header and date information for each transaction.
vDate	Holds the date information for each transaction.
vAcctNumString	Holds the ACCT_NUM: header and account number for each transaction.
vAcctNum	Holds the account number for each transaction.
vTypeString	Holds the TRANSACTION: header and transaction type information for each transaction.
vType	Holds information about the transaction type for each transaction.
vAmountString	Holds the AMOUNT: header and the monetary amount of each transaction.
vAmount	Holds information about the monetary amount of each transaction.
vRefNumString	Holds the REF_NUM: header and reference number for each transaction.
vRefNum	Holds the reference number for each transaction.
vSpreadsheetRow	Holds the number of the spreadsheet row where the bot writes data. It is initialized to 2, which means that the bot skips row 1, which is a header row, and starts writing data in row 2.



Note

It is a good practice to prefix all local variable names with a lowercase v. This prefix helps distinguish between local variables and other types of variables, such as system variables, when you are developing bots. It is also a good practice to use camel case, such as vDateString, when naming variables.

2.1. Exploring the Variable Manager

1. In the right menu of the Enterprise Client, click **VARIABLE MANAGER**.



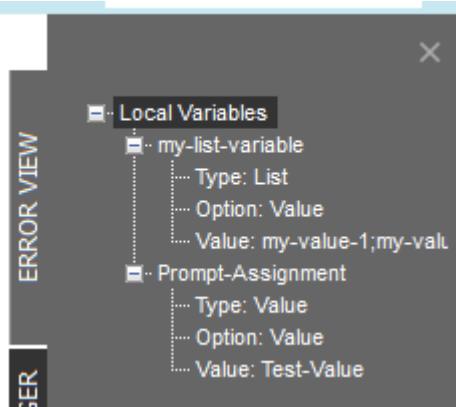
By default, the Variable Manager opens on the Local Variables list.



The Enterprise Client has two predefined local variables:

- my-list-variable: A sample list variable
- Prompt-Assignment: A sample value variable

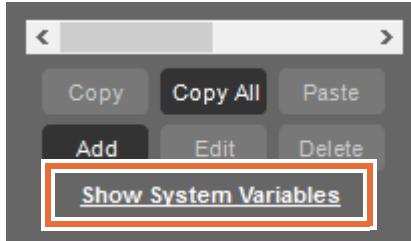
2. In the Local Variables list, expand my-list-variable and Prompt-Assignment.



From the Variables list, you can see the name of the variable, and some of its details such as type and any defined values.

IBM Robotic Process Automation with Automation Anywhere also has a number of predefined system variables, which you view next. You can use system variables in all automation tasks.

- ___ 3. In the Variable Manager, click **Show System Variables**.



The list of available system variables opens, and they are organized by category.

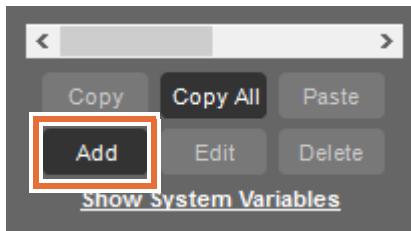
System Variables				
Caption	Name	Return Value	Description	
[+] Date/Time			(This category contains	
[+] Loop			(This category contains	
[+] Excel			(This category contains	
[+] Email			(This category contains	
[+] Trigger			(This category contains	
[+] PDF			(This category contains	
[+] System			(This category contains	

- ___ 4. Explore the available system variables by expanding the categories and browsing through them.
- ___ 5. When you are finished looking through the system variables, close the window by clicking **OK**.

2.2. Creating local variables to hold the tab-delimited text data

The values in these variables are used as the source for the string operation command later in this exercise.

- ___ 1. In the Variable Manager, click **Add**.



- ___ 2. In the Add Variable window, make sure that **Create New Variable** is selected.
- ___ 3. Enter the following information in the form:
 - **Type:** Keep **Value** selected.
 - **Name:** vDateString

- **Select:** Keep **Value** selected.
- **Value:** Leave this field empty.



Information

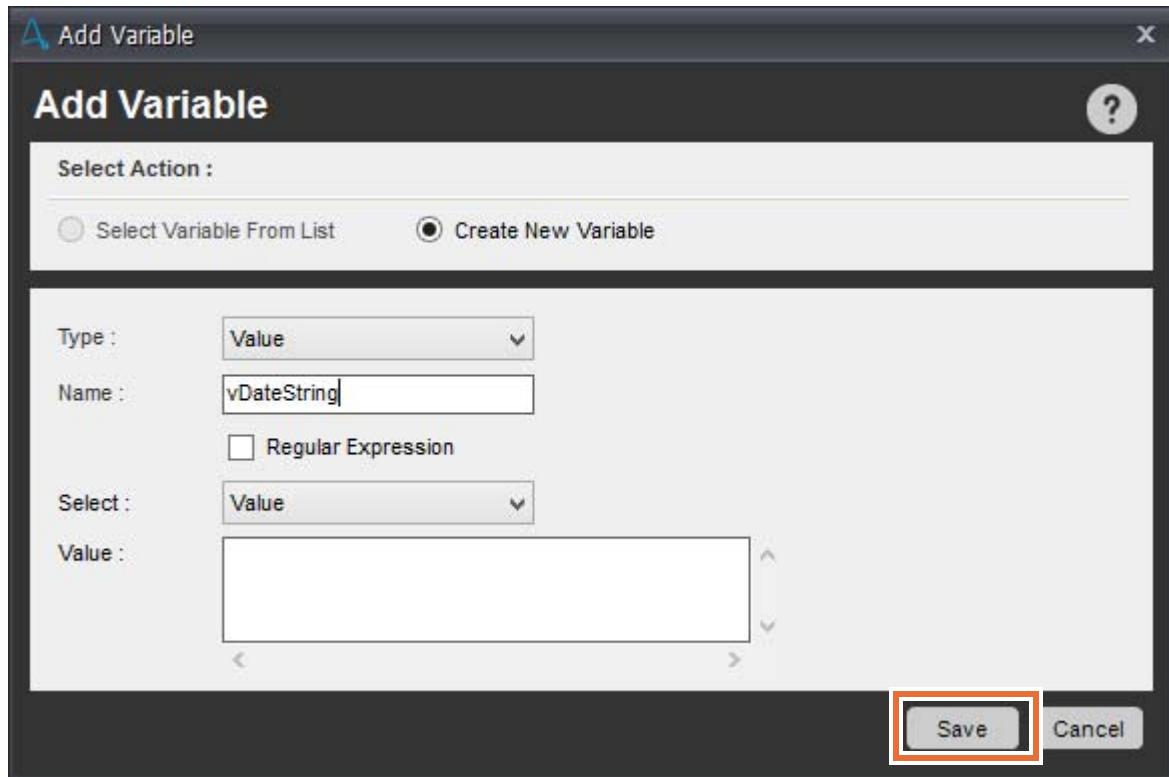
The Enterprise Client identifies variables by enclosing variable names in a pair of dollar signs (\$). For example, in Workbench commands, the vDateString variable is shown as:

\$vDateString\$

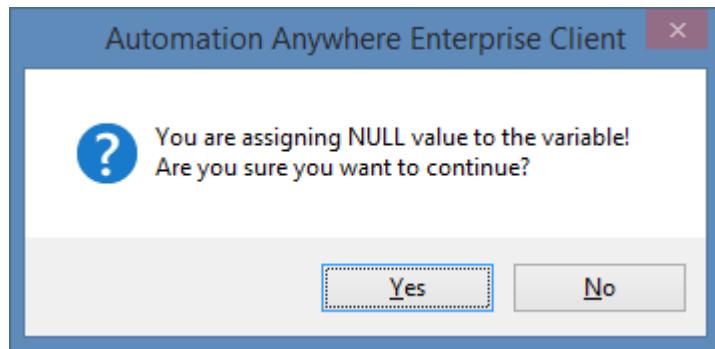
In some cases, such as when you enter variable names into fields, omitting the enclosing dollar signs can cause the bot to not work properly.

You can press F2 to open an Insert Variable window. (Depending on your operating system or computer, you might need to press Fn+F2.) You can use this window to add variable names to bot commands, instead of typing the variable names manually.

-
- 4. Click **Save**.



- ___ 5. In the Automation Anywhere Enterprise Client window, click **Yes** to assign an initial null value to this variable.



The variable is added to the Local Variables list.

- ___ 6. Repeat these variable creation steps to add the following variables:

Name	Type	Value
vAcctNumString	Value	Keep this field empty
vTypeString	Value	Keep this field empty
vAmountString	Value	Keep this field empty
vRefNumString	Value	Keep this field empty

2.3. Defining local variables to hold transaction data

- ___ 1. In the Variable Manager, click **Add**.
- ___ 2. In the Add Variable window, add the vDate variable.
 - ___ a. Make sure that **Create New Variable** is selected.
 - ___ b. Make sure that **Value** is selected as the variable type.
 - ___ c. In the **Name** field, enter: vDate
 - ___ d. Click **Save**.
- ___ 3. Click **Yes** to confirm that you want to assign an initial null value to this variable.
- ___ 4. Repeat these steps to add the following variables:

Name	Type	Value
vAcctNum	Value	Keep this field empty
vType	Value	Keep this field empty
vAmount	Value	Keep this field empty
vRefNum	Value	Keep this field empty

2.4. Adding a counter variable to track spreadsheet rows

You need a counter variable to track the number of loops required to complete the spreadsheet.

- ___ 1. In the Variable Manager, click **Add**.

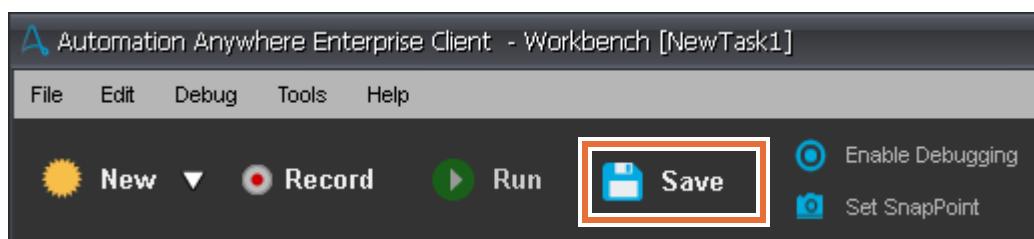
- ___ 2. In the Add Variable window, add the `vSpreadsheetRow` variable.
 - ___ a. Make sure that **Create New Variable** is selected.
 - ___ b. Make sure that the **Type** is set to **Value**.
 - ___ c. In the **Name** field, enter: `vSpreadsheetRow`
 - ___ d. In the **Value** field, enter: `2`

Setting the value to `2` makes sure that the bot starts writing data to the second row of the spreadsheet. The first row is a header row, and should be skipped.

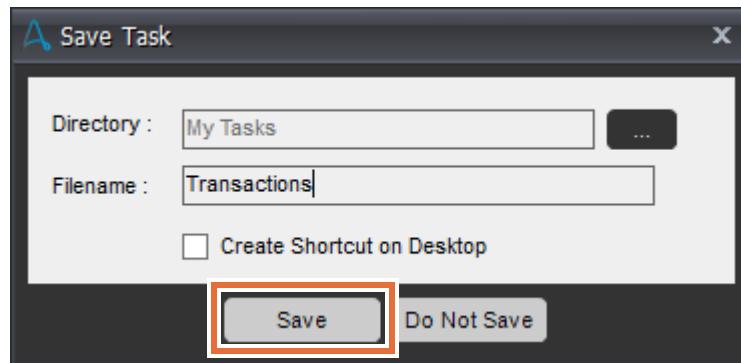
- ___ e. Click **Save**.

2.5. Saving your bot

- ___ 1. On the Workbench toolbar, click **Save** to save your work.



- ___ 2. In the Save Task window, in the **Filename** field, enter: Transactions
- ___ 3. Click **Save**.



Section 3. Creating the beginning bot actions

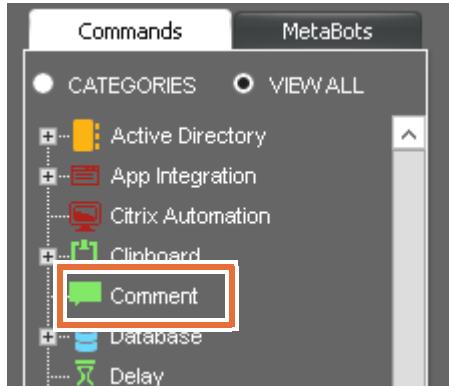
In this section, you code the initial bot actions, including:

- Opening the `transactions.xlsx` spreadsheet
- Setting up the loop that reads data from the `transactions.txt` file

3.1. Adding a comment

___ 1. Add a comment to the Actions List by double-clicking the command from the **Commands** list.

___ a. In the **Commands** list, double-click the **Comment** command.



___ b. In the Comment window, enter:

Open the `transactions.xlsx` Excel spreadsheet

___ c. Click **Save**.



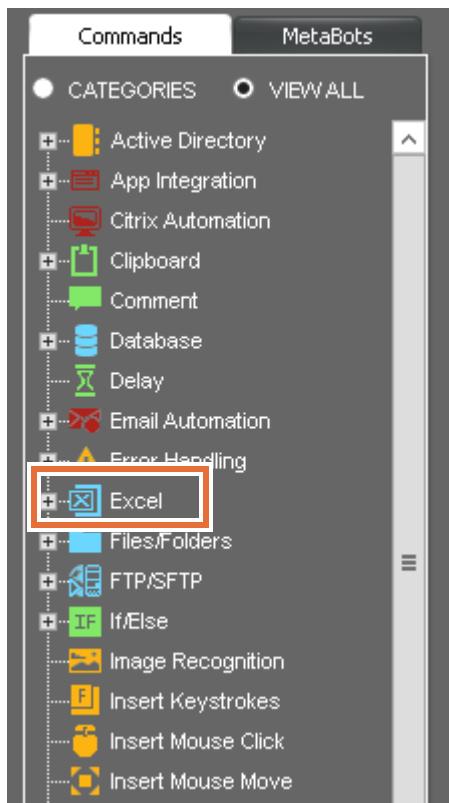
Note

It is a good practice to include explanatory comments in the task code. Comments can help other bot developers understand what the task code is supposed to accomplish.

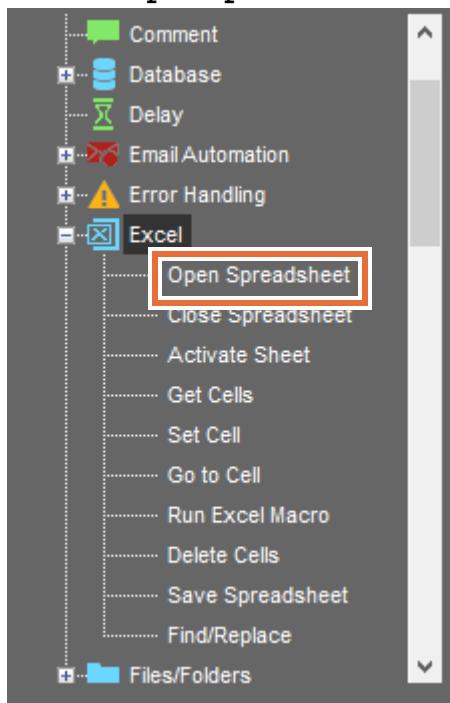
3.2. Automate opening an Excel spreadsheet

Next, you code the bot to open an existing spreadsheet. The spreadsheet for this exercise is in the C:\labfiles\transactions directory.

- ___ 1. In the **Commands** list, double-click the **Excel** section.



- ___ 2. In the **Excel** section, double-click **Open Spreadsheet** to add it to the Workbench.



- ___ 3. In the Excel window, confirm that **Open Spreadsheet** is selected as the Excel command.



Information

You can rearrange commands by selecting a line and dragging it. The command will be placed after the highlighted line in the Workbench.

For example, in this screen capture, the **Comment** line is dragged from line 4 in the code. The **Excel: Open Spreadsheet** command is highlighted on line 2. When the mouse is released, the **Comment** line is dropped at line 3, and the **Resize Window** command moves to line 4.

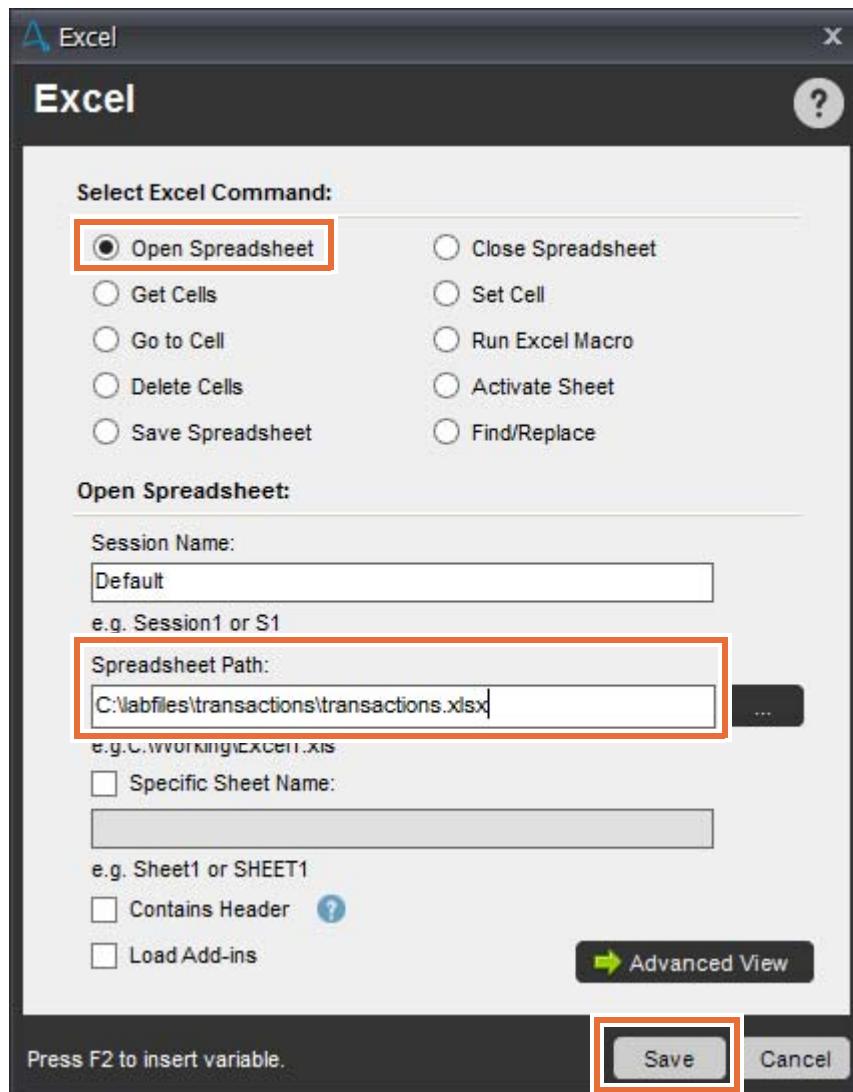
```
1 Comment: Open the transactions.xlsx Excel spreadsheet
2 Excel: Open Spreadsheet "C:\labfiles\transactions.xlsx"
3 Comment: Read from the transactions.txt file
4 Comment: Read from the transactions.txt file
```

- ___ 4. In the Open Spreadsheet section, in the **Spreadsheet Path** field, enter:

C:\labfiles\transactions\transactions.xlsx

Alternatively, you can click **Browse** (ellipses [...]) to select the spreadsheet. After clicking **Browse**, go to C:\labfiles\transactions, select transactions.xlsx, and click **Open**.

- ___ 5. Click **Save**.



3.3. Resizing and moving the Excel window

You add a command to resize and place the Microsoft Excel window in a specific location on the monitor screen. To code this manually, you must first open the Excel window, then, select the commands from the **Commands** list to manipulate that window.

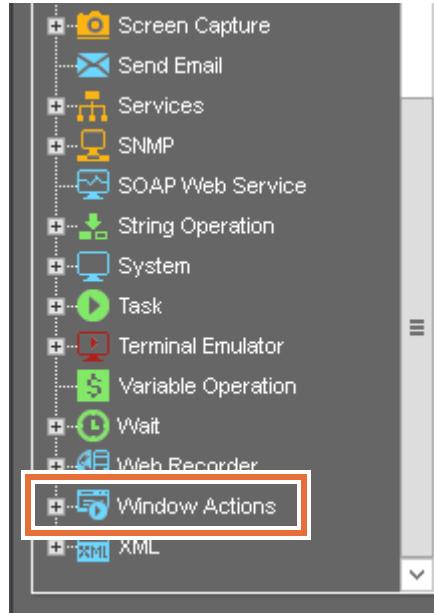


Note

It is a good practice to maximize application windows to ensure that the application interface elements are in a consistent location. However, in this exercise, you set the Excel window to a specific size and location so that you can watch the bot actions and variable definitions when you run the bot.

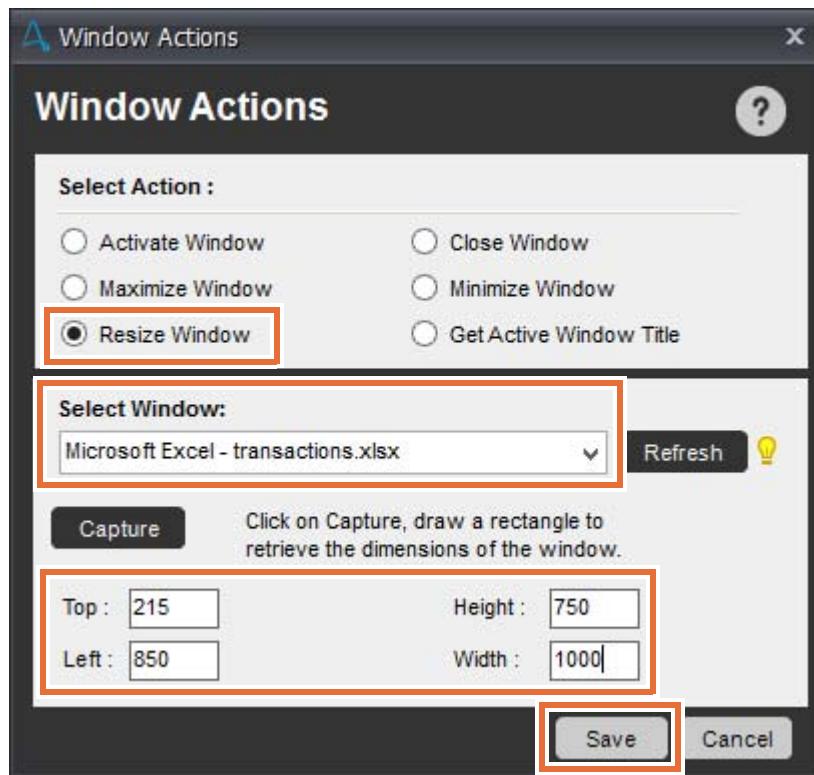
- ___ 1. In Windows Explorer, go to the C:\labfiles\transactions directory, and double-click transactions.xlsx to open it.

- __ 2. If you see a Microsoft Office activation prompt, click **Next** to activate the license over the internet.
- __ 3. Return to the Workbench and in the **Commands** list, double-click **Window Actions** to expand it.



- __ 4. Double-click the **Resize Window** command to add it to the Workbench.
- __ 5. In the Window Actions window, confirm that **Resize Window** is the selected action.
- __ 6. From the **Select Window** list, select **Microsoft Excel - transactions.xlsx**.
- __ 7. Enter the following values:
 - **Top:** 215
 - **Left:** 850
 - **Height:** 750
 - **Width:** 1000

___ 8. Click **Save**.



___ 9. Close the transactions.xlsx Excel spreadsheet.

Section 4. Working with loops

In this section, you work with the **Read From CSV/Text** command, which uses a loop to read text from a file.

Then, you define the loop actions. For this task, the command reads data until it reaches a tab and extracts that data to a system variable, then restarts extracting data until it reaches the next tab. The tab-delimited data is read and extracted until the end of the line. This command *loops* or repeats for each line in the file.

4.1. Defining a loop to read from a text file

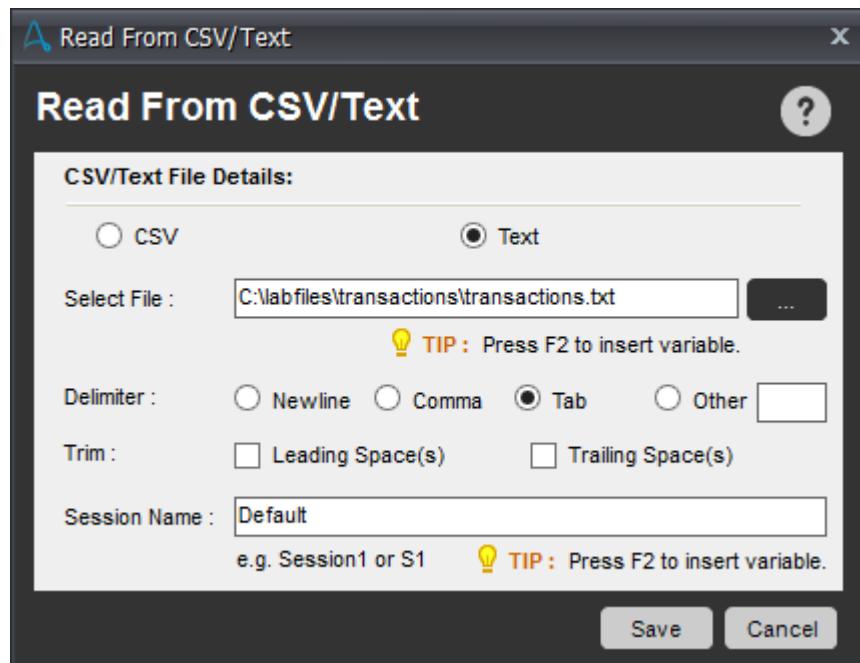
Next, you code the bot to read from a text file by using the **Read From CSV/Text** command. This command uses a loop to extract the data from each line of the file.

- ___ 1. Add a comment to introduce the next bot task of reading the text file.
 - ___ a. In the **Commands** list, double-click the **Comment** command to add it to the Actions List.
 - ___ b. In the **Comment** field, enter:
Read from the transactions.txt file
 - ___ c. Click **Save**.
- ___ 2. In the **Commands** list, double-click the **Read From CSV/Text** command.



- ___ 3. In the Read From CSV/Text window, select **Text**.
- ___ 4. In the **Select File** field, enter:
C:\labfiles\transactions\transactions.txt
You can also use **Browse** (ellipses [...]) to select transactions.txt from the C:\labfiles\transactions directory.
- ___ 5. For the **Delimiter** option, select **Tab**.
- ___ 6. Leave the remaining options (**Trim**, **Session Name**) at their default settings.

- ___ 7. Click **Save**.



A set of loop commands (Start Loop and End Loop) is automatically added to the Workbench as part of the Read From Text File command.

- 5 Read From Text File: "C:\labfiles\transactions\transactions.txt" Delimiter: 'Tab' Header: 'No' Truncate: 'Yes'
- 6 Start Loop "Each row in a CSV/Text file of Session: Default"
- 7 Comment: Use \$Filedata Column\$ variable for each record in File.
- 8 End Loop



Note

Notice the default comment that is entered into the Loop commands:

Use \$Filedata Column\$ variable for each record in File.

\$Filedata Column\$ is a system variable. You use this variable to assign data from the text file to the string local variables that you defined in [Section 2, "Working with bot variables"](#).

- ___ 8. Click the default loop Comment command and add another comment that reads:

Assign tab-delimited text file data to the string variables

4.2. Assigning the extracted data to variables

In this section, you use the Variable Operation command inside the Loop command to assign data to your local variables.

Each line in the transactions.txt file has five tabbed sets of data:

- DATE
- ACCOUNT_NUM
- TYPE
- AMOUNT
- REF_NUM

Each tab is treated as a column. The data in each column is assigned to the system variable \$Filedata.Column().\$. You specify the column number in the \$Filedata.Column()\$ system variable to define which set of data to assign. For example, the data in column 1 is assigned to \$Filedata.Column(1)\$.

In the transactions.txt file, the first line includes the following data:

DATE: 01-26-2018 ACCT_NUM: 87431 TYPE: DEBIT AMOUNT: 30000 REF_NUM: 0109

In this example:

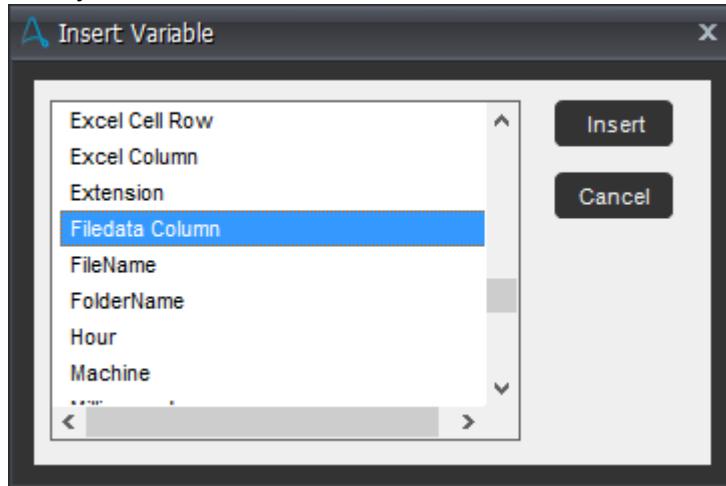
- The first tab is after “DATE: 01-26-2018” so this data is in column 1.
- In the first loop, the string DATE: 01-26-2018 is read from the file and assigned to the system variable \$Filedata.Column(1)\$.

Next, you can assign the contents of \$Filedata.Column(1)\$ to your local variable, the vDateString variable, where it is held until you extract the date as a substring later in this exercise.

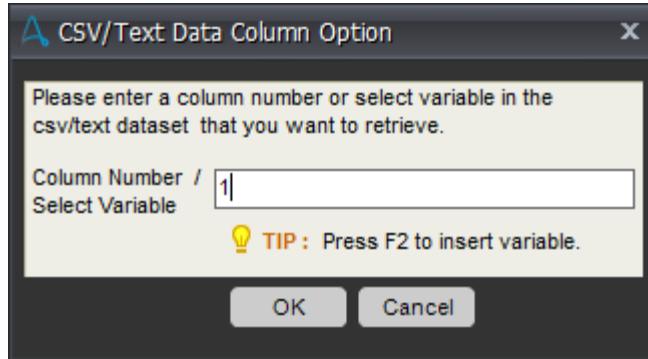
To assign the variables:

1. Ensure that the comment **Assign tab-delimited text file data to the string variables** is selected.
2. From the **Commands** list, double-click the **Variable Operation** command.
The Variable Operation window opens.
3. Make sure that **User Variables** is selected.
4. From the **Specify Variable** list, select vDateString.
5. In the **Specify value** field, set the field to \$Filedata.Column(1)\$.
 - a. Make sure that the cursor is in the field.
 - b. Press F2 (or Fn+F2 on Windows 10) to open the **Insert Variable** window.

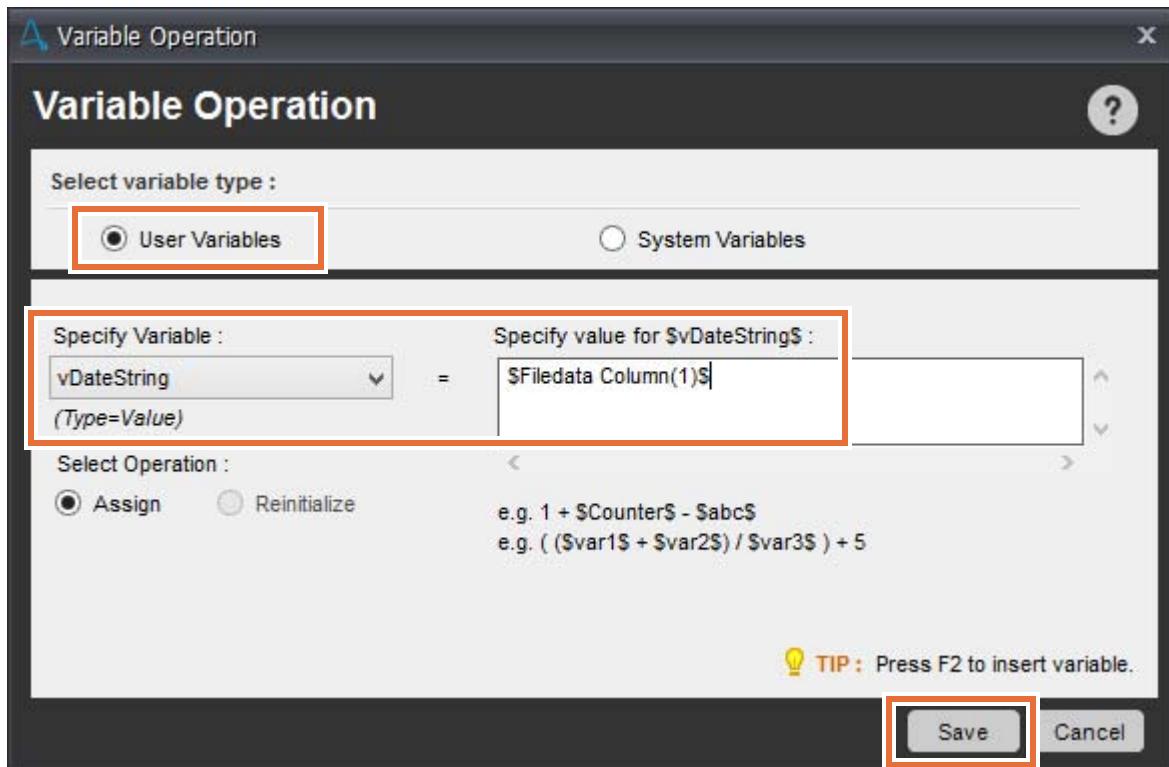
- ___ c. Scroll to the \$System Variables\$ section, and click **Filedatal Column**.



- ___ d. Click **Insert**.
___ e. In the Column Number field, type 1, and click **OK**.



- ___ 6. Click **Save**.



- ___ 7. Make sure that the Variable Operation is added after the *Assign tab-delimited text file data to the string variables* comment.

```

1  Comment: Open the transactions.xlsx Excel spreadsheet
2  Excel: Open Spreadsheet "C:\Labfiles\transactions\transactions.xlsx". ActiveSheet: "Default". Session: Default
3  Resize Window: "Microsoft Excel - transactions.xlsx" with dimensions Top: 215, Left: 850, Height: 750, Width: 1000
4  Comment: Read from the transactions.txt file
5  Read From Text File: "C:\Labfiles\transactions\transactions.txt" Delimiter: 'Tab' Header: 'No' Trim Leading Space: "No" T
6  Start Loop "Each row in a CSV/Text file of Session: Default"
7  Comment: Use $Filedata Column$ variable for each record in file.
8  Comment: Assign tab-delimited text file data to the string variables
9  Variable Operation: $Filedata Column(1)$ To $vDateString

```

- ___ 8. Repeat [Step 1](#) through [Step 6](#) to add the following Variable Operation commands to the Workbench.

Specify Variable	Specify Value
vAcctNumString	\$Filedata Column(2)\$
vTypeString	\$Filedata Column(3)\$
vAmountString	\$Filedata Column(4)\$
vRefNumString	\$Filedata Column(5)\$

Make sure that you add each Variable Operation command after the previous command so that they appear in order in the Actions List.

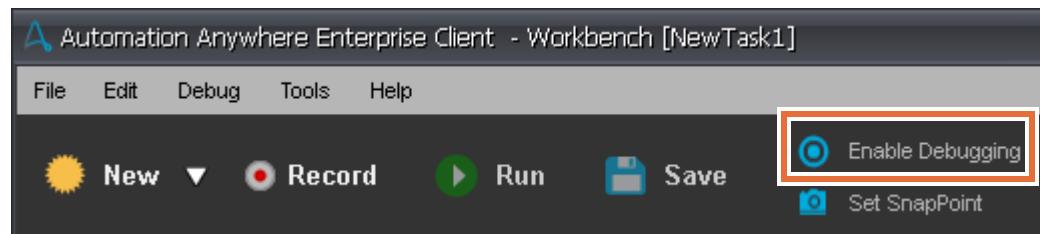
6	Start Loop "Each row in a CSV/Text file of Session: Default"
7	Comment: Use \$Filedata Column\$ variable for each record in File.
8	Comment: Assign tab-delimited text file data to the string variables
9	Variable Operation: \$Filedata Column(1)\$ To \$vDateString\$
10	Variable Operation: \$Filedata Column(2)\$ To \$vAcctNumString\$
11	Variable Operation: \$Filedata Column(3)\$ To \$vTypeString\$
12	Variable Operation: \$Filedata Column(4)\$ To \$vAmountString\$
13	Variable Operation: \$Filedata Column(5)\$ To \$vRefNumString\$

- ___ 9. Save your bot.

4.3. Testing the bot

You can run the bot in Debugging mode to make sure that the bot runs without errors, and that the variables are being assigned with the correct values.

- ___ 1. On the Workbench toolbar, click **Enable Debugging**.

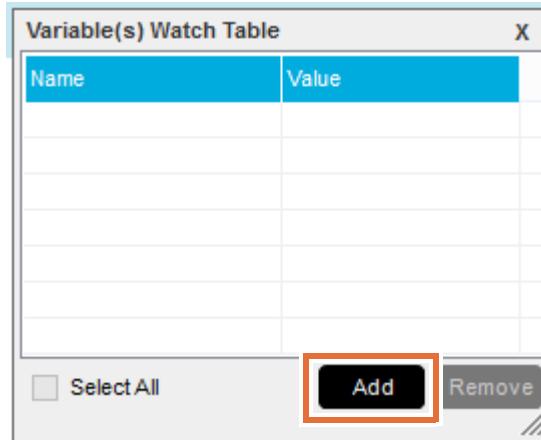


The Variable(s) Watch Table window opens.

- ___ 2. Move the Variable(s) Watch Table to the upper-left area of the Actions List window so that you can watch the table during debugging.

The screenshot shows the 'Actions List' window. At the top, there are two tabs: 'NORMAL' (which is selected) and 'VISUALIZE'. Below the tabs is a section titled 'Variable(s) Watch Table' which contains a table with two columns: 'Name' and 'Value'. At the bottom of this section are two buttons: 'Select All' and 'Add' (highlighted in blue). To the right of the table, there is a vertical scroll bar and some text that appears to be part of the bot's script or log.

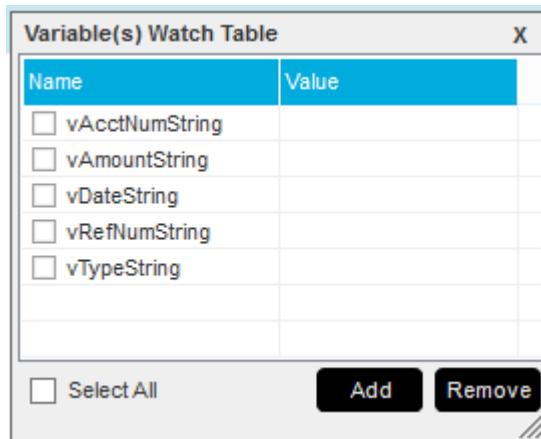
___ 3. In the Variable(s) Watch Table window, click **Add**.



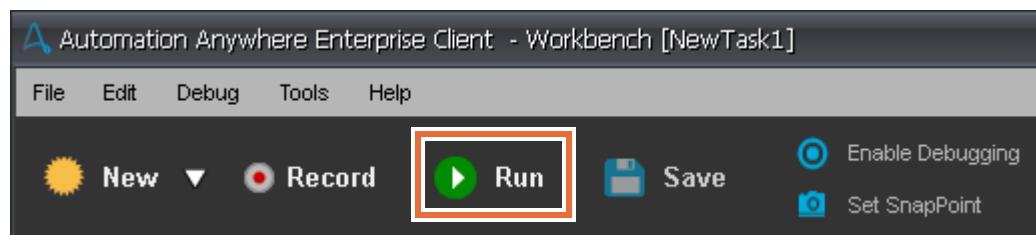
___ 4. In the Add Variable(s) In Watch table, select the following variables, and click **Add**:

- vAcctNumString
- vAmountString
- vDateString
- vRefNumString
- vTypeString

The string variables are now listed in the table.



___ 5. In the Workbench toolbar, click **Run**.



The transactions.xlsx spreadsheet opens, is resized, and is moved to the coordinates that you specified. As the bot runs through the loop, the Variable(s) Watch Table populates with the variable values.

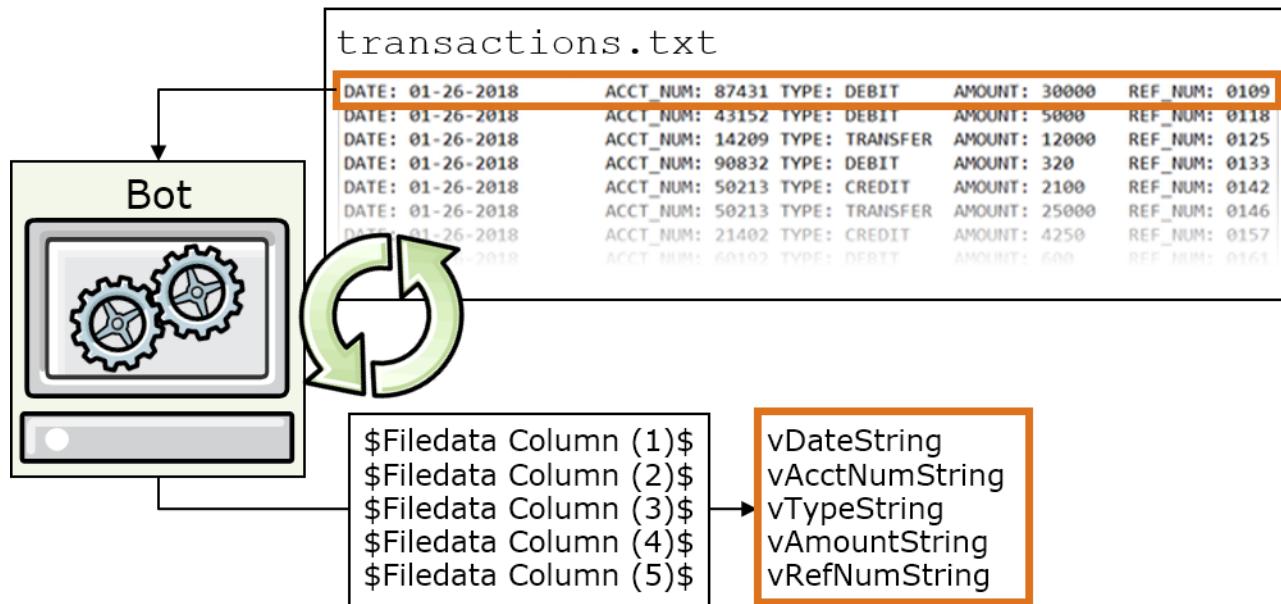
Variable(s) Watch Table	
Name	Value
vAcctNumString	ACCT_NUM: 68291
vAmountString	AMOUNT: 19500
vDateString	DATE: 01-26-2018
vRefNumString	REF_NUM: 0204
vTypeString	TYPE: TRANSFER

Select All Add Remove

- ___ 6. If the bot stops with errors, match the line with the error to its corresponding step in the exercise instructions, and review the steps.

Bot actions:

- The bot enters a loop for iterating through the transactions.txt file
- As the bot goes through each line in the text file, it puts each individual tab-delimited transaction data into the \$Filedata Column()\$ variable
- The \$Filedata Column()\$ data is transferred to the local variables



- ___ 7. Close the transactions.xlsx Excel spreadsheet.
- ___ 8. Close the Variable(s) Watch table, and on the Workbench toolbar, click **Disable Debugging** to turn off Debugging mode.

Section 5. Defining loop actions: Extracting substrings and assigning them to variables

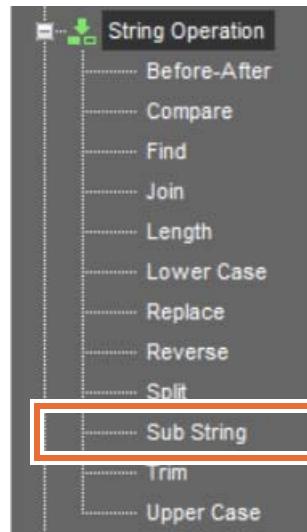
In this section, you use the Sub String String Operation commands to extract data from the string variables. You then assign the substrings to the transaction data variables.

5.1. Extracting substrings

- ___ 1. After the final string variable command, add the following comment:

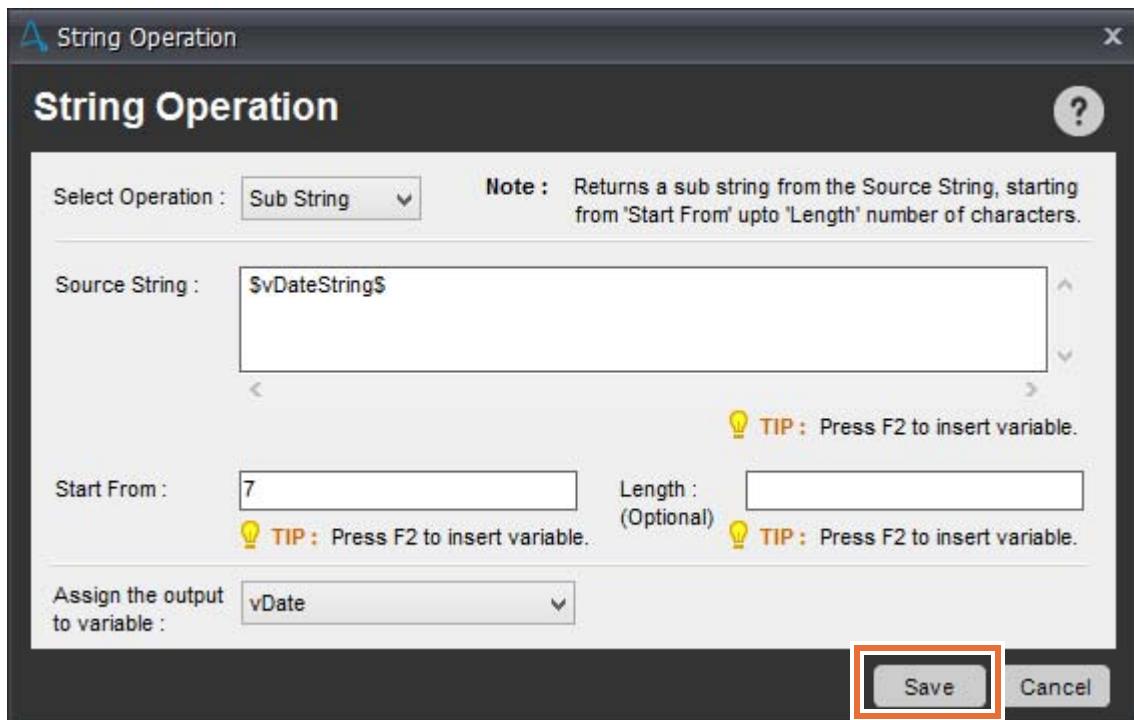
Use the Sub String command to extract transaction data from string variables and assign data to the transaction data variables

- ___ 2. Add a Sub String command that extracts the transaction data from the string variable, and assigns it to its corresponding transaction data variable.
 - ___ a. In the **Commands** list, expand String Operation and double-click the **Sub String** command.



- ___ b. In the String Operation window, confirm that **Sub String** is the selected operation.
- ___ c. In the **Source String** field, enter: \$vDateString\$
You can press F2 (or Fn+F2) to enter the variable through the Insert Variable window.
- ___ d. In the **Start From** field, enter: 7
- ___ e. From the **Assign the output to variable** list, select **vDate**.

- ___ f. Click **Save**.



Information

The **Start From** field indicates the number of characters, including spaces, to skip before reaching the target substring.

Consider this string:

TYPE: DEBIT

The target substring is: DEBIT

This substring starts on the seventh character of the line, so you set the **Start From** value to 7. The Sub String command skips the first 6 characters of the line: "TYPE: " (including the space).

- ___ 3. In the Actions List, make sure that the `Sub String` command is after the `Comment` that you added in Step 1.
- ___ 4. Repeat [Step 2](#) to add the following `Sub String` commands to the Workbench.

Source String	Start From	Assign the output to variable
\$vAcctNumString\$	11	vAcctNum
\$vTypeString\$	7	vType
\$vAmountString\$	9	vAmount
\$vRefNumString\$	10	vRefNum

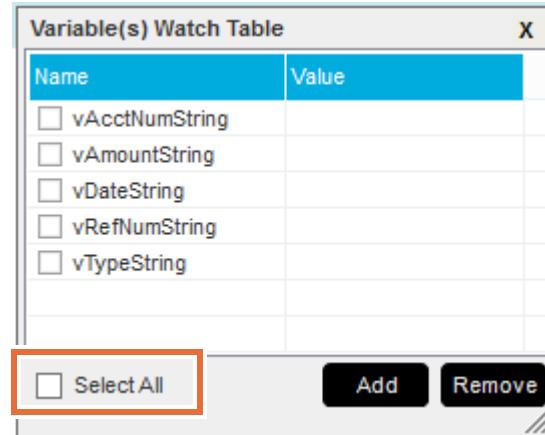
Make sure that each new Sub String command follows the previous command.

Comment: Use the Sub String command to extract transaction data from the string variables and assign data to the transaction data:
 String Operation: Extract substring from "\$vDateString\$" and assign output to \$vDate\$
 String Operation: Extract substring from "\$vAcctNumString\$" and assign output to \$vAcctNum\$
 String Operation: Extract substring from "\$vTypeString\$" and assign output to \$vType\$
 String Operation: Extract substring from "\$vAmountString\$" and assign output to \$vAmount\$
 String Operation: Extract substring from "\$vRefNumString\$" and assign output to \$vRefNum\$

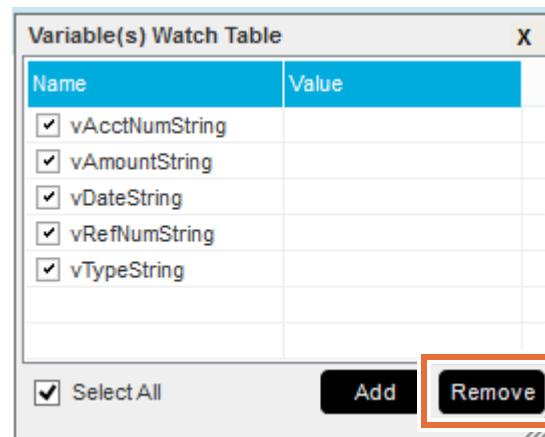
- ___ 5. Save your work.

5.2. Testing the bot

- ___ 1. On the Workbench toolbar, click **Enable Debugging**.
- ___ 2. Remove the string variables from the Variable(s) Watch Table, and add the transaction data variables.
 - ___ a. In the Variable(s) Watch Table, select the **Select All** check box.



- ___ b. Click **Remove**.

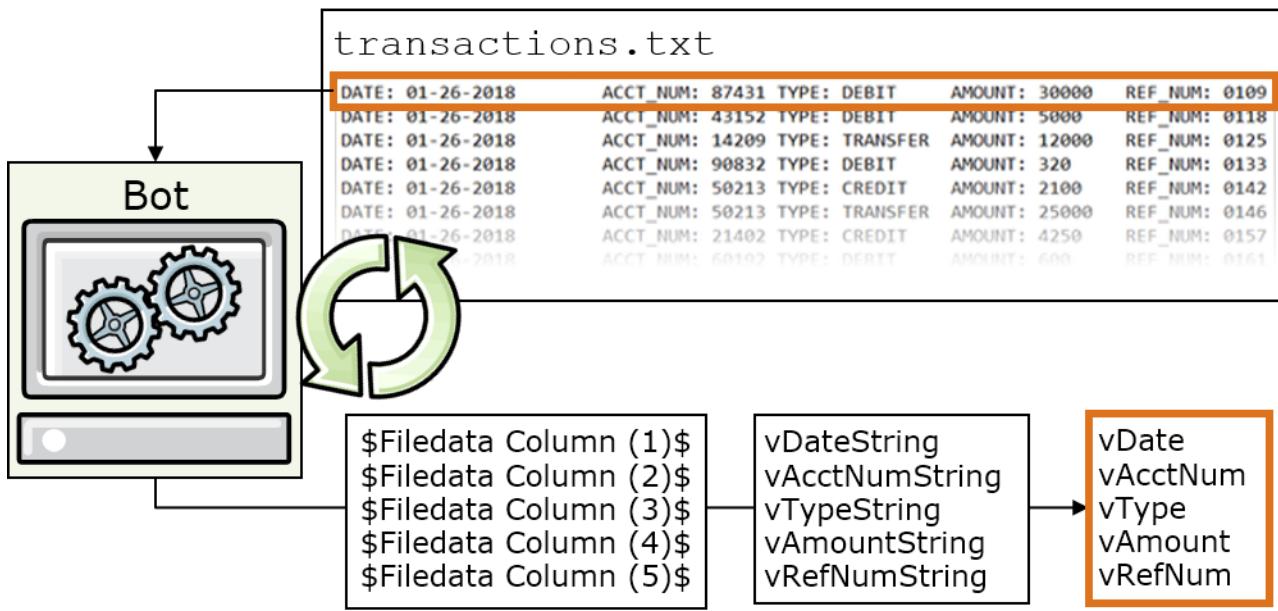


- ___ c. Click **Add**, and in the Add Variable(s) In Watch window, select the following variables:
 - vAcctNum
 - vAmount
 - vDate

- vRefNum
 - vType
- ___ d. Click **Add**.
- ___ 3. Click **Run**.

Bot actions:

- The bot enters a loop for iterating through the `transactions.txt` file.
- As the bot goes through each line in the text file, it puts each individual tab-delimited transaction data into the `$Filedata Column()` variable
- The `$Filedata Column()` data is transferred to the local string variables
- The bot extracts a substring from the local variable and assigns that substring to another variable



- ___ 4. Confirm that the transaction data variables populate correctly, and that the bot does not show any errors.

If you see errors, go to the line with the identified error in the Workbench, and review its corresponding steps.

Name	Value
vAcctNum	68291
vAmount	19500
vDate	01-26-2018
vRefNum	0204
vType	TRANSFER

Select All Add Remove

- ___ 5. Close the Variable(s) Watch Table and in the Workbench toolbar, click **Disable Debugging**.
- ___ 6. Close the transactions.xlsx spreadsheet.

Section 6. Defining loop actions: Writing data to the transactions.xlsx Excel spreadsheet

In this section, you use the `Set Cell Excel` command to write transaction data to the spreadsheet.

- ___ 1. After the final `String Operation` command in the Workbench, add the following comment:
Write the transaction data to the transactions.xlsx Excel spreadsheet
- ___ 2. In the Workbench, add a `Set Cell` command to write transaction data to the transactions.xlsx spreadsheet.
 - ___ a. In the **Commands** list, go to the Excel section.
If needed, double-click **Excel** to expand the section.
 - ___ b. Add the `Set Cell Excel` command to the Workbench.
 - ___ c. In the Excel window, confirm that **Set Cell** is the selected Excel command.
 - ___ d. In the Set Cell section, leave the **Session Name** as Default.
 - ___ e. Select **Specific Cell**, and in the **Specific Cell** field, enter: `A$vSpreadsheetRow$`

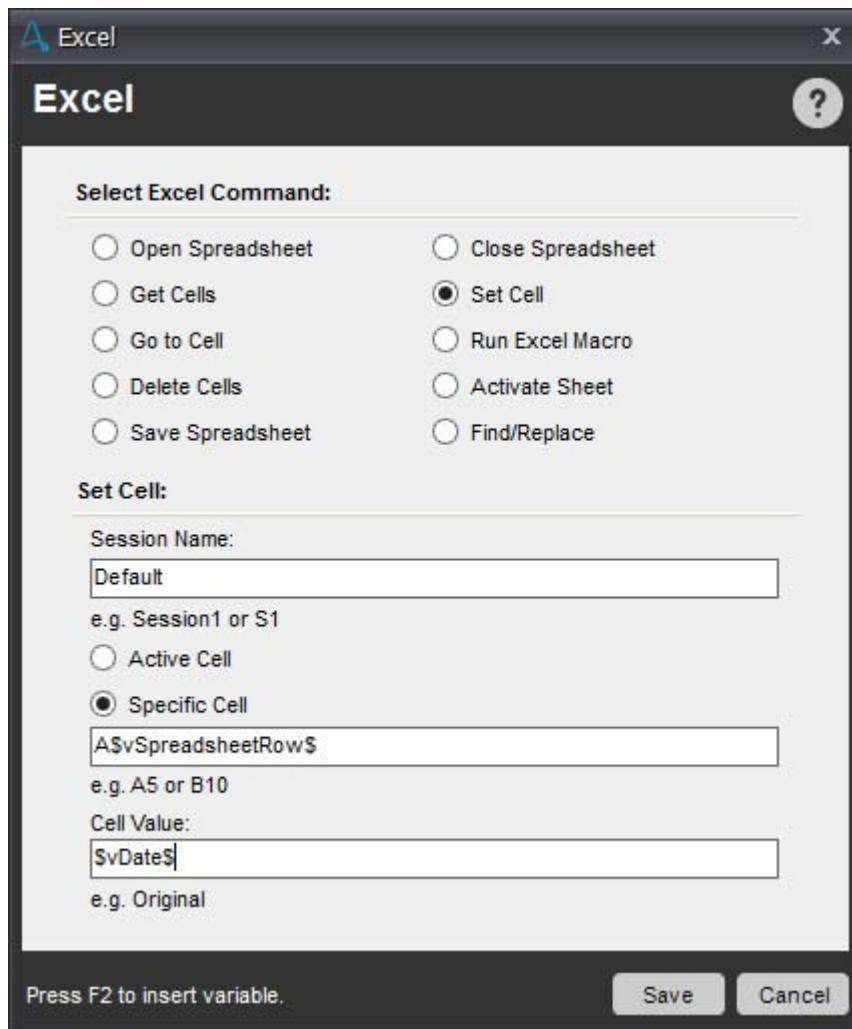


Reminder

The `vSpreadsheetRow` variable is initialized to 2. Thus, in this step, the specific cell is cell A2.

- ___ f. In the **Cell Value** field, enter: `$vDate$`

- ___ g. Click Save.



- ___ 3. Repeat Step 2 to add the following Set Cell commands to the Workbench:

Specific Cell	Cell Value
B\$vSpreadsheetRow\$	\$vAcctNum\$
C\$vSpreadsheetRow\$	\$vType\$
D\$vSpreadsheetRow\$	\$vAmount\$
E\$vSpreadsheetRow\$	\$vRefNum\$

- ___ 4. Add the following comment after the last Set Cell command:

Increment the value of vSpreadsheetRow by 1 to write data to the next spreadsheet row

- ___ 5. Add a Variable Operation command to increment the number of the spreadsheet row by 1.

This command makes sure that each line of transaction data is written to a new row in the spreadsheet.

- ___ a. From the **Commands** list, add a Variable Operation command to the Workbench. Make sure that it is after the comment that you added in Step 4.

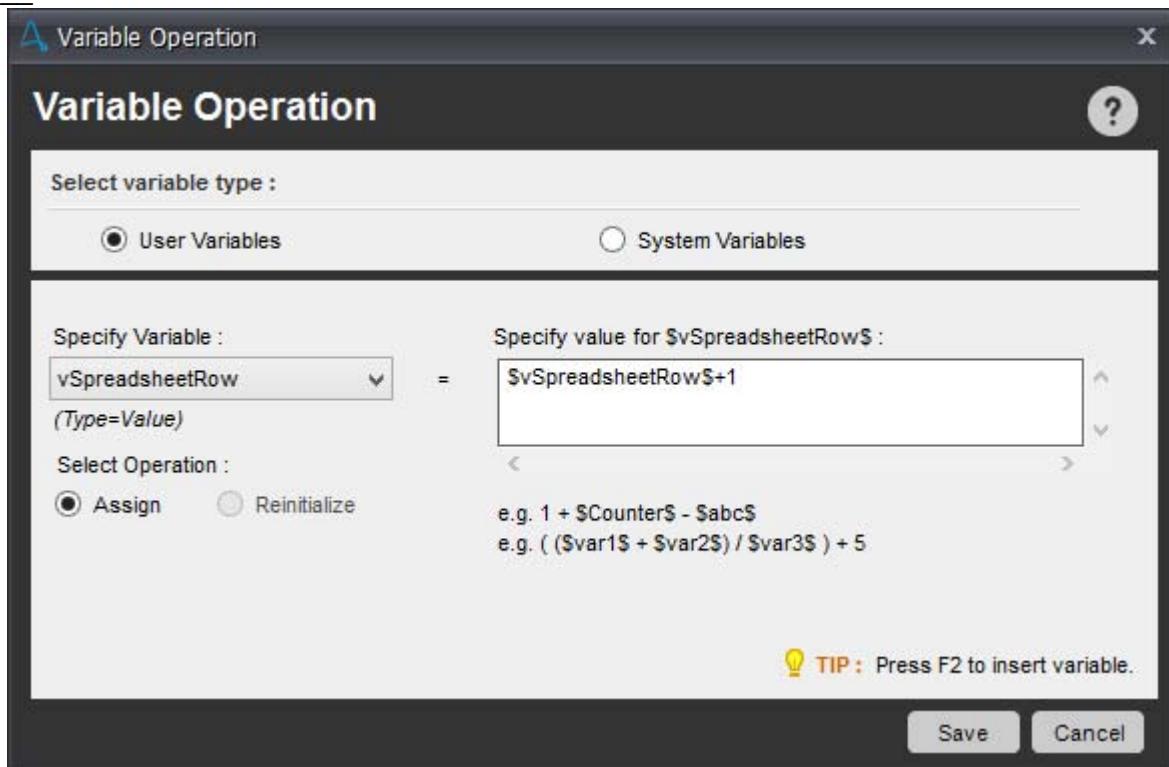
___ b. In the Variable Operation window, enter the following settings:

- **Specify Variable** list: Select **vSpreadsheetRow**.

- **Specify value** field: `$vSpreadsheetRow$+1`

You can use F2 (or Fn+F2) to enter the variable in the **Specify value** field.

___ c. Click **Save**.



___ 6. Save your work.

Your work with the loop is complete.

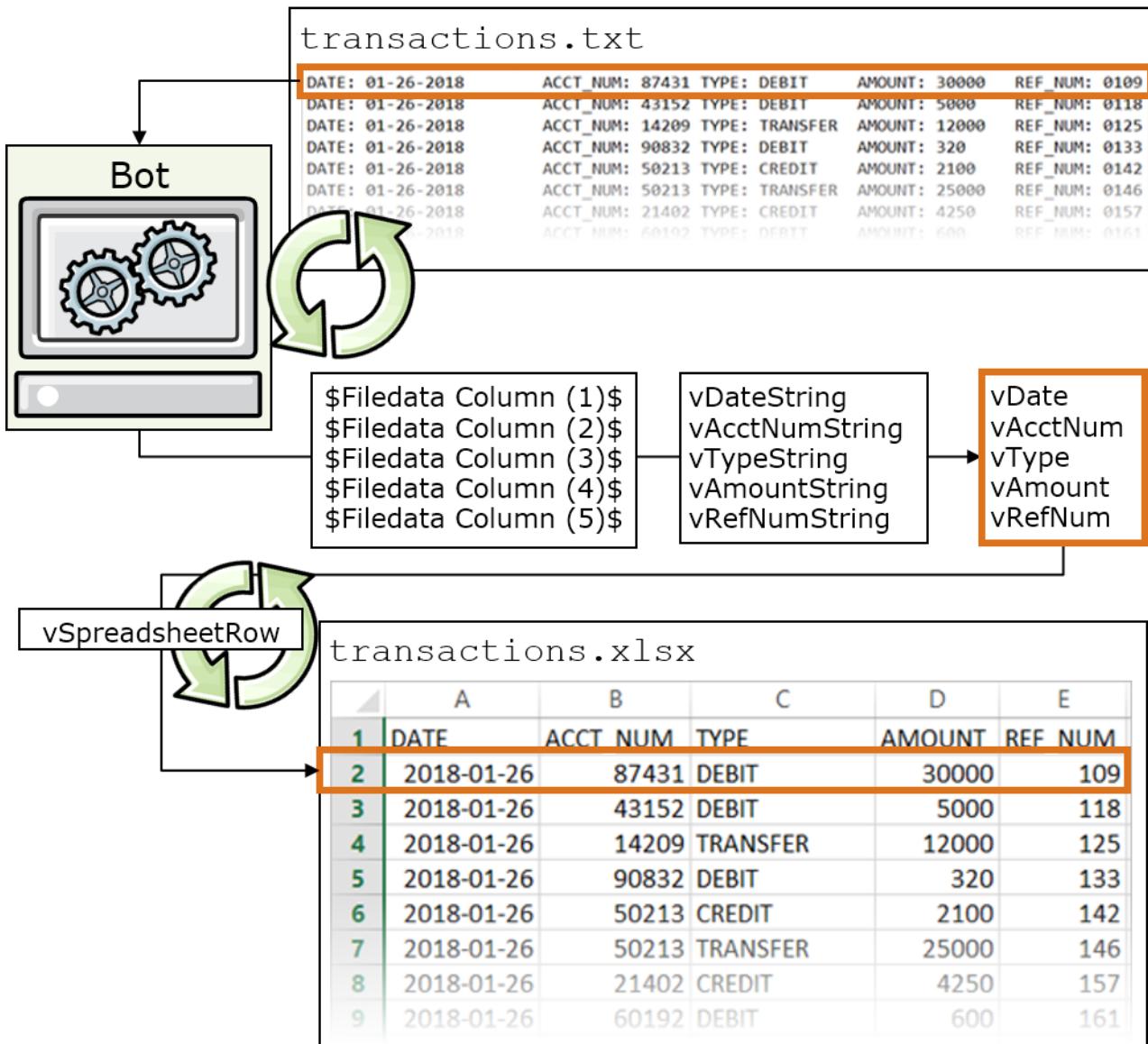
Section 7. Finishing and running the bot

In this section, you add the final `Save Spreadsheet` and `Close Spreadsheet` commands to the bot. You also run the bot to confirm that it works. Finally, you review the completed spreadsheet.

- 1. In the Workbench, click the `End Loop` command to select it.
Both of the last two commands are outside the loop.
- 2. Add the `Save Spreadsheet` and `Close Spreadsheet` commands to the Workbench.
 - a. In the Commands list, go to the **Excel** section.
 - b. Double-click the `Save Spreadsheet` command and click **Save**. This command should be after the `End Loop` command.
 - c. Double-click the `Close Spreadsheet` command and click **Save**. This command should be the final command.
- 3. Save your work.
- 4. In the Workbench toolbar, click **Run** to run the completed bot.

Bot actions:

- The bot enters a loop for iterating through the `transactions.txt` file.
- As the bot goes through each line in the text file, it puts each individual tab-delimited transaction data into the `$Filedata.Column()` variable
- The `$Filedata.Column()` data is transferred to the local string variables
- The bot extracts a substring from the local variable and assigns that substring to another local variable
- The bot opens the `transactions.xlsx` spreadsheet, writes the information from each variable into a specific column, and loops through each row of the data by using the `vSpreadsheetRow` variable as a counter.
- Finally, the bot saves and closes the spreadsheet.



Troubleshooting

If you see errors, go to the identified line in the Workbench, and review its corresponding exercise steps.

- ___ 5. Open and review the transactions.xlsx spreadsheet and the transactions.txt file.
- ___ a. Go to C:\labfiles\transactions, and double-click transactions.xlsx to open it.

	A	B	C	D	E	F
1	DATE	ACCT_NUM	TYPE	AMOUNT	REF_NUM	
2	1/26/2018	87431	DEBIT	30000	109	
3	1/26/2018	43152	DEBIT	5000	118	
4	1/26/2018	14209	TRANSFER	12000	125	
5	1/26/2018	90832	DEBIT	320	133	
6	1/26/2018	50213	CREDIT	2100	142	
7	1/26/2018	50213	TRANSFER	25000	146	
8	1/26/2018	21402	CREDIT	4250	157	
9	1/26/2018	60192	DEBIT	600	161	
10	1/26/2018	29218	DEBIT	3000	166	
11	1/26/2018	78213	DEBIT	17085	179	
12	1/26/2018	90832	CREDIT	875	184	
13	1/26/2018	90832	TRANSFER	1400	187	
14	1/26/2018	21402	DEBIT	6350	193	
15	1/26/2018	30459	TRANSFER	565	199	
16	1/26/2018	68291	TRANSFER	19500	204	

- ___ b. Double-click the transactions.txt file to open it.

```

DATE: 01-26-2018 ACCT_NUM: 87431 TYPE: DEBIT AMOUNT: 30000 REF_NUM: 0109
DATE: 01-26-2018 ACCT_NUM: 43152 TYPE: DEBIT AMOUNT: 5000 REF_NUM: 0118
DATE: 01-26-2018 ACCT_NUM: 14209 TYPE: TRANSFER AMOUNT: 12000 REF_NUM: 0125
DATE: 01-26-2018 ACCT_NUM: 90832 TYPE: DEBIT AMOUNT: 320 REF_NUM: 0133
DATE: 01-26-2018 ACCT_NUM: 50213 TYPE: CREDIT AMOUNT: 2100 REF_NUM: 0142
DATE: 01-26-2018 ACCT_NUM: 50213 TYPE: TRANSFER AMOUNT: 25000 REF_NUM: 0146
DATE: 01-26-2018 ACCT_NUM: 21402 TYPE: CREDIT AMOUNT: 4250 REF_NUM: 0157
DATE: 01-26-2018 ACCT_NUM: 60192 TYPE: DEBIT AMOUNT: 600 REF_NUM: 0161
DATE: 01-26-2018 ACCT_NUM: 29218 TYPE: DEBIT AMOUNT: 3000 REF_NUM: 0166
DATE: 01-26-2018 ACCT_NUM: 78213 TYPE: DEBIT AMOUNT: 17085 REF_NUM: 0179
DATE: 01-26-2018 ACCT_NUM: 90832 TYPE: CREDIT AMOUNT: 875 REF_NUM: 0184
DATE: 01-26-2018 ACCT_NUM: 90832 TYPE: TRANSFER AMOUNT: 1400 REF_NUM: 0187
DATE: 01-26-2018 ACCT_NUM: 21402 TYPE: DEBIT AMOUNT: 6350 REF_NUM: 0193
DATE: 01-26-2018 ACCT_NUM: 30459 TYPE: TRANSFER AMOUNT: 565 REF_NUM: 0199
DATE: 01-26-2018 ACCT_NUM: 68291 TYPE: TRANSFER AMOUNT: 19500 REF_NUM: 0204

```

- c. Review and compare the data in the text file and the spreadsheet.
-

**Note**

In the `transactions.txt` file, notice that the date format uses hyphens to separate month, day, and year. The `transactions.xlsx` file does not include the hyphens, and substitutes forward slashes (/) as the separator because of how Excel handles date formats.

As you work on bots that send data from one application to another, be aware of how different applications format data.

- d. When you are finished looking over the data, close `transactions.xlsx` and, if you opened it then close, `transactions.txt`.
6. Return to the Workbench and close it.

Section 8. *Optional:* Viewing the solution file

If you were unable to complete and run the bot successfully, you can open the solution file for this exercise to see how the bot should be coded.

- __ 1. Go to C:\labfiles\transactions\solution, right-click the Transactions solution.atmx file, and click **Edit**.
The file opens in the Workbench.
 - __ 2. To run the bot, click **Run** on the Workbench toolbar.
-



Information

If you want to load the solution file from the Enterprise Client, move the Transactions solution.atmx file to the following directory: C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks

The Enterprise Client **My Tasks** view automatically accesses files in that folder.

From the **My Tasks** list, you can either run the bot by double-clicking it or you can view it by right-clicking the task and clicking **Edit**.

End of exercise

Exercise review and wrap-up

In this exercise, you created a bot and defined local variables for the bot. You saw how to use loops to read a text file, and how to use variables to transfer data from one file to another. You also saw how to work with Microsoft Excel spreadsheets.

Exercise 4. Automating data entry to a business application and a database

Estimated time

02:00

Overview

In this exercise, you build a bot that copies customer data from a spreadsheet and enters the data into both a Windows-based business application and a separate database.

Objectives

After completing this exercise, you should be able to:

- Use the Smart Recorder and the Object Cloning command to capture interface components and actions
- Work with variables to pass data between different applications
- Build loops to automate repetitive tasks across several different applications
- Use the Object Cloning command and the String Operation command to capture and extract data from a message window

Introduction

In this exercise, you create a bot that fully automates the account opening process by performing each of these steps. You use both the Enterprise Client Smart Recorder and Workbench to create bot actions and copy data. You also use DB2 and a Visual Basic application that emulates a CRM data entry form.

Scenario: The SIB customer relationship management (CRM) system, which is a Windows-based application, is not connected to the MyBank CRM system. To synchronize customer data between SIB and MyBank, a data entry clerk must manually update the data in two separate systems.

As a bot developer, you must develop a bot that automates these tasks:

- Open and iterate through a spreadsheet to copy new customer data to the SIB CRM system
- Record the cross-reference number that is generated by the CRM system back in the spreadsheet for each new account
- Update the SIB database with the data for each customer

The exercise includes these sections:

- [Section 1, "Defining custom variables for customer data"](#)
- [Section 2, "Opening the spreadsheet and connecting to the database"](#)
- [Section 3, "Opening the SIB CRM application"](#)
- [Section 4, "Using a loop to perform repetitive tasks"](#)
- [Section 5, "Finishing and running the bot"](#)
- [Section 6, "Optional: Viewing the solution file"](#)

Requirements

This exercise requires the computer lab environment that was developed for this lab.

Section 1. Defining custom variables for customer data

In this part of the exercise, you use the IBM Robotic Process Automation with Automation Anywhere Enterprise Client to create a new task and create custom variables to hold customer account data.

1.1. Creating a task

In this section, you log in to the Control Room as the RPA administrator `rpaAdmin`, and log in to the Enterprise Client as the developer user `devUser1`. You then create a task in the Workbench.

- ___ 1. Go to the Enterprise Client.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

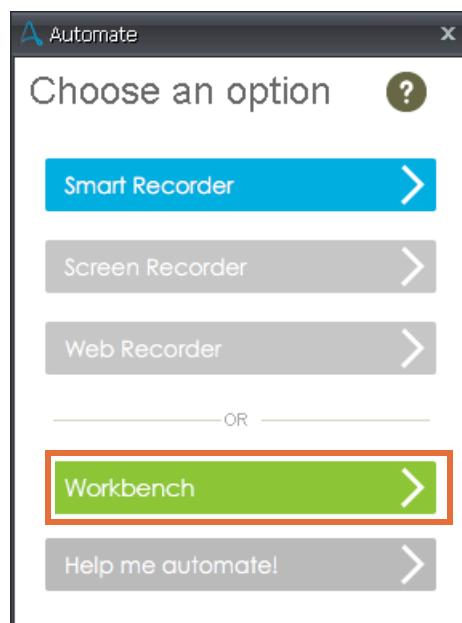
test

- ___ 2. Create a task with the Workbench.

- ___ a. On the Enterprise Client toolbar, click **New**.



- ___ b. In the Automate window, click **Workbench**.

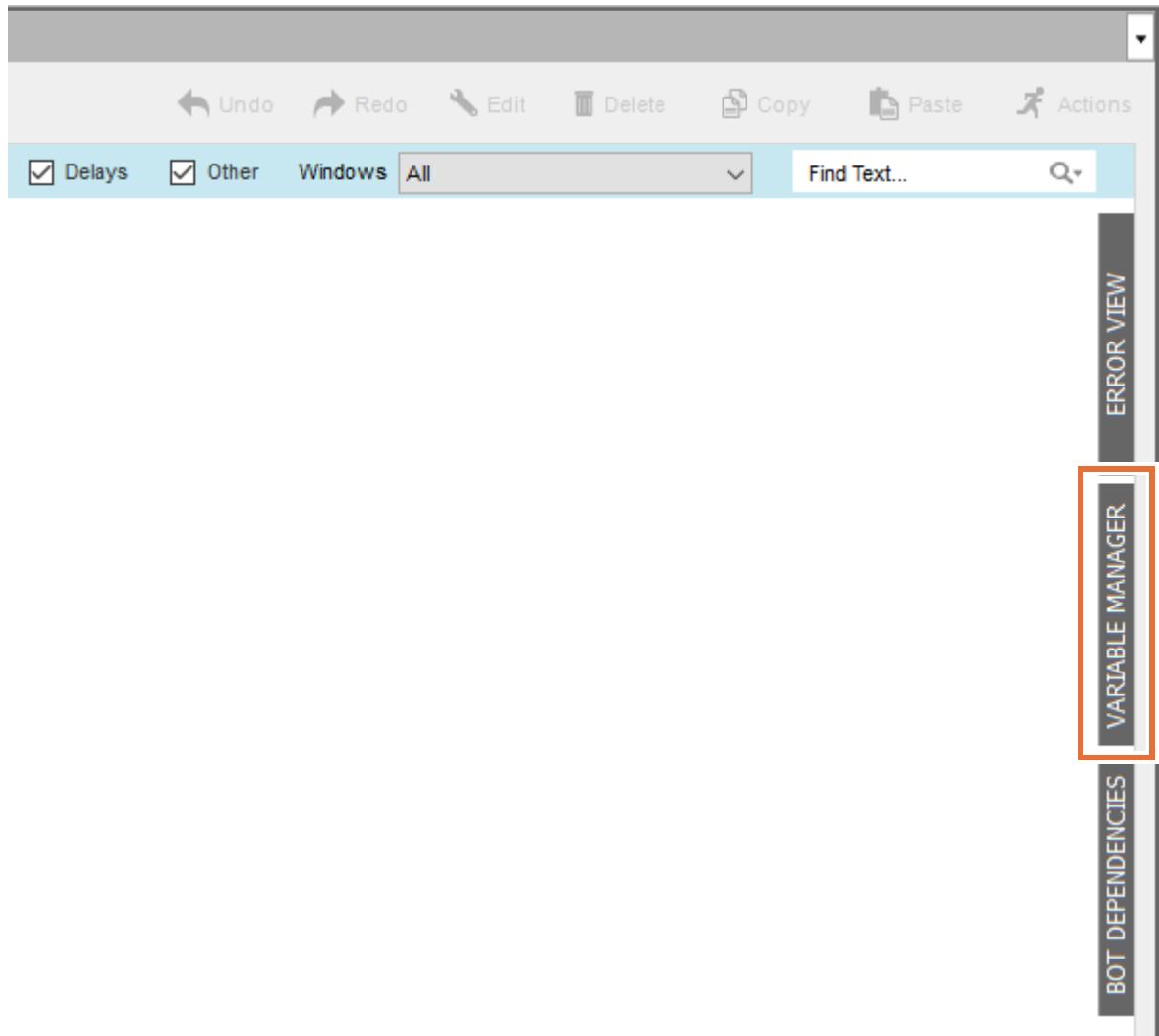


A new task opens in the Workbench.

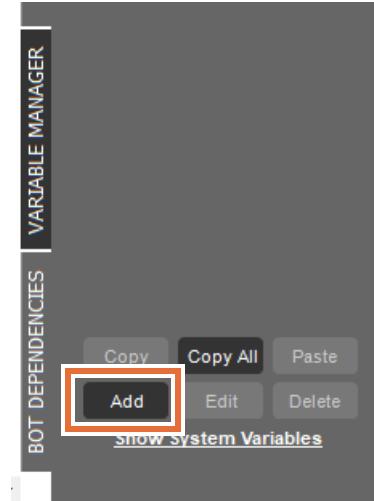
1.2. Creating custom variables

In this section, you use the Variable Manager to create local (user) variables that the bot can use to pass customer account data between different applications.

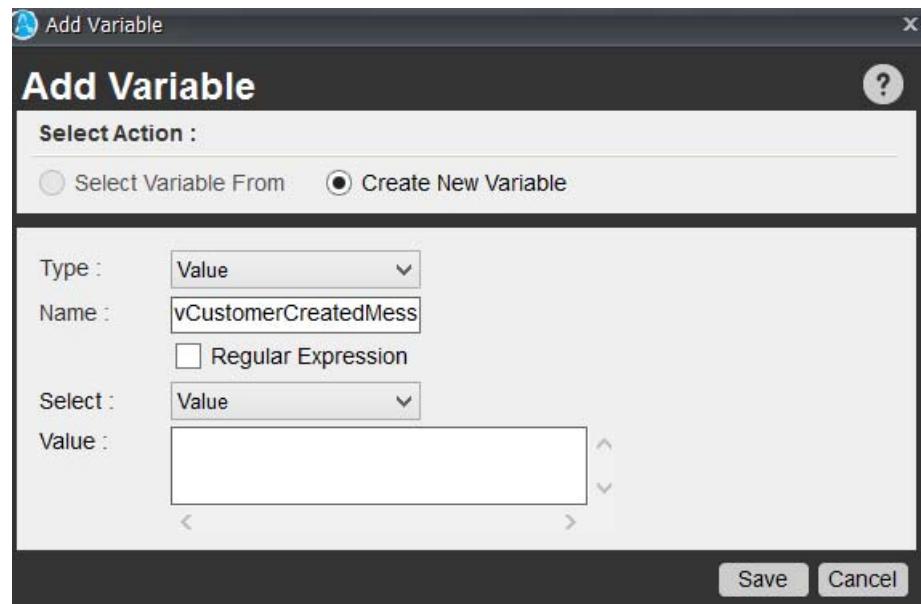
- 1. In the Workbench, click the **VARIABLE MANAGER** vertical tab to expand it.



- __ 2. In the Variable Manager pane, click **Add**.

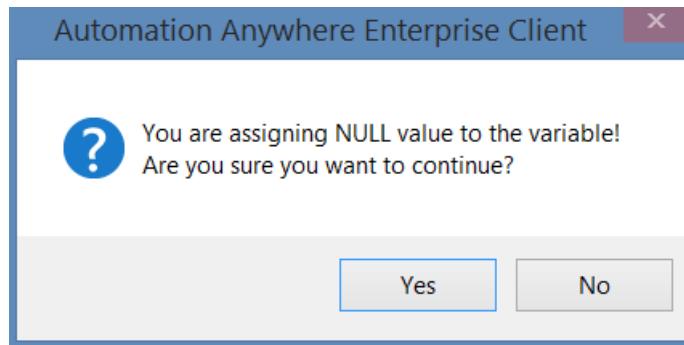


- __ 3. Define the vCustomerCreatedMessage variable, which holds the text that is extracted from a message window as a string.
- Make sure that **Create New Variable** is selected.
 - In the **Type** list, make sure that **Value** is selected.
 - In the **Name** field, enter: vCustomerCreatedMessage
 - Leave the **Value** field empty.



- __ 4. Click **Save**.

- ___ 5. In the Automation Anywhere Enterprise Client window, click **Yes** to accept a null value for the variable.

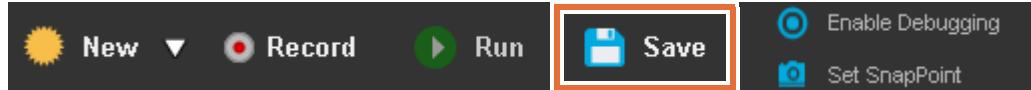


- ___ 6. Repeat these steps to create the following variables:

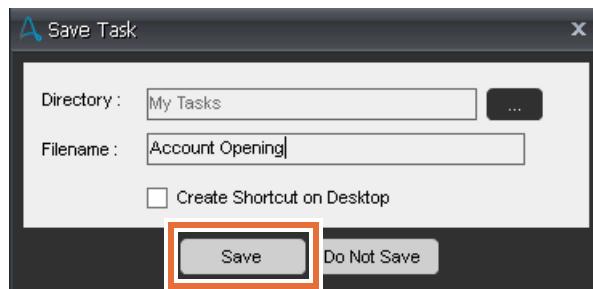
Variable name	Type	Value	Description
vCustomerRef	Value		This variable holds the customer reference number, which is extracted as a substring from the value in vCustomerCreatedMessage.
vRowNumber	Value	1	This variable is used to iterate through the data in the new_customers.xlsx Excel spreadsheet.

- ___ 7. Save your work, and name the task: Account Opening

- ___ a. On the toolbar, click **Save**.



- ___ b. In the Save Task window, keep the default **Directory** setting of **My Tasks**.
 ___ c. In the **Filename** field, enter: Account Opening
 ___ d. Click **Save**.



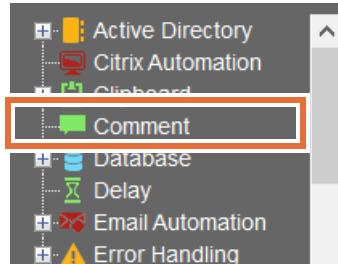
The task file is now saved, and the local variables are now ready for the bot to use.

Section 2. Opening the spreadsheet and connecting to the database

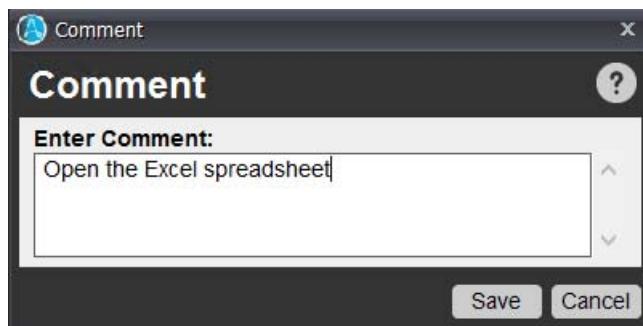
In this section, you define the first actions that the bot performs: opening the `new_customers.xlsx` spreadsheet and connecting to the SIB DB2 database.

2.1. Opening the spreadsheet

- ___ 1. Enter a comment that describes the first action of opening the Excel spreadsheet.
 - ___ a. In the Workbench Commands list, double-click the **Comment** task.

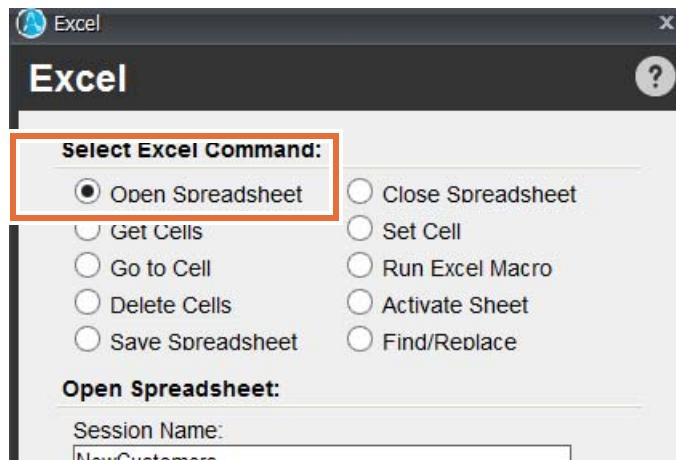


- ___ b. In the Comment window, enter: Open the Excel spreadsheet

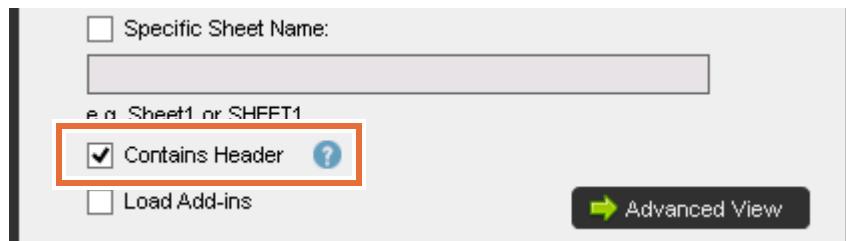


- ___ c. Click **Save**.
- ___ 2. Use the Open Spreadsheet Excel command to open the `newcustomers.xlsx` spreadsheet.
 - ___ a. In the Workbench, make sure that the **Comment** line is highlighted. If it is not, click **Comment** to select it.

- ___ b. In the Commands list, expand **Excel**, and double-click **Open Spreadsheet** to add it to the Actions List.

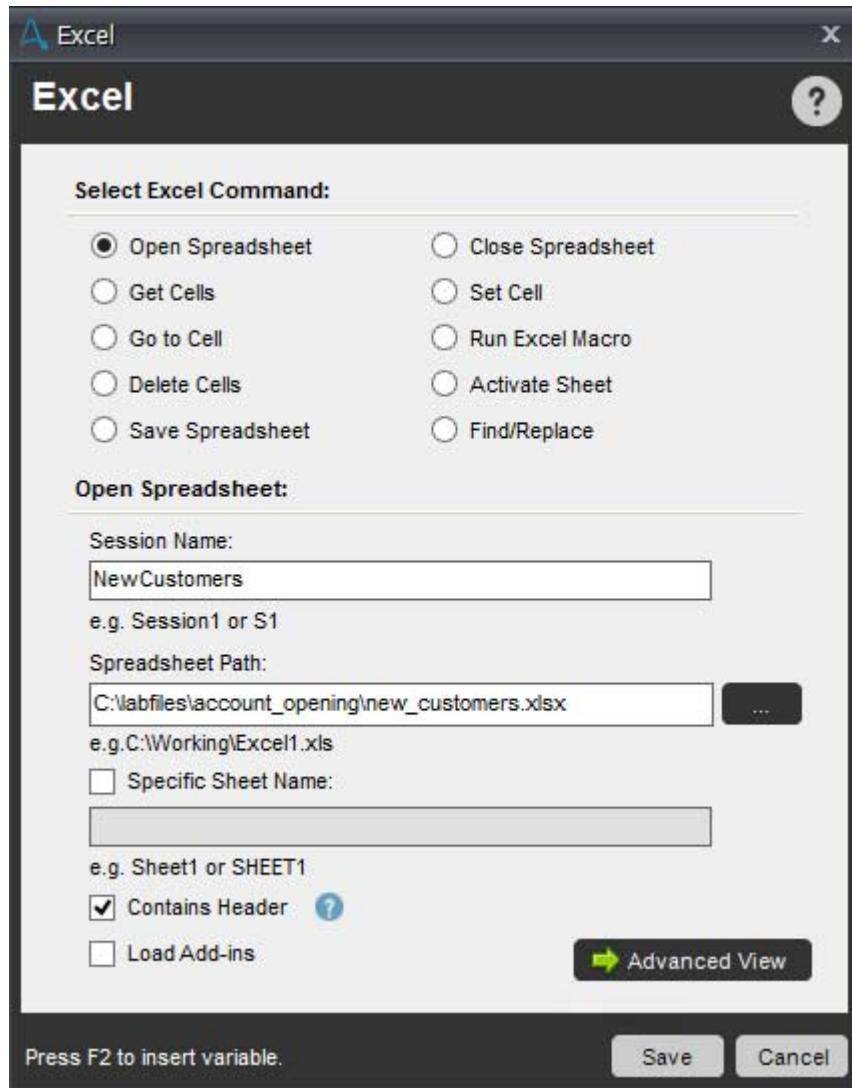


- ___ c. In the Excel window, keep the **Open Spreadsheet** option selected.
___ d. In the **Session Name** field, enter: NewCustomers
___ e. In the **Spreadsheet Path** field, enter:
C:\labfiles\account_opening\new_customers.xlsx
___ f. In the Excel command window, select **Contains Header**.



When you select this option, the bot skips the first row in the spreadsheet.

- ___ g. Click **Save**.



- ___ 3. On the Enterprise Client toolbar, click **Save** to save your work.



Note

It is a good practice to save your work often.

- ___ 4. Maximize the Excel window.

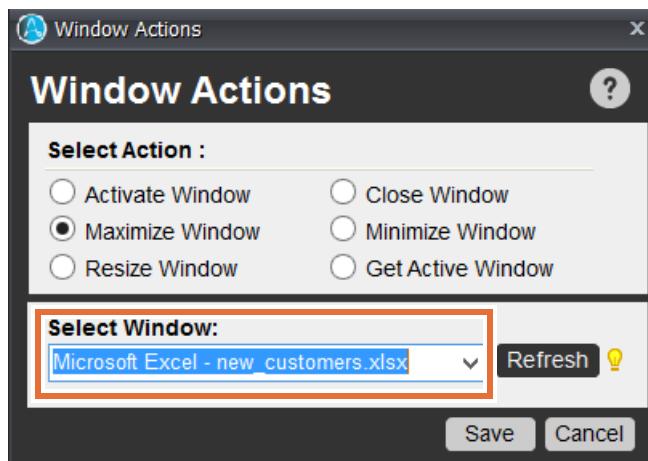
- ___ a. In Windows Explorer, go to C:\labfiles\account_opening, and double-click the new_customers.xlsx spreadsheet to open it.
- ___ b. If needed, minimize the new_customers.xlsx window.
- ___ c. Go back to Workbench.
- ___ d. In the **Commands** list, expand **Window Actions**, and drag **Maximize Window** to the empty space of the Workbench after the Open Spreadsheet command.

**Note**

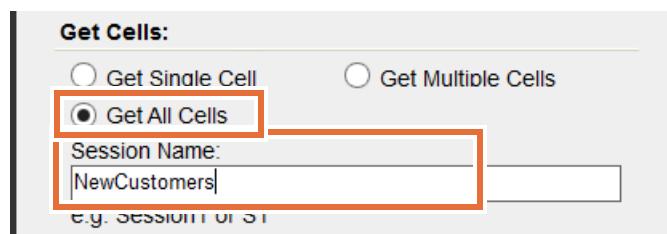
When you add commands to the Workbench by dragging them from the Commands list, be careful where you drag commands. You can add a command to an incorrect location, which would cause the task steps to be out of sequence.

If you add a command to the wrong location, you can drag it to its correct location in the sequence of task steps.

- ___ e. In the Window Actions window, from the **Select Window** list, select **new_customers.xlsx**, and click **Save**.



- ___ 5. Use the Get All Cells Excel command on the **new_customers.xlsx** spreadsheet.
 - ___ a. In the Commands list, go to the Excel section.
 - ___ b. Add the **Get Cells** command. Make sure that it is added after the **Maximize Window** command.
 - ___ c. In the Excel window, in the Get Cells section:
 - Select **Get All Cells**.
 - In the **Session Name** field, make sure that the value is: **NewCustomers**



- ___ d. Click **Save**.
- ___ 6. Save your work.

2.2. Connecting to the database

- ___ 1. In the **Commands** list, expand **Database**, and double-click the **Connect** command.

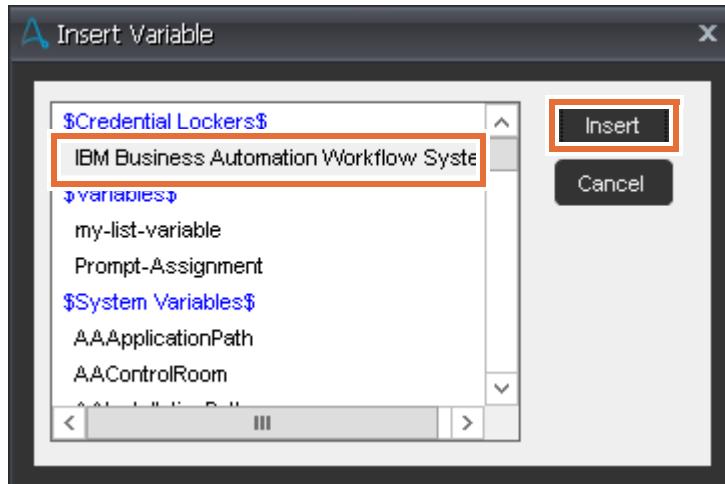
__ 2. Define the database connection.

- __ a. In the Connect section, leave the **Session Name** value at: Default
- __ b. Click the **Connection String** field, and press F2 (or Fn+F2) to insert a variable.

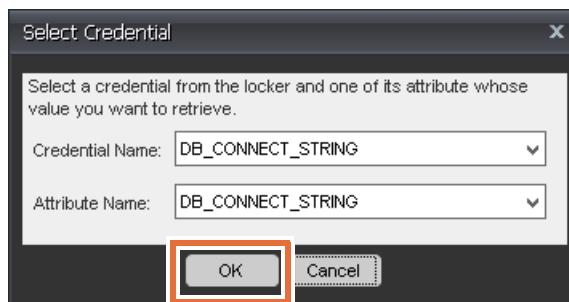
Instead of manually defining a connection to the DB2 database, you use a systemwide credential variable that is defined in the Control Room Credential Manager:

DB2_CONNECT_STRING. The connection string includes information that is needed to connect to the database, such as the data provider, user ID, and user password.

- __ c. In the Insert Variable window, in the \$Credential Lockers\$ section, select **IBM Business Automation Workflow Systems**, and click **Insert**.



- __ d. In the Select Credential window, select DB_CONNECT_STRING for both the **Credential Name** and **Attribute Name**, and click **OK**.

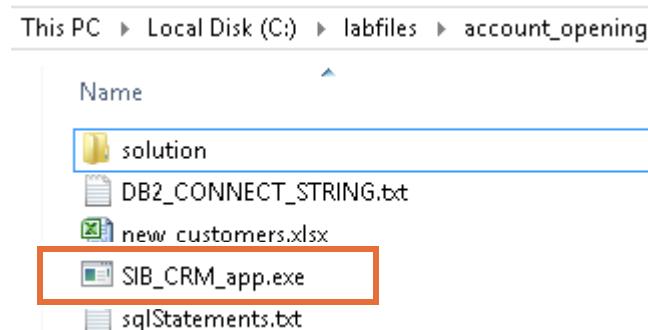


- __ e. Click **Save** to save the database connection command.

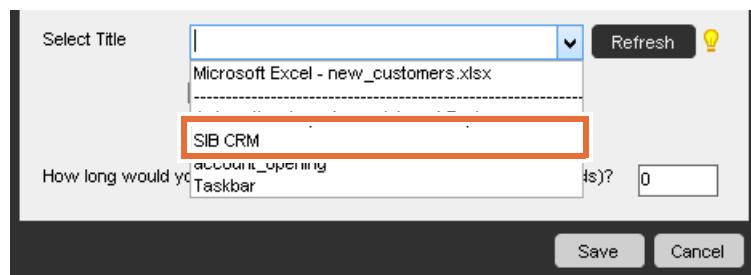
Section 3. Opening the SIB CRM application

In this section, you add an `If` command to make sure that the SIB CRM application is running. You then set the window size and location.

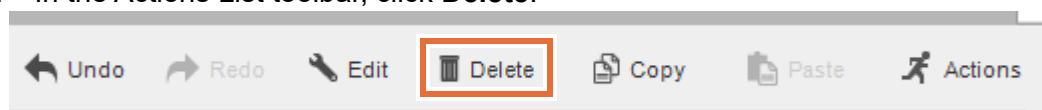
- ___ 1. From the **Commands** list, add a `Comment` command to the Workbench to explain the purpose of the `If` command. Make sure that it is after the database `Connect` command.
Ensure that the SIB CRM application is running
- ___ 2. Start the SIB CRM application by going to `C:\labfiles\account_opening` in Windows Explorer and double-clicking `SIB_CRM_app.exe`.



- ___ 3. In the Workbench window, add an `If` command that checks to see whether the SIB CRM application window is open.
 - ___ a. In Workbench, from the **Commands** list, expand **If/Else** and double-click **Window Does Not Exist**.
 - ___ b. From the **Select Title** list, select **SIB CRM**.



- ___ c. Click **Save**.
- ___ 4. Remove the `If` command comment.
 - ___ a. Select the `If` command comment with the text `Please enter the conditional commands here.`
 - ___ b. In the Actions List toolbar, click **Delete**.



- ___ c. Click **Yes** to confirm the deletion.

- ___ 5. Complete the If command so that if the SIB CRM application is not running, it starts.
 - ___ a. In the **Commands** list, add **Open Program/File** to the line after the **If Window Does Not Exist** command.
 - ___ b. In the Open Program/File window, click **Browse** next to the **Step 1: Program/File Path** field.



- ___ c. In the Open window, go to C:\\labfiles\\account_opening, select **SIB_CRM_app.exe**, and click **Open**.
- ___ d. Back in the Open Program/File window, click **Save**.
- ___ 6. Add the following comment after the **End If** command:
Resize and move the SIB CRM window
- ___ 7. Resize and relocate the SIB CRM application window to a specific set of display coordinates.



Note

In this exercise, you size the window to a specific set of dimensions and set it to a specific location on your monitor. You set the size and location that you can see the automated data entry in the CRM application window.

In general, it is a good practice to maximize application windows to ensure consistent locations for user interface items.

- ___ a. In the **Commands** list, go to **Windows Actions**, and add **Resize Window** to the Workbench. Make sure to add this command after the comment that you entered in Step 5.
- ___ b. In the Window Actions window, from the **Select Window** list, select **SIB CRM**.
- ___ c. In the Select Window section, enter the following values:
 - **Top:** 320
 - **Left:** 400
 - **Height:** 420
 - **Width:** 380
- ___ d. Click **Save**.
- ___ 8. Save your work.

Your bot code is ready to test.

```

1  Comment: Open the Excel spreadsheet
2  Excel: Open Spreadsheet "C:\labfiles\account_opening\new_customers.xlsx".ActiveSheet:
3  Maximize Window: "Microsoft Excel - new_customers.xlsx"
4  Excel: Get All Cells Session: NewCustomers
5  Connect to "$DB_CONNECT_STRING(DB2_CONNECT_STRING)$" Session:'Default'
6  Comment: Ensure that the SIB CRM application is running
7  IF Window Does Not Exist ("SIB CRM") Then
8    Open: "C:\labfiles\account_opening\SIB_CRM_app.exe"
9  End If
10 Comment: Resize and move the SIB CRM window
11 Resize Window: "SIB CRM" with dimensions Top: 320, Left: 400, Height: 420, Width: 380

```

- ___ 9. Close the SIB CRM window and the new_customers.xlsx spreadsheet.

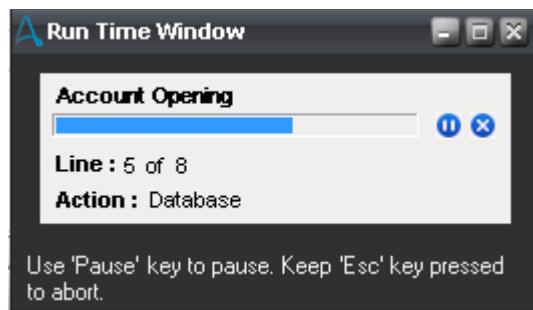
3.1. Testing the bot

Run the bot to see whether it has any errors.

- ___ 1. In the Enterprise Client toolbar, click **Run**.



The Run Time window opens, and shows the progress of the bot.



Bot actions:

- Open and maximize the new_customers.xlsx spreadsheet
 - Open the SIB CRM application window and place it at the coordinates that you defined
-

- ___ 2. If you see errors, compare the identified Workbench line from the error message to the corresponding step in the lab instructions.
- ___ 3. Do not close the SIB CRM window. You work with this window in the next section of the lab.
- ___ 4. Minimize the SIB CRM window and the new_customers.xlsx spreadsheet.

Section 4. Using a loop to perform repetitive tasks

In this section, you create a loop to handle repetitive tasks, such as:

- Reading spreadsheet data
- Entering customer information into a business application
- Writing a customer reference number into a spreadsheet
- Inserting customer information in a database

4.1. Setting up the loop

In this section, you add a loop command to iterate through the `new_customers.xlsx` Excel spreadsheet.

- ___ 1. Go back to the Workbench.
- ___ 2. In the **Commands** list, expand **Loop**, and add **Each Row In An Excel Dataset** to the Workbench so that it is after the `Resize Window` command.
- ___ 3. In the Loop Window, configure the loop.
 - ___ a. In the Select Loop Command section, keep **Start Loop** selected.
 - ___ b. In the Loop For section, keep **Each row in an Excel dataset** selected.
 - ___ c. In the **Session Name** field, replace the default name with: `NewCustomers`
- ___ 4. Click **Save**.



- ___ 5. Edit the default comment to include information about the purpose of the loop.
 - ___ a. In the Workbench, click **Comment: Please enter your commands to loop** to select the statement.

- ___ b. In the Actions List toolbar, click **Edit**.



- ___ c. In the Comment window, delete the default text, and enter:

For each row, create a customer

- ___ d. Click **Save** to save the updated comment.

- ___ 6. Add a Variable Operation command to update the row number.

- ___ a. In the **Commands** list, double-click the **Variable Operation** command.
- ___ b. In the Variable Operation window, keep the **Select variable type** option at **User Variables**.
- ___ c. In the Select Variable section, click **Select a variable**, and select **vRowNumber**.
- ___ d. In the **Specify value for \$vRowNumber\$** field, enter:

`$vRowNumber$+1`

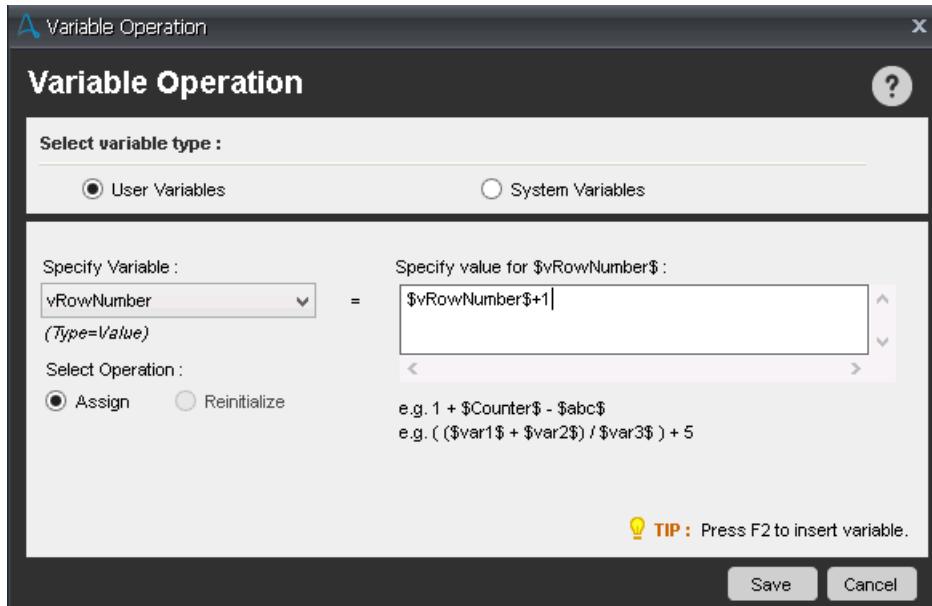


Note

You can insert the `$vRowNumber$` variable by pressing F2 (or Fn+F2, depending on your operating system and keyboard), and selecting it from the list.

You can then complete the variable operation by typing: +1

- ___ e. For Select Operation, keep **Assign** as the option.
- ___ f. Click **Save**.



- ___ 7. Save your work.

4.2. Using the Smart Recorder to automate data entry tasks

In this section, you use the Object Cloning command and the Smart Recorder to capture form fields from the SIB CRM window, and to automate data entry.

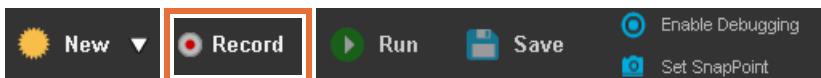
When you use the Smart Recorder to capture interface objects, it adds object cloning commands to the Workbench. You can then edit the object cloning commands in the Workbench.



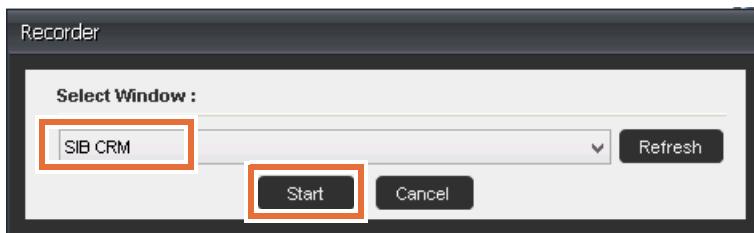
Important

The Smart Recorder reacts to movements of your mouse, so if you need to scroll through these exercise steps while recording your screen, you might capture extra mouse movements. Try to position these instructions on your screen so that you do not need to scroll.

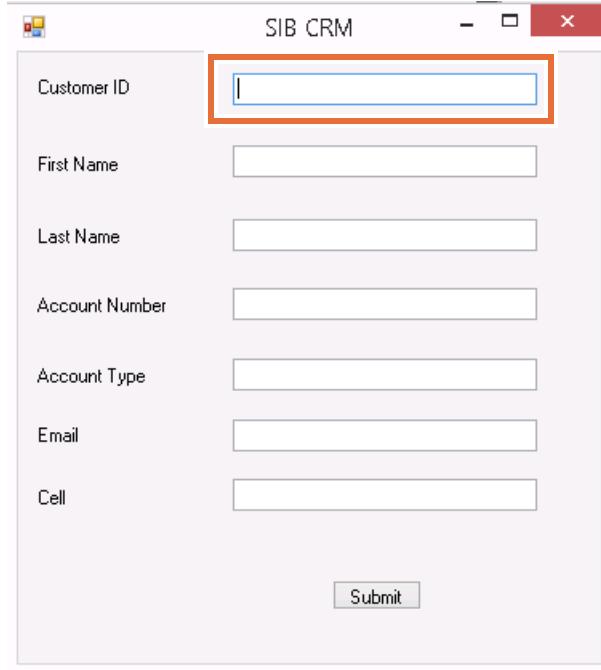
- ___ 1. In the Workbench, add the following comment after the Variable Operation command:
Enter the data
- ___ 2. Capture the SIB CRM application form fields by using the Smart Recorder.
 - ___ a. In the Workbench, make sure that the Comment: Enter the data line is selected.
 - ___ b. In the toolbar, click **Record**.



- ___ c. In the Recorder window, from the **Select Window** list, select **SIB CRM**, and click **Start**.



- ___ d. In the SIB CRM window, click the **Customer ID** field to select it.



- ___ e. In the Recording window, which runs in the lower-right part of your screen, click **Stop**.



The Object Cloning command for the “Customer ID” text box is now in the Workbench.

```

15   Comment: Enter the data
16   Object Cloning: Set Text of TextBox "Customer ID" in window 'SIB CRM'; Value:""
17   End Loop

```

- ___ 3. If the Object Cloning Image window is open, you can close it.

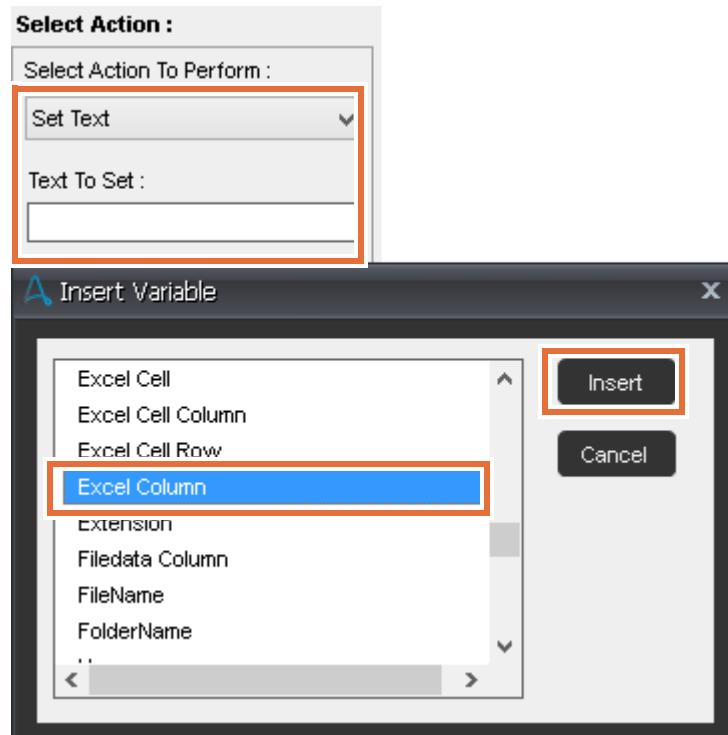




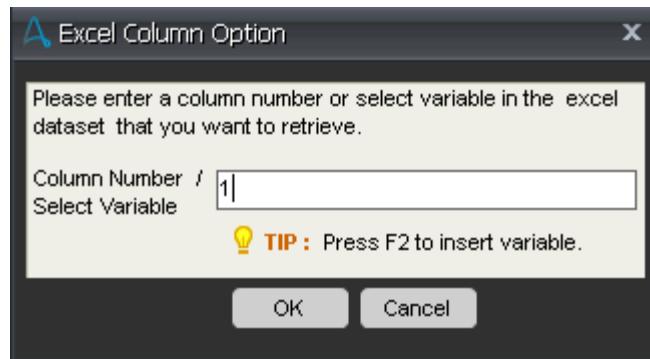
Troubleshooting

The Smart Recorder records each interaction with the user interface, such as clicks and keystrokes. When you accidentally record extra or incorrect Object Cloning interactions, you can delete them from the Actions List.

- ___ 4. Edit the Object Cloning command to set the text in the field to the value of the cell in its corresponding spreadsheet column.
 - ___ a. Make sure that the Object Cloning command is selected, and in the Actions List toolbar, click **Edit**.
 - ___ b. In the Object Cloning window, in the Select Action section, make sure that **Set Text** is selected.
 - ___ c. In the **Text To Set** field, press F2 (or Fn+F2) to insert a variable.
 - ___ d. In the **\$SystemVariables\$** part of the list, select **Excel Column**, and click **Insert**.



- ___ e. In the Excel Column Option window, set the **Column Number>Select Variable** field to: 1
- ___ f. Click **OK**.



The Object Cloning window now includes the action to set the text in the field to the value in the corresponding spreadsheet column.



- ___ g. In the Object Cloning window, click **Save**.
- ___ 5. Repeat [Step 2](#) through [Step 4](#) to add the following SIB CRM application fields to the bot, and to edit each Object Cloning command to define the Excel column values for each field.

You might see errors when you edit the Object Cloning command. See the next troubleshooting note.

SIB CRM application field	Object ID	Excel column value
First Name	tFname	2
Last Name	tLname	3

Account Number	tAccNum	4
Account Type	tAccType	5
Email	tEmail	6
Cell	tCell	7



Troubleshooting

After recording, you might see that the Actions List uses the wrong application form fields (such as **Last Name**, **Account Number**, **Account Type**, **Email**, and **Cell**) as **First Name**. However, each field is correctly captured in the Object Cloning command.

If you edit an `Object Cloning` command for one of the SIB CRM application fields, you can see that it is correctly identified in the Object Details section.

For example, the `Object Cloning` command in the Actions List might incorrectly identify the **First Name** field as the field that was captured. But when you open the Object Cloning window, the **Object ID** `tCell` field is correctly set to the field that was captured.

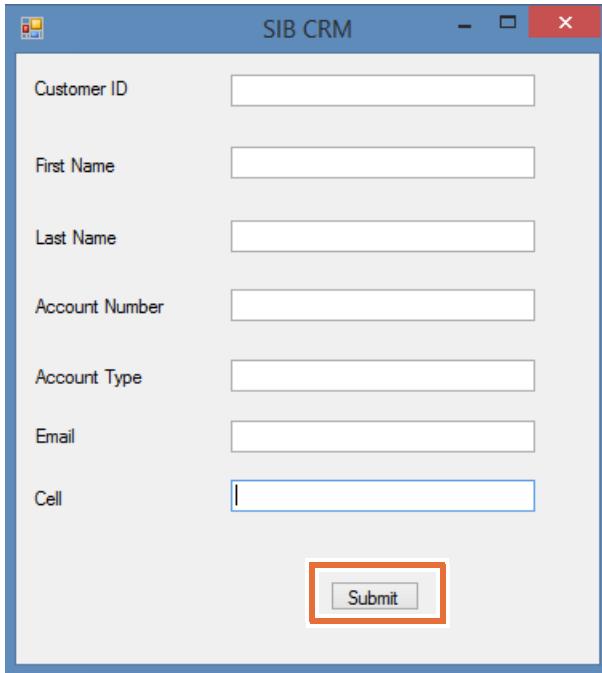


Important

For each field that you capture with the Smart Recorder, confirm that its **Object ID** property is correct before you assign the `$Excel Column()` variable to the **Text To Set** field.

- ___ 6. Save your work.
- ___ 7. Use the Smart Recorder to capture the **Submit** button-click action.
 - ___ a. In the Workbench, select the final `Object Cloning` command.
 - ___ b. Click **Record**, make sure that **SIB CRM** is selected, and click **Start**.

- ___ c. In the SIB CRM application window, click **Submit**.



- ___ d. In the Recording window, click **Stop**.

When you click **Submit**, you see a Data Submitted window that displays a message with a customer reference number.



You work with the Data Submitted window in the next section of this lab.

Your bot code should match this screen capture.

```

14   Comment: Enter the data
15   Object Cloning: Set Text of TextBox "Customer ID" in window 'SIB CRM'; Value: "$Excel Column(1$)"; Source: Window
16   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(2$)"; Source: Window
17   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(3$)"; Source: Window
18   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(4$)"; Source: Window
19   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(5$)"; Source: Window
20   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(6$)"; Source: Window
21   Object Cloning: Set Text of TextBox "First Name" in window 'SIB CRM'; Value: "$Excel Column(7$)"; Source: Window
22   Object Cloning: Click On PushButton "Submit" in window 'SIB CRM'; Click Type: Left Click; Source: Window; Play Type: Click

```

4.3. Using the Smart Recorder to submit the form and using a string operation to extract the reference number from a message window

In this section, you extract the reference number from the Data Submitted message window, and assign it to the vCustomerID variable. You also capture the action of closing the Data Submitted message window.

- ___ 1. Add the following comment to the Workbench after the final Object Cloning command:

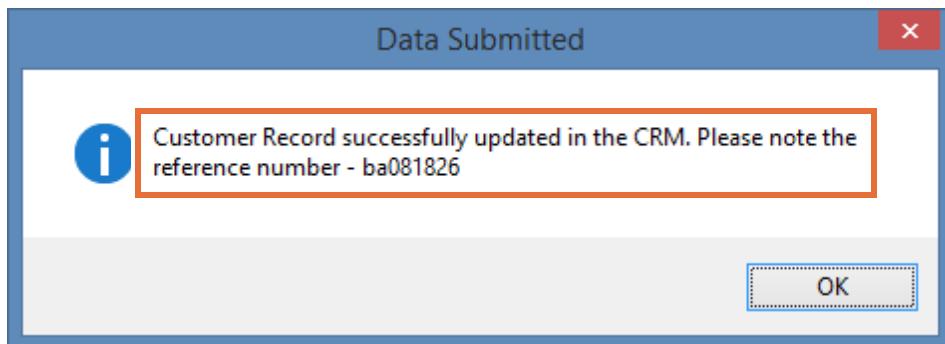
Extract the reference number from the message

- ___ 2. Use the Smart Recorder to capture the message from the Data Submitted window.

- ___ a. In the Workbench toolbar, click **Record**.
- ___ b. In the Recorder window, make sure that **Data Submitted** is selected, and click **Start**.



- ___ c. In the **Data Submitted** window, click the message text, then click **Stop** in the Recording window.



The Object Cloning command should read:

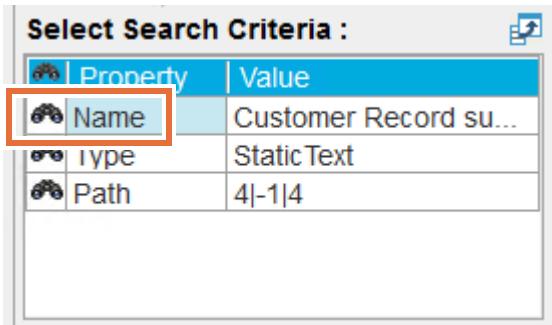
Click On StaticText "Customer Record successfully updated in the CRM."

25		Comment: Extract the reference number from the message
26		Object Cloning: Click On StaticText "Customer Record successfully updated in the CRM. Please note the reference"



Make sure that you click the message text. If you click another part of the window, you might capture a different object.

- ___ 3. Edit the Object Cloning command for the Data Submitted window to assign the message to the vCustomerCreatedMessage variable.
- Select the Object Cloning command for the Data Submitted window.
 - In the Actions List toolbar, click **Edit**.
 - Remove the **Name** property from the Select Search Criteria list by clicking the binoculars icon next to **Name**.

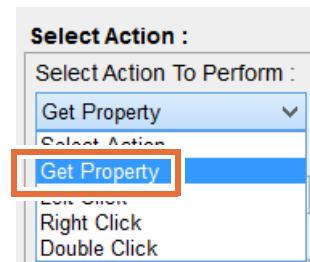
**Note**

If you do not disable the **Name** property in the search criteria, the bot searches for message text that has the same reference number that is in the captured message text.

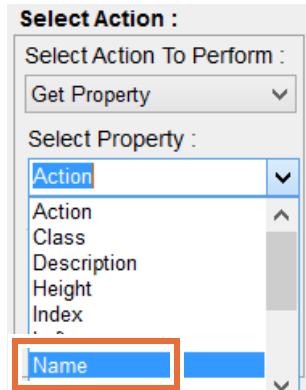
Because each reference number is unique, the bot fails if it cannot find the exact message text that is specified in an enabled **Name** property.

By not specifying a **Name** property in the search criteria, you can configure the bot to take the value of the **Name** property for each Data Submitted window. It can then assign the message text to the vCustomerCreatedMessage variable.

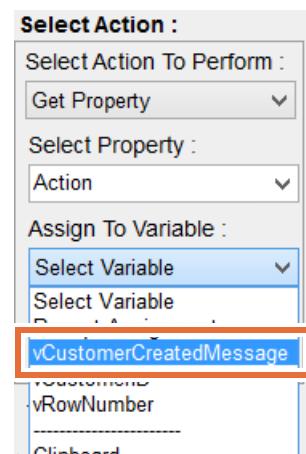
- ___ d. From the **Select Action To Perform** list, select **Get Property**.



- ___ e. From the **Select Property** list, select **Name**.



- ___ f. From the **Assign To Variable** list, select **vCustomerCreatedMessage**.



- ___ g. Change the value in the **Wait for the object to exist** field to: 5

The screenshot shows a horizontal input field with the placeholder text 'Wait for the object to exist:'. To its right is a small input box containing the number '5', followed by the word 'seconds'.

- ___ h. Click **Save**.

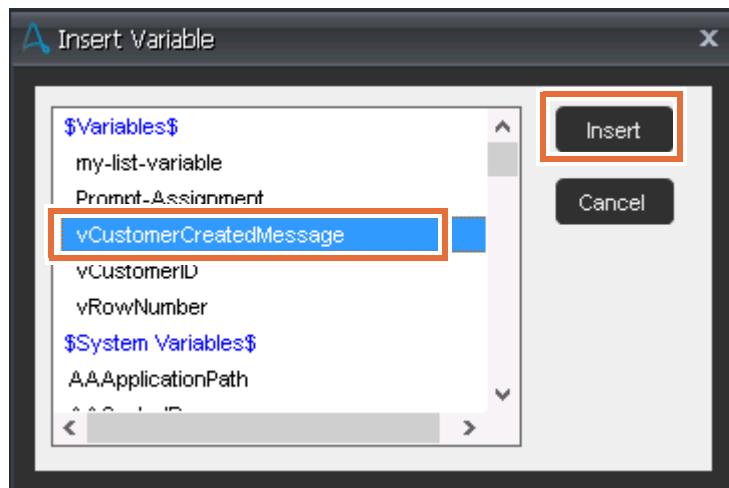
- ___ 4. Extract the customer reference number by using a **String Operation** command.

- ___ a. In the **Commands** list, expand **String Operation**, and add the **Sub String** command to the Workbench. Make sure that it is after the Object Cloning: Get Property "Name" command.

- ___ b. In the String Operation window, click the **Source String** field, and press F2 (or Fn+F2) to insert a variable.



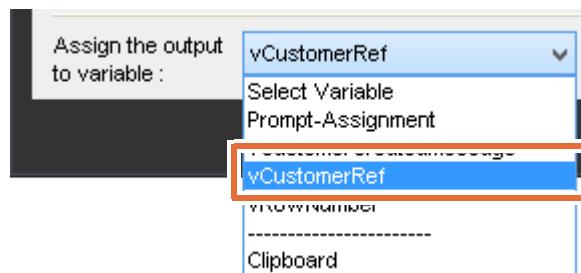
- ___ c. In the Insert Variable window, select **vCustomerCreatedMessage**, and click **Insert**.



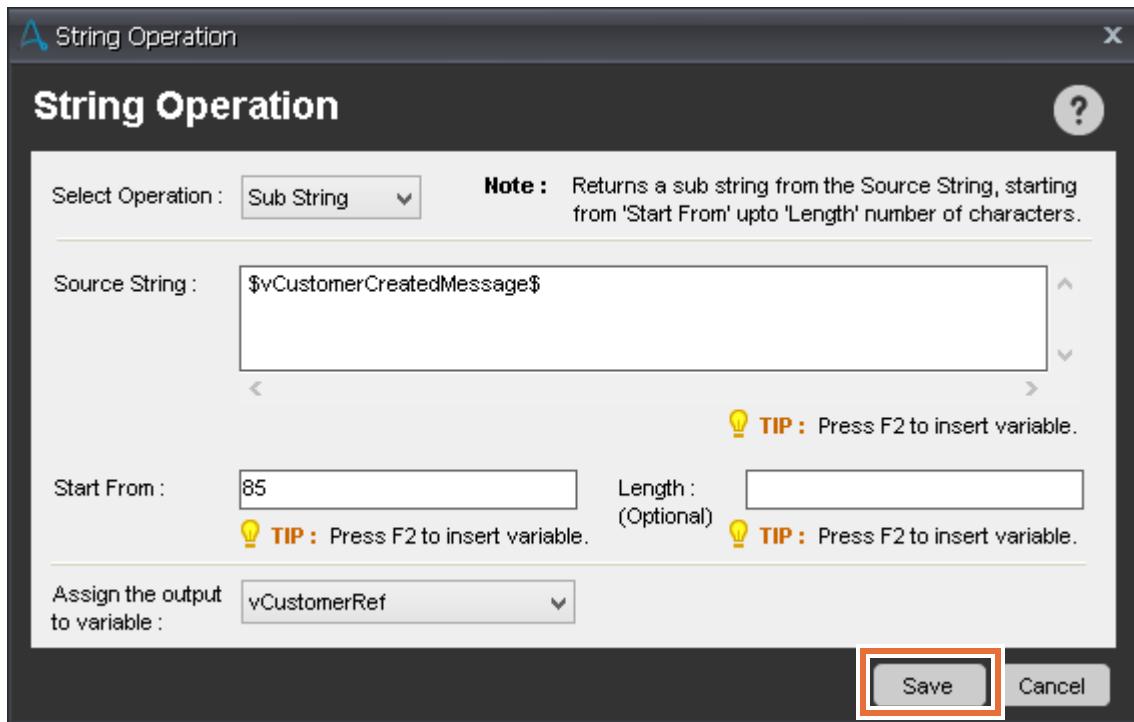
- ___ d. In the **Start From** field, enter: 85

This setting identifies the substring as starting at 85 characters from the beginning of the string, including spaces.

- ___ e. From the **Assign the output to variable** list, select **vCustomerRef**.



- ___ f. Click **Save**.



- ___ 5. Use the Smart Recorder to capture the closing of the Data Submitted window.
- In the Workbench, make sure that the String Operation command is selected.
 - On the Enterprise Client toolbar, click **Record**.
 - In the Recorder window, select **Data Submitted**, and click **Start**.
 - In the Data Submitted window, click **OK**.
 - In the Recording window, click **Stop**.

An Object Cloning command for clicking **OK** in the Data Submitted window should now be in the Workbench.

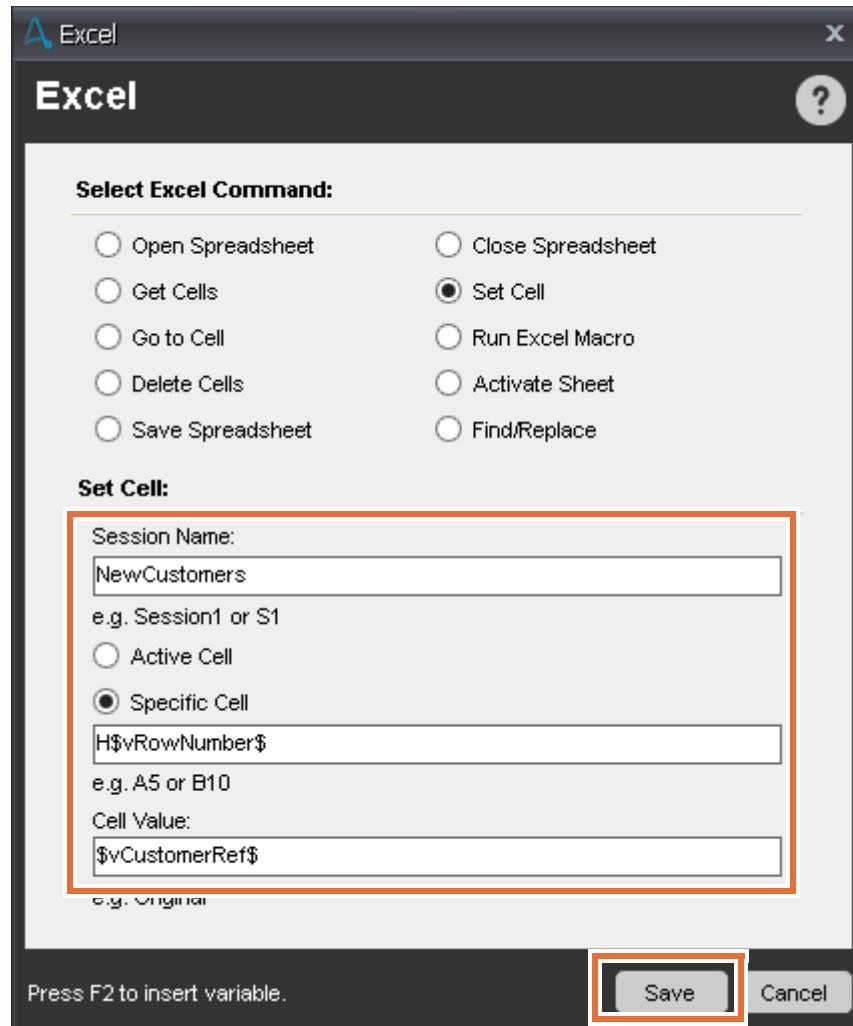
- ___ 6. Save your work.

4.4. Storing the customer reference number in the new_customers.xlsx spreadsheet

In this section, you store the newly extracted customer reference number in the new_customers.xlsx spreadsheet.

- In the Workbench, add the following comment after the final Object Cloning command:
Write the customer reference number in the Excel spreadsheet
- Add the **Excel > Set Cell** command after the comment that you entered in Step 1.

- ___ 3. Configure the Set Cell command to store the customer reference number in the correct spreadsheet column.
 - ___ a. Make sure that the **Session Name** is set to: NewCustomers
 - ___ b. In the Set Cell section, click **Specific Cell**, and enter the following values:
 - **Specific Cell:** H\$vRowNumber\$
 - **Cell Value:** \$vCustomerRef\$
 - ___ c. Click **Save**.



- ___ 4. Save your work.
- ___ 5. Close the SIB CRM window and the `new_customers.xlsx` window.

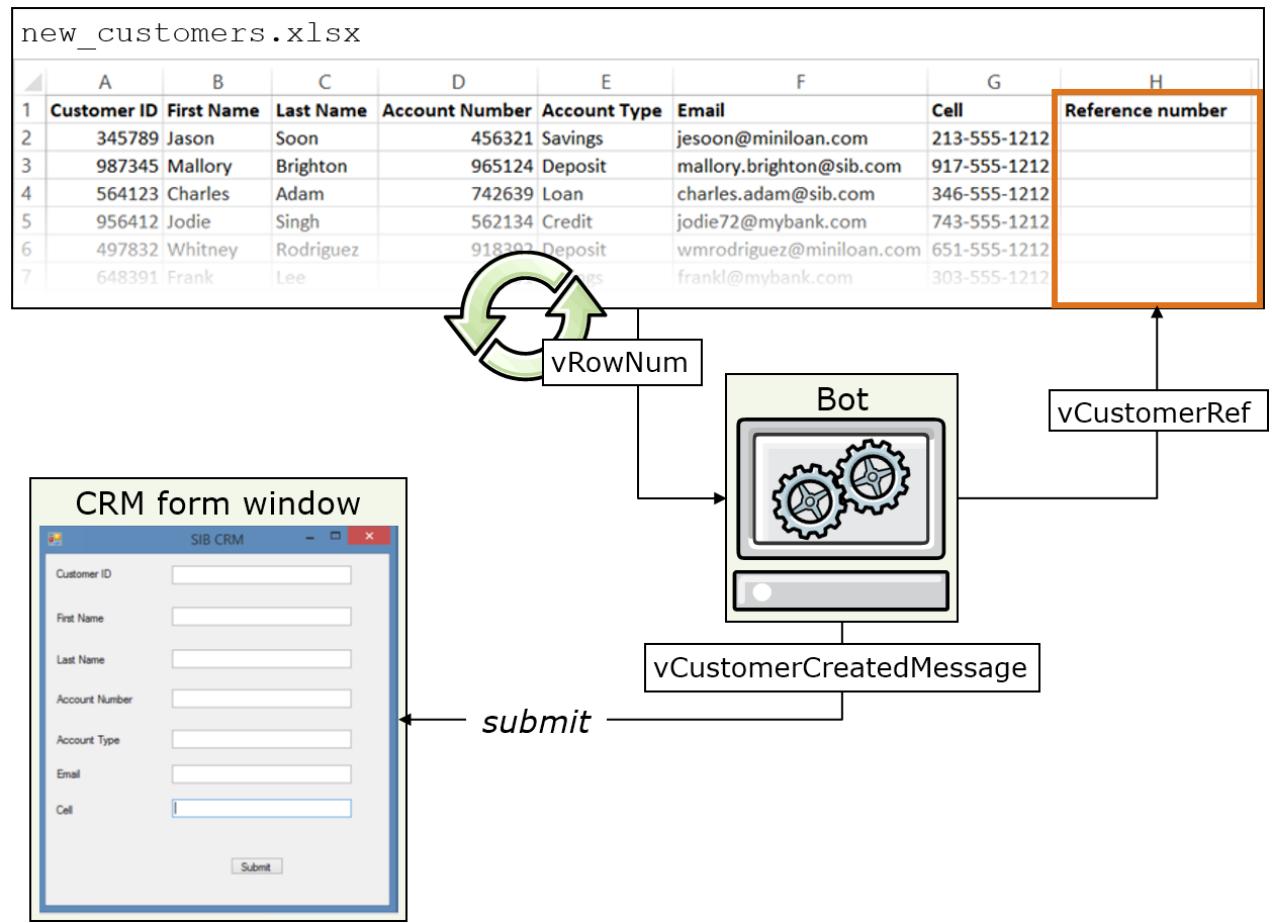
4.5. Testing your bot

Run the bot to make sure that you do not see any errors.

- ___ 1. On the Workbench toolbar, click **Run**.

Bot actions:

- Open and maximize the new_customers.xlsx spreadsheet
- Open the SIB CRM application window
- Write data from the new_customers.xlsx spreadsheet into the SIB CRM window fields and submit the form
- Extract the customer reference number from the Data Submitted window, and write it to the new_customers.xlsx spreadsheet
- Close the Data Submitted window



- 2. If you see errors, compare the identified Workbench line from the error message to the corresponding step in the lab instructions.
- 3. Close the SIB CRM window.
- 4. Close the new_customers.xlsx spreadsheet, but *do not save the changes*.

4.6. Inserting customer data into a database

In this section, you insert the customer data into the CUST_CROSS_REF table in the SIB DB2 database.

- ___ 1. In the Workbench, add the following comment after the Excel Set value of Cell command:
Insert customer data into database
- ___ 2. Add the Insert/Update/Delete database command, and configure it to insert customer data into the SIB database.
 - ___ a. From the **Commands** list, add **Database > Insert/Update/Delete** panel to the Workbench. Make sure that it is after the comment that you entered in Step 1.
 - ___ b. In the **Enter Insert/Update/Delete Statement** field, enter the following statement:

```
INSERT INTO DB2ADMIN.CUST_CROSS_REF VALUES ('$Excel Column(1)$', '$Excel Column(2)$', '$Excel Column(3)$', '$Excel Column(4)$', '$Excel Column(5)$', '$Excel Column(6)$', '$Excel Column(7)$', '$vCustomerRef$')
```



Note

You can copy the INSERT statement from the following file, and paste it into the Database command window:

C:\labfiles\account_opening\sqlStatements.txt



Important

If you copy and paste the INSERT statement from the sqlStatements.txt file, make sure that it does not include any line breaks.

- ___ c. Click **Save**.
- ___ 3. Save your work.

The loop is now complete.

Section 5. Finishing and running the bot

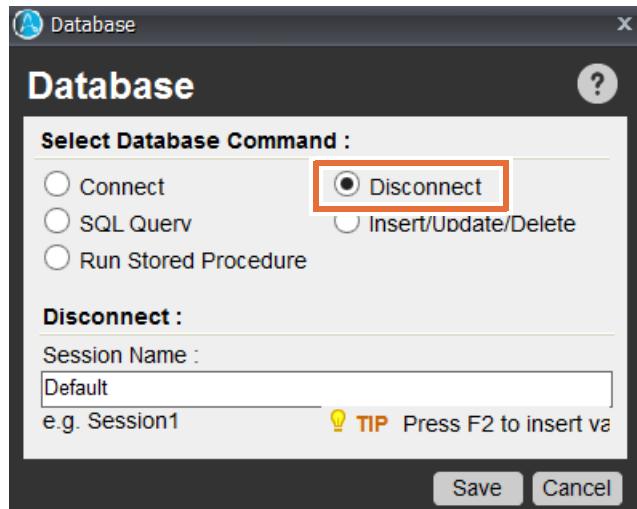
In this section, you define the final bot actions:

- Disconnect from the database
- Save and close the `new_customers.xlsx` spreadsheet
- Close the SIB CRM application window

You run the completed bot to make sure that it works. As a final step, you confirm that the bot wrote data to the database.

5.1. Completing the bot

- ___ 1. Add the command to disconnect from the database.
 - ___ a. From the **Commands** list, add the **Database > Disconnect** command. Make sure that it is after the `End Loop` command in the Workbench.
 - ___ b. In the Database window, confirm that **Disconnect** is selected, and click **Save**.



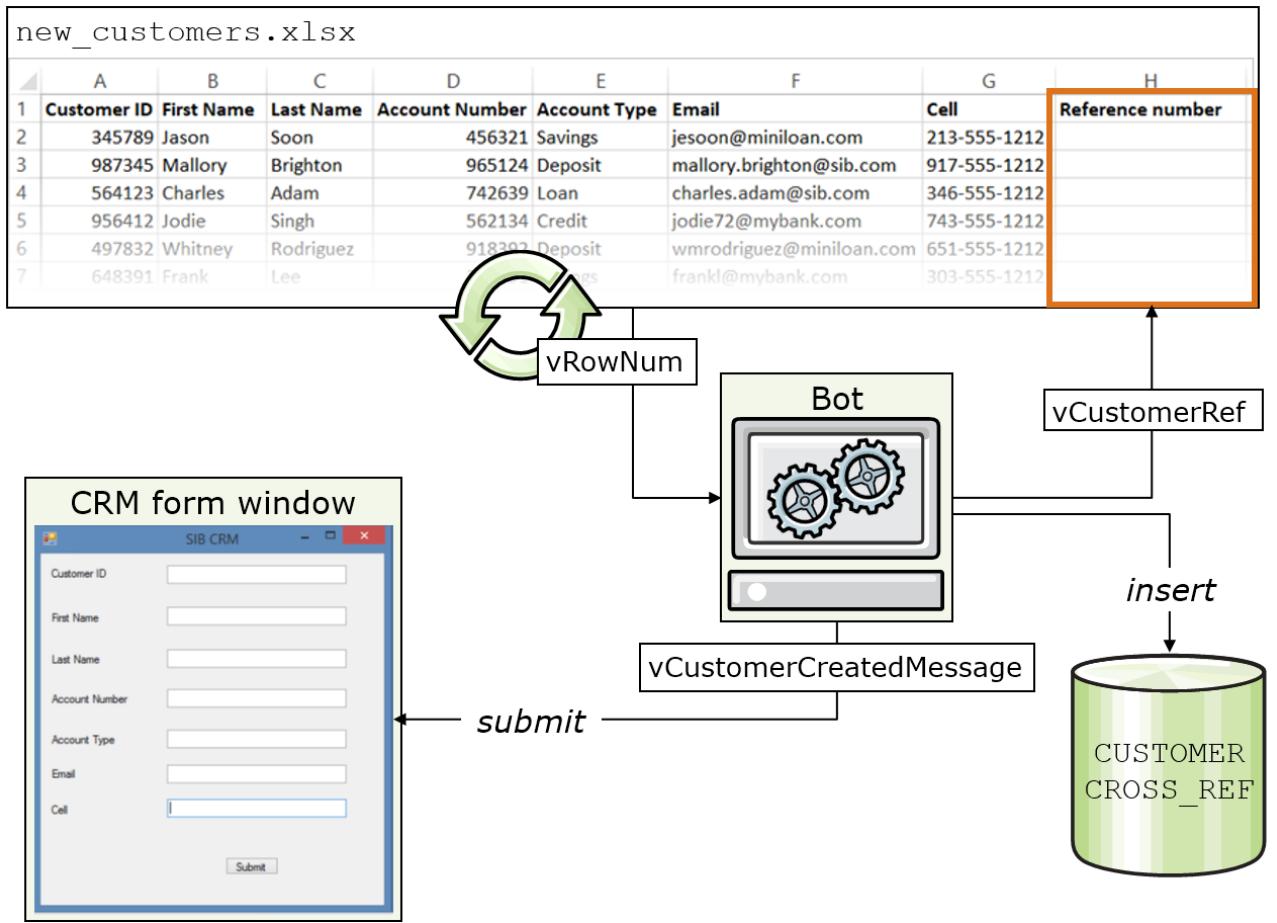
- ___ c. Confirm that this command is outside of the loop.
- ___ 2. Add commands to save and close the `new_customers.xlsx` spreadsheet.
 - ___ a. From the **Commands** list, add **Excel > Save Spreadsheet** after the last line in the Workbench, and click **Save**.
 - ___ b. From the **Commands** list, add **Excel > Close Spreadsheet** after the last line in the Workbench, and click **Save**.
- ___ 3. Add a command to close the SIB CRM application window.
 - ___ a. From the **Commands** list, add **Window Actions > Close Window** after the last line in the Workbench.
 - ___ b. From the **Select Window** list, select **SIB CRM** and click **Save**.
- ___ 4. Save your work.

5.2. Running the bot

- ___ 1. In the Workbench toolbar, click **Run** to run the completed bot.
 - ___ 2. If you see errors from running the bot, compare the identified problem line in the Actions List with the corresponding steps from the exercise.
-

Bot actions:

- Open and maximize the `new_customers.xlsx` spreadsheet
- Open the SIB CRM application window
- Connect to the database
- Write data from the `new_customers.xlsx` spreadsheet into the SIB CRM window fields and submit the form
- Extract the customer reference number from the Data Submitted window, write it to the `new_customers.xlsx` spreadsheet, and close the Data Submitted window
- Insert customer data into the database
- Disconnect from the database after all data from the `new_customers.xlsx` spreadsheet is processed
- Save and close the `new_customers.xlsx` spreadsheet
- Close the SIB CRM application window



5.3. Confirming the database INSERT operations

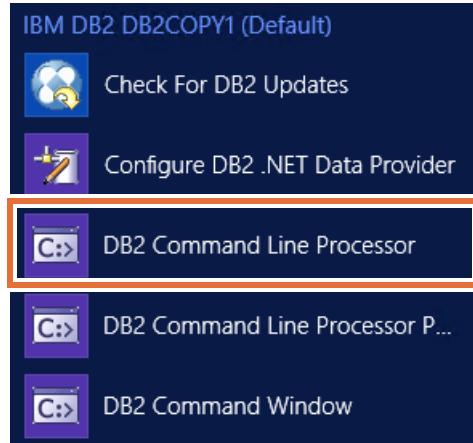
In this section, you confirm that the bot wrote the spreadsheet data to the database.



These instructions describe how to use the DB2 command line processor to query the database. A separate note at the end of this section describes how you can also use IBM Data Studio to view the data.

1. Open the DB2 command line processor.
 - a. Click the Windows icon at the lower-left of your screen, and click the down arrow to open the Apps list.

- ___ b. In the IBM DB2 DB2COPY1 (Default) section, click **DB2 Command Line Processor**.



2. Connect to the SIB database as the student user.

- ___ a. In the command line processor window, type:

```
CONNECT TO SIB USER student USING Education1
```



Note

You can copy the CONNECT statement from the following file, and paste it into the command line processor window:

```
C:\labfiles\account_opening\sqlStatements.txt
```

- ___ b. Press Enter.

The database responds with the connection information.

```
DB2 CLP - DB2COPY1 - C:\PROGRA~1\IBM\SQLLIB\BIN\db2setcp.bat DB2SETC...
db2 > CONNECT TO SIB USER student USING Education1
Database Connection Information
Database server      = DB2/NT64 11.1.3.3
SQL authorization ID = STUDENT
Local database alias = SIB
```

- ___ 3. Query the CUST_CROSS_REF table by typing the following command and pressing Enter.

```
SELECT * FROM DB2ADMIN.CUST_CROSS_REF
```

**Note**

You can copy the SELECT statement from the following file and paste it into the command line processor window:

C:\labfiles\account_opening\sqlStatements.txt

The database returns the data that was inserted by the bot.

```
DB2 CLP - DB2COPY1 - C:\PROGRA~1\IBM\SQLLIB\BIN\db2setcp.bat DB2SETC...
db2 => CONNECT TO SIB USER student USING Education1

Database Connection Information

Database server      = DB2/NT64 11.1.3.3
SQL authorization ID = STUDENT
Local database alias = SIB

db2 => SELECT * FROM CUST_CROSS_REF
SQL0204N "STUDENT.CUST_CROSS_REF" is an undefined name. SQLSTATE=42704
db2 => SELECT * FROM db2admin.CUST_CROSS_REF

CUSTID    FIRST_NAME          LAST_NAME          ACCT_NUM      ACCT_REF_NUM
YPE EMAIL
ER

345789    Jason Soon        456321 213-555-1212 Savin
s jesoon@miniloan.com          Brighton 965124 917-555-1212 Depos
t mallory.brighton@sib.com          Adam     742639 346-555-1212 Loan
564123    Charles Singh      562134 743-555-1212 Credi
charles.adam@sib.com          Rodriguez 918392 651-555-1212 Depos
956412    Jodie Singh       260491 303-555-1212 Savin
jodie72@mybank.com          Rodriguez 456321 213-555-1212 Savin
497832    Whitney Lee       965124 917-555-1212 Depos
t wmrodriguez@miniloan.com          Rodriguez 742639 346-555-1212 Loan
648391    Frank Lee         562134 743-555-1212 Credi
frankl@mybank.com          Rodriguez 918392 651-555-1212 Depos
345789    Jason Soon        260491 303-555-1212 Savin
s jesoon@miniloan.com          Rodriguez 456321 213-555-1212 Savin
987345    Mallory Brighton 965124 917-555-1212 Depos
t mallory.brighton@sib.com          Rodriguez 742639 346-555-1212 Loan
564123    Charles Singh      562134 743-555-1212 Credi
charles.adam@sib.com          Rodriguez 918392 651-555-1212 Depos
956412    Jodie Singh       260491 303-555-1212 Savin
jodie72@mybank.com          Rodriguez 456321 213-555-1212 Savin
497832    Whitney Lee       965124 917-555-1212 Depos
t wmrodriguez@miniloan.com          Rodriguez 742639 346-555-1212 Loan
648391    Frank Lee         562134 743-555-1212 Credi
frankl@mybank.com          Rodriguez 918392 651-555-1212 Depos

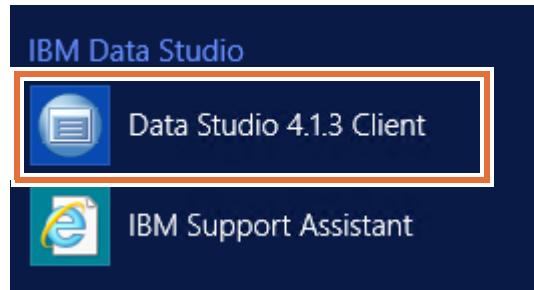
```

4. When you are finished reviewing the data, close the DB2 command line processor window.

**Note**

If you prefer to use a GUI-based application to check the data, you can use IBM Data Studio to view the data in the `CUST_CROSS_REF` table:

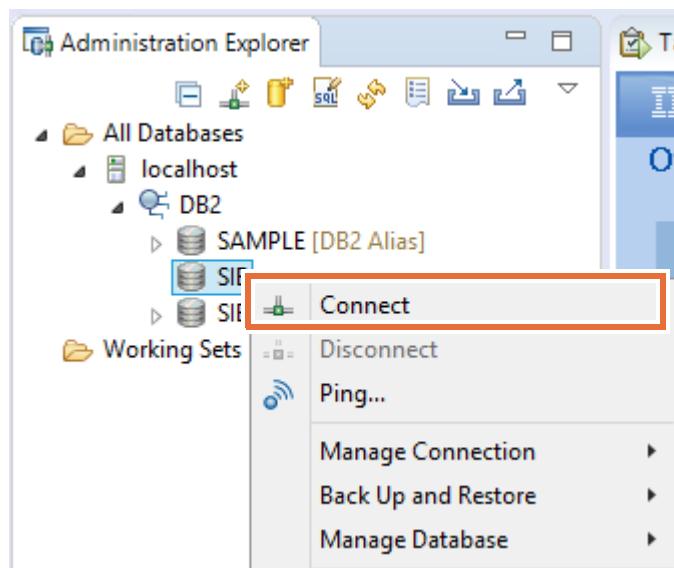
1. Click **Start** and then the down arrow to open the list of Apps.
2. In the IBM Data Studio section, click **Data Studio 4.1.3 Client**.



3. In the Eclipse Launcher window, click OK to accept the default workspace.
4. After the Data Studio window opens, make sure that you are in the **Database Administration** view.

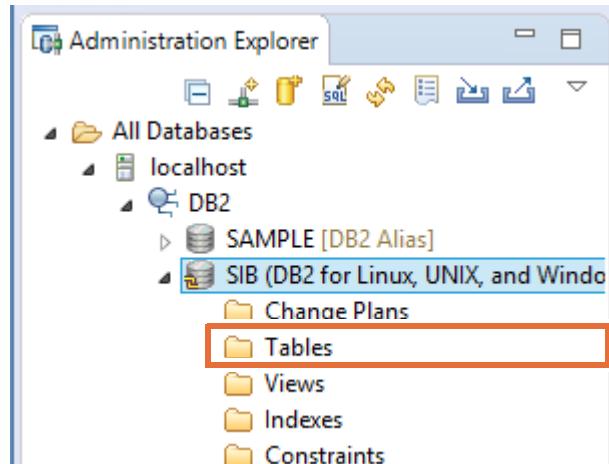


5. Connect to the SIB database.
 - ___ a. In the Administration Explorer pane, expand **All Databases > localhost > DB2**.
 - ___ b. Right-click **SIB** and click **Connect**.



- ___ c. If needed, in the **User name** and **Password** fields, enter `student \ Education1` and click **OK**.

6. In the Administration Explorer pane, expand **SIB** and click **Tables**.



7. In the list of tables in the main pane, right-click **CUST_CROSS_REF** and click **Data > Browse Data**.

Owner	Name	Organization	Perce...	Cardinality	Primary Key	Partition ...	Re
ADMIN	CUST_CROSS_REF	DB2_ORGANIZATION	1	6			U
ADMIN	CUST_MEMBER_LEVEL						U
IBM	SYSATTRIBUTES						SY
IBM	SYSAUDITEXCEPTION						SY
IBM	SYSAUDITPOLICIES						SY
IBM	SYSAUDITUSE						SY
IBM	SYSBUFFERPOOLNAME						SY
IBM	SYSBUFFERPOOLS						SY
IBM	SYSCHECKS						SY
IBM	SYSCODEPROPERTIES						SY
IBM	SYSCOLAUTH						SY

8. A new tab opens with the **CUST_CROSS_REF** table and its data.

	CUSTID [VARCHAR(10 OCTETS)]	FIRST_NAME [VARCHAR(20 OCTETS)]	LAST_NAME [VARCHAR(20 OCTETS)]
1	345789	Jason	Soon
2	987345	Mallory	Brighton
3	564123	Charles	Adam
4	956412	Jodie	Singh
5	497832	Whitney	Rodriguez
6	648391	Frank	Lee

9. When you are finished reviewing the data, close IBM Data Studio.

Section 6. *Optional:* Viewing the solution file

If you were unable to complete and run the bot successfully, you can open the solution file for this exercise to see how the bot should be coded.

- ___ 1. Go to C:\labfiles\account_opening\solution, right-click the Account Opening solution.atmx file and click **Edit**.

The file opens in the Workbench.

- ___ 2. To run the bot, make sure that the database connection is set.

- ___ a. Double-click the **Connect** command in the Actions List.

```

1  Comment: Open the Excel spreadsheet
2  Excel: Open Spreadsheet "C:\labfiles\account_opening\new_customers.xlsx". ActiveSheet: "Default". Contains Header: "True"
3  Maximize Window: "Microsoft Excel - new_customers.xlsx"
4  Excel: Get All Cells Session: NewCustomers
5  Connect to "$DB_CONNECT_STRING(DB2_CONNECT_STRING)$" Session:'Default'
| 6  If Window Does Not Exist ("SIB CRM") Then

```

- ___ b. Reenter the connection as described in [Step 2 of Section 2.2, "Connecting to the database"](#).
- ___ c. Click **Save** on the Workbench toolbar.
- ___ d. Click **Run** on the Workbench toolbar.

End of exercise

Exercise review and wrap-up

In this exercise, you created a task iterate through the customer data in the spreadsheet, submit the data to the CRM application, and insert the data to a database.

Exercise 5. Creating a bot to sum check declines, query a database, and send an email

Estimated time

02:00

Overview

In this exercise, you learn how to trigger a bot that opens a CSV file, queries a database, and sends an email.

Objectives

After completing this exercise, you should be able to:

- Extract values from files
- Use variables to perform calculations
- Work with the `Email` command
- Configure a bot trigger

Introduction

In this exercise, you create a bot that extracts values from a CSV file, calculates the sums for each account, pulls customer membership levels from a database for each account, and combines results in an email.

The exercise includes these sections:

- [Section 1, "Creating a bot and adding custom variables"](#)
- [Section 2, "Extracting values from a file and calculating totals"](#)
- [Section 3, "Extracting information from a database"](#)
- [Section 4, "Sending results by email"](#)
- [Section 5, "Configuring the trigger"](#)
- [Section 6, "Running the bot and confirming email delivery"](#)

Section 1. Creating a bot and adding custom variables

In this part of the exercise, you create a task and create custom variables.

1.1. Creating a task

- 1. Go to the Enterprise Client.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

- 2. In the Enterprise Client, create a task with the Workbench, by clicking **New** on the toolbar and selecting **Workbench** when prompted for a tool option.

1.2. Creating the value variables for the bot

You can create four types of variables: Value, List, Array, and Random. First, you create the following Value variables.

Variable	Default	Description
vAcctRow	1	Tracks rows in the vCheckSum array
vAcctToLookup		Used in an SQL statement to pass account numbers
vCustLevel		Contains the customer level that is retrieved from the database
vEmailTable		String value that is included in the email with the results
vNextAcctNum	2	Determines when to sum checks by account
vTotalAccts		Total tally of all distinct account numbers in CSV file and determines the number of loops when pulling values from database

- 1. In the Workbench, open the **Variable Manager**.

___ 2. Create the vAcctRow variable.

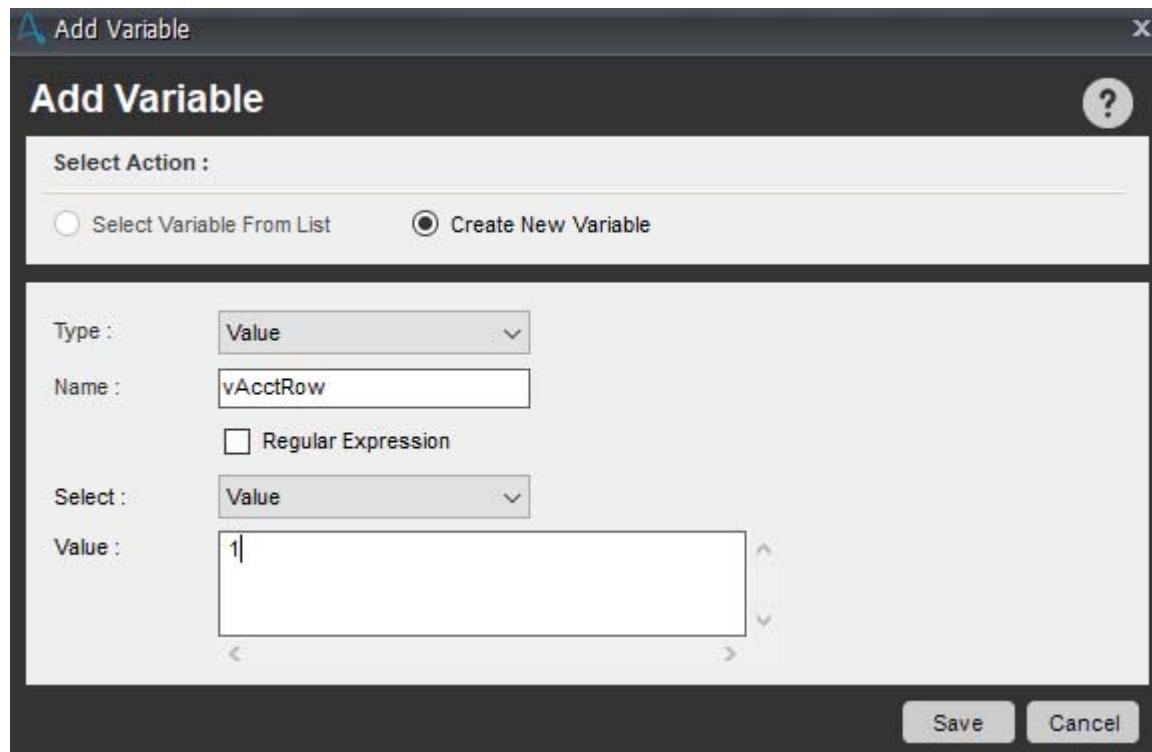
___ a. In the Variable Manager pane, click **Add**.



___ b. In the **Name** field, type: vAcctRow

___ c. In the **Value** field, enter: 1

__ d. Click **Save**.



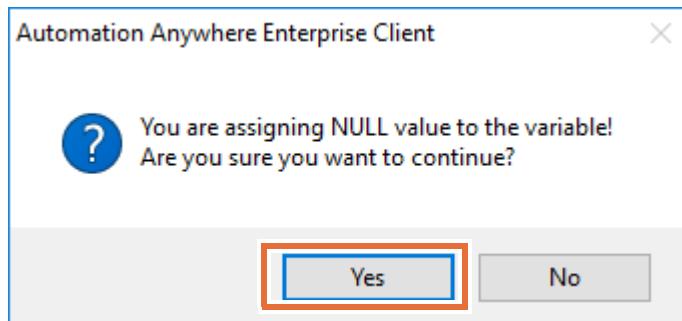
__ 3. Repeat [Step 2](#) to create the remaining value variables.

Variable	Value
vAcctToLookup	
vCustLevel	
vEmailTable	
vNextAcctNum	2
vTotalAccts	



Note

For the variables that have no default value, a dialog box opens. You can click **Yes** to close it.

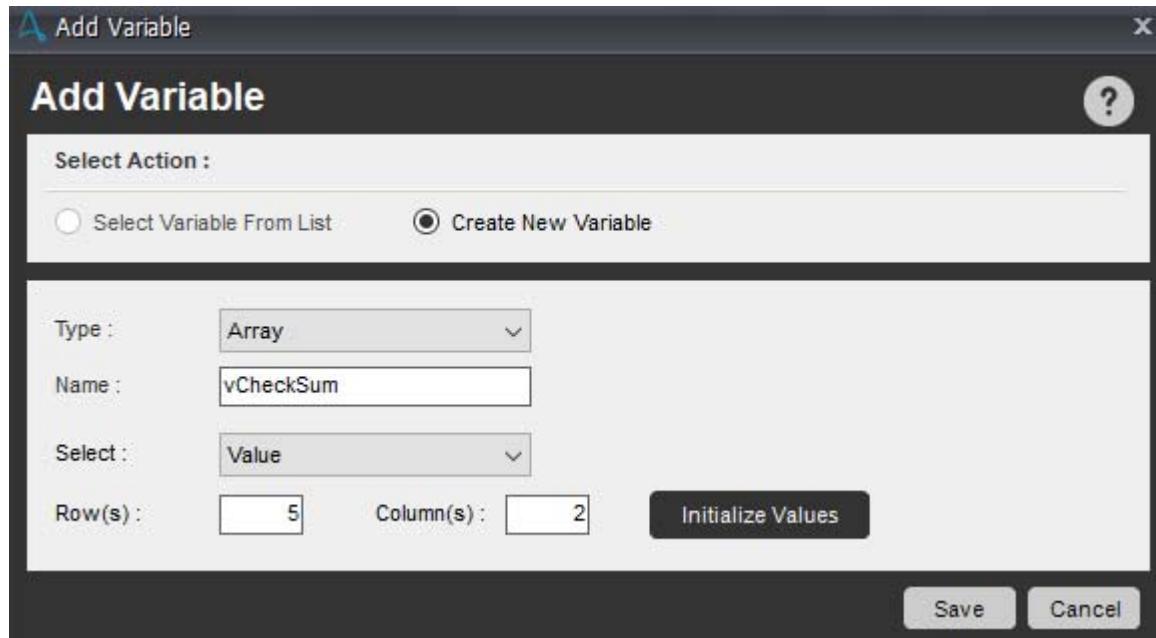


1.3. Creating the array variables

Next, you create these array variables.

Variable	Description
vCheckSum	Contains all check sums by account
vCheckDeclines	Contains raw check decline data that is pulled from CSV file

- ___ 1. Create the vCheckSum variable.
 - ___ a. In the Variable Manager, click **Add**.
 - ___ b. Set the following fields and values:
 - **Type:** Array
 - **Name:** vCheckSum
 - **Row:** 5
 - **Column:** 2
 - ___ c. Click **Save**.



- ___ 2. Create the vCheckDeclines variable.
 - ___ a. In the Variable Manager, click **Add**.
 - ___ b. Set the following fields and values:
 - **Type:** Array
 - **Name:** vCheckDeclines
 - **Select:** Read from excel/csv file
 - **Select File:** Browse to the C:\labfiles\check_declines\CHECK_DECLINES.csv file

___ c. Click **Save**.

The screenshot shows the Variable Manager dialog box. The 'Type' dropdown is set to 'Array'. The 'Name' field contains 'vCheckDeclines'. The 'Select' dropdown is set to 'Read from excel/csv file'. The 'Select File' field shows the path 'C:\labfiles\check_declines\CHECK_DECLINES.csv'. Below this are buttons for 'Open File' and 'How to use?'. At the bottom, there are radio buttons for 'All Cells' (selected), 'Entire Row', 'Entire Column', and 'Range'. The 'Save' and 'Cancel' buttons are at the bottom right.

1.4. Creating list variables

You create the following list variable to contain the results.

Variable	Description
vResultSet	Dynamically builds the result set to be included in the email

- ___ 1. In the Variable Manager, click **Add**.
- ___ 2. Set these fields:
 - **Type:** List
 - **Name:** vResultSet
- ___ 3. In **List Value**, enter 1
- ___ 4. Click **Add To List**, enter 2 and repeat to enter 3 and 4.
- ___ 5. Click **Save**.

The screenshot shows the Variable Manager dialog box for a list variable. The 'Type' dropdown is set to 'List'. The 'Name' field contains 'vResultSet'. The 'Select' dropdown is set to 'Value'. On the right, the 'List Value' field contains a list box with values 1, 2, 3, and 4. At the bottom, there are buttons for 'Add To List' (which is highlighted in blue), 'Edit', and 'Delete'. The 'Save' and 'Cancel' buttons are at the bottom right.

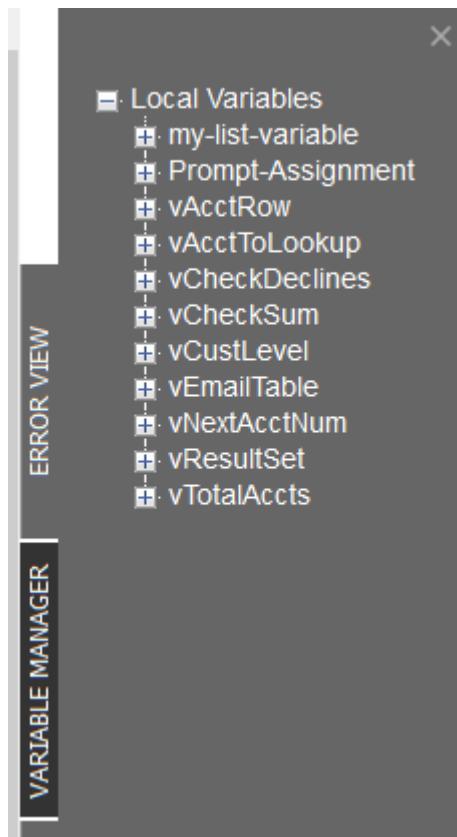


Information

By adding “1, 2, 3, 4” in the List Value field, the list variable is initialized for use with four items. The items are not required to be in a particular order.

1.5. Saving your work

- 1. Review the list of variables in the Variable Manager to make sure that you added all nine custom variables (prefixed with “v”).



- 2. Save your work by clicking **Save** on the toolbar.



- 3. When prompted, enter **Check Declines** for the **Filename** field and click **Save**.



Reminder

Save your work often.

Section 2. Extracting values from a file and calculating totals

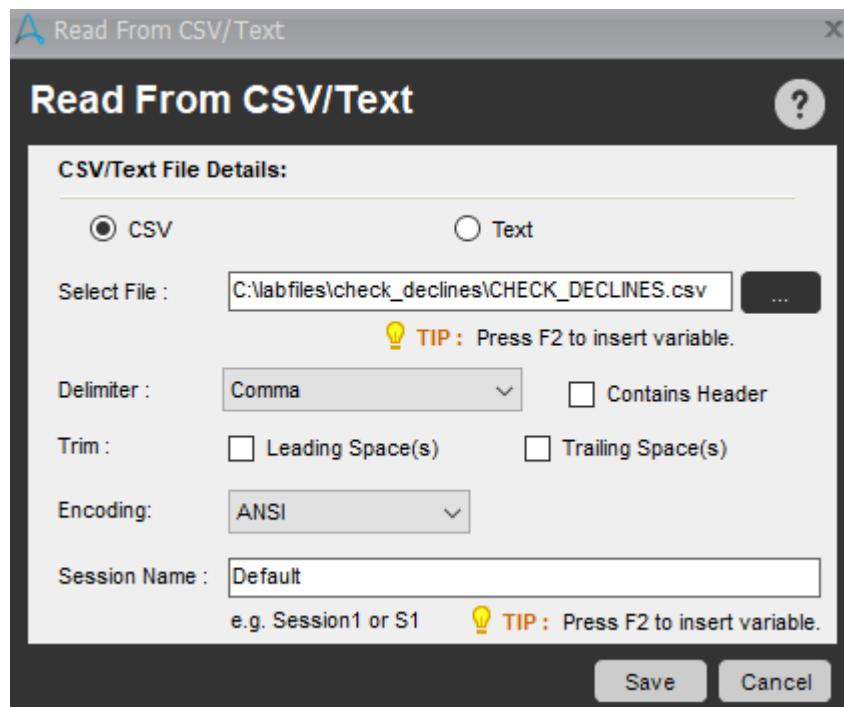
2.1. Automating extraction of values from a CSV file

- ___ 1. Add a comment to your bot.

 - ___ a. Double-click the **Comment** command in the **Commands** list and enter the following text:
Pull check values from file and add to running total for each account then add to result set.
 - ___ b. Click **Save**.

- ___ 2. Configure a command to read from the CSV file.

 - ___ a. In the **Commands** list, double-click **Read From CSV/Text**.
 - ___ b. In **Select File**, enter: C:\labfiles\check_declines\CHECK_DECLINES.csv
Alternatively, you can click the ellipses (...) and go to the file.
 - ___ c. Click **Save**.

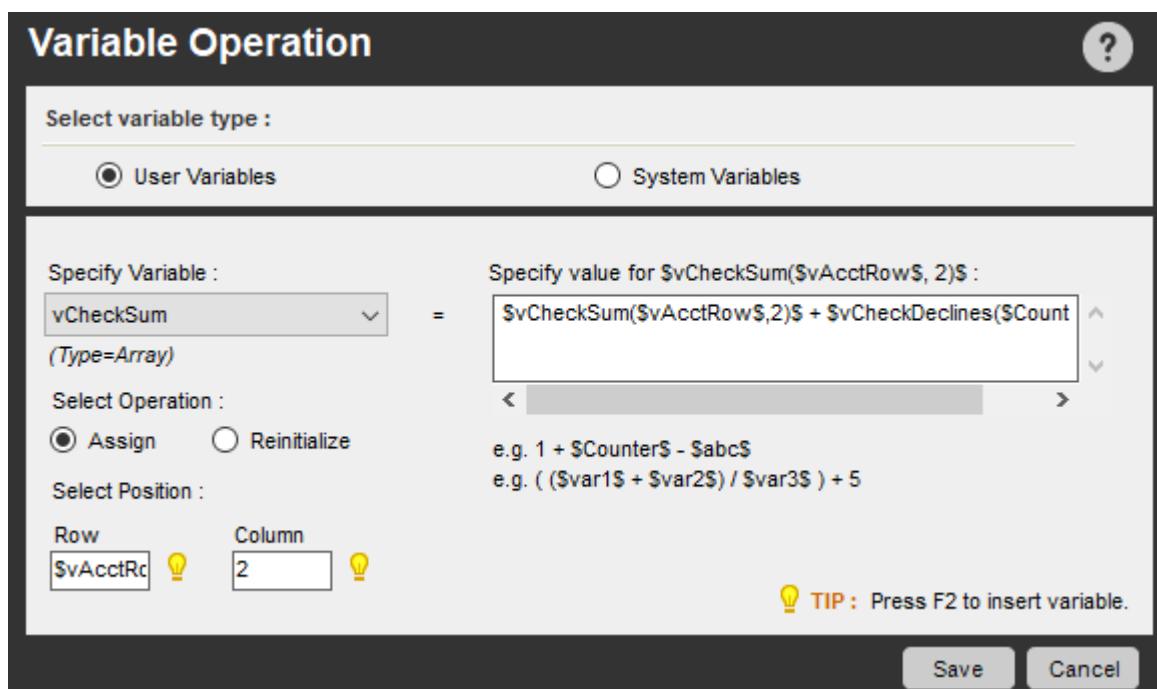


The Start Loop command is automatically entered in the bot.



2.2. Calculating a running total

- ___ 1. Double-click the **Comment** line that was added in the Start Loop command, and change the text to:
Add check values to running total.
- ___ 2. Configure a Variable Operation command to add check values to running total.
 - ___ a. In the **Commands** list, double-click the **Variable Operation** command.
 - ___ b. From the **Specify Variable** list, select **vCheckSum**.
 - ___ c. In the **Row** field, place the cursor in the **Row** field, press F2 (or Fn+F2), select **vAcctRow** and click **Insert**.
 - ___ d. In the **Column** field, enter: 2
 - ___ e. In the **Specify value** field, enter:
`$vCheckSum($vAcctRow$, 2)$ + $vCheckDeclines($Counter$, 2)$`
 - ___ f. Click **Save**.



The Variable Operation should appear after the comment line but before the End Loop line.

```

| Read From CSV file: "C:\labfiles\check_declines\CHECK_DECLINES.csv" Delimiter: "Comma" Header: "Yes"
| Start Loop "Each row in a CSV/Text file of Session: Default"
|   Comment: Add check values to running total.
|     Variable Operation: $vCheckSum($vAcctRow$,2)$ + $vCheckDeclines($Counter$,2)$ To $vCheckSum
|   End Loop

```



Hint

You can also drag commands to the Workbench and rearrange commands.

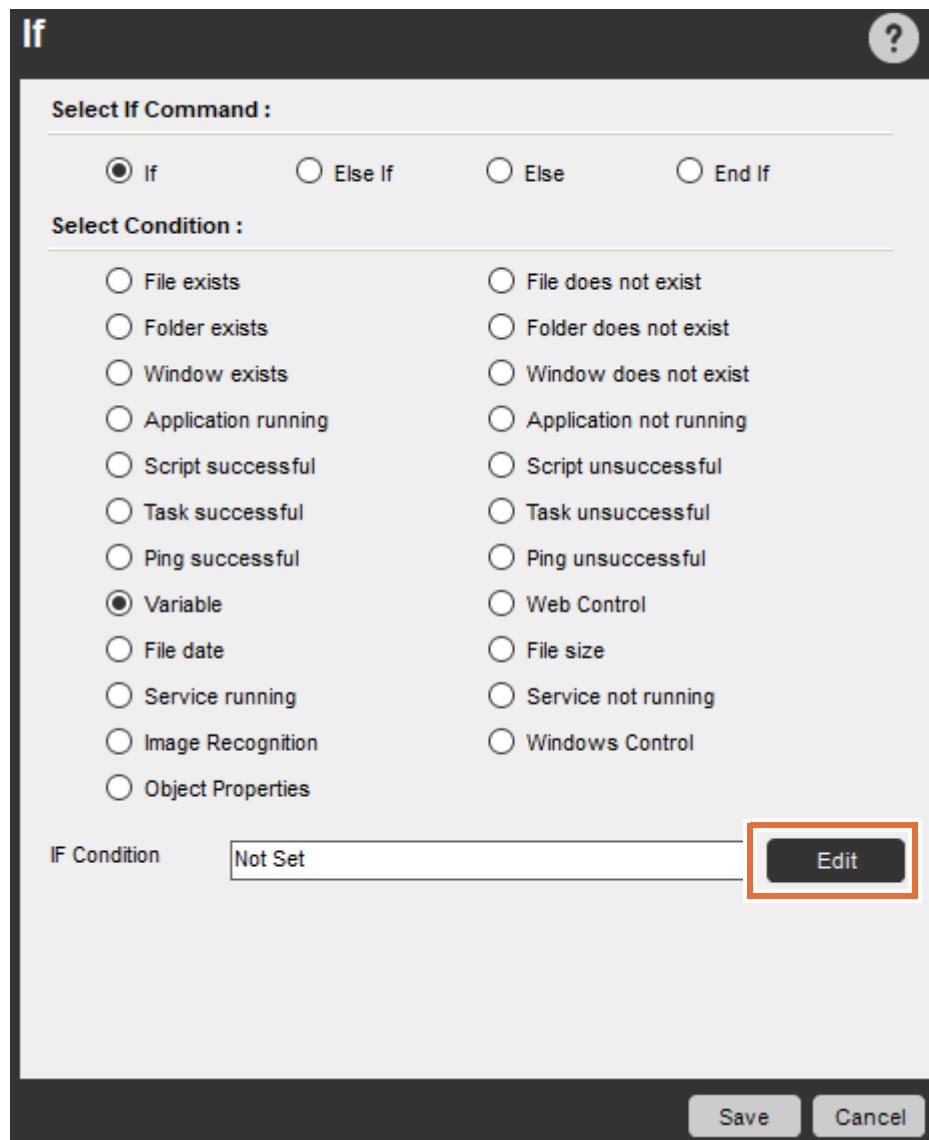
- ___ 3. Add an **If/Else** command to evaluate whether the current check value is the last one for the account.
 - ___ a. Make sure that the Variable Operation line is highlighted in the bot editor.

```
1  Comment: Pull check values from file and add to running total for each account then add to result set.
2  Read From CSV file: "C:\labfiles\Check_Declines\CHECK_DECLINES.csv" Delimiter: "Comma" Header:ter: "Comma" Header:
3  Start Loop "Each row in a CSV/Text file of Session: Default"
4  Comment: Add check values to running total.
5  $ Variable Operation: $vCheckSum($vAcctRow$,2$) + $vCheckDeclines($Counter$,2$) To $vCheckSum($vAcctRow$, 2$)
6  ✎ End Loop
```

- ___ b. In the **Commands** list, double-click **If/Else** and then double-click **Variable**.

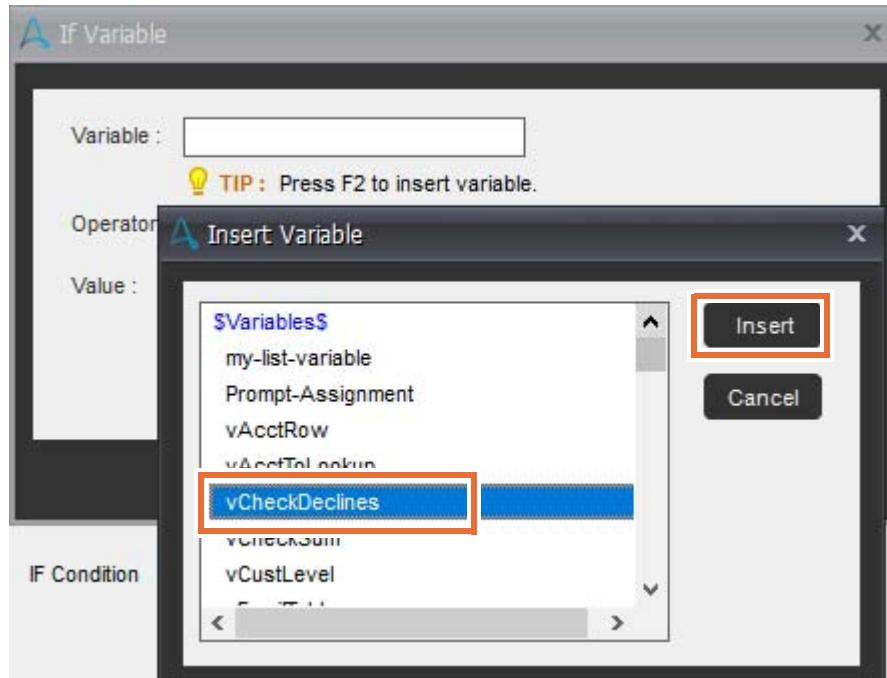
___ 4. Configure the If condition in the If dialog box.

___ a. For the **IF Condition** field, click **Edit**.

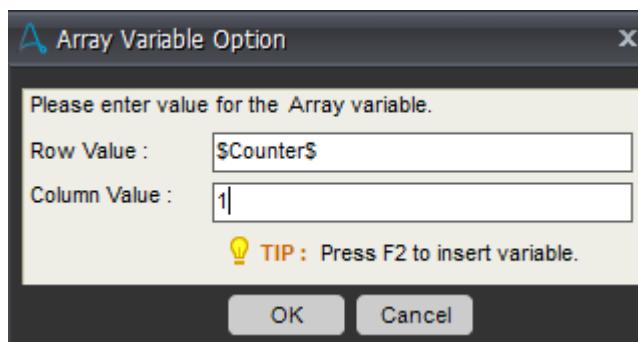


___ b. Place the cursor in the **Variable** field and press F2 (or Fn+F2).

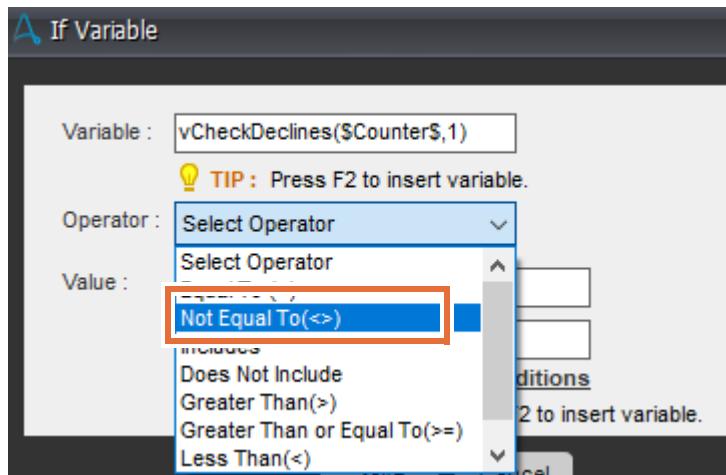
- ___ c. In the Insert Variable dialog box, select **vCheckDeclines** and click **Insert**.



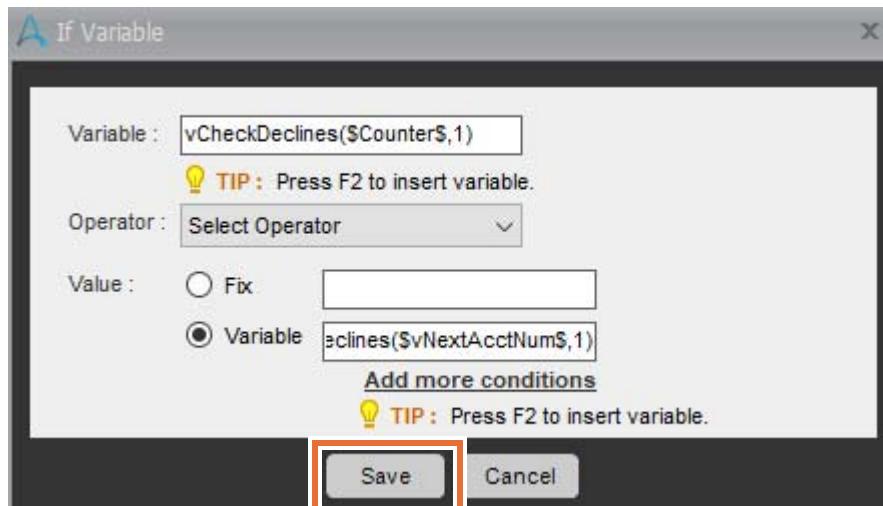
- ___ d. In the Array Variable Option dialog box, place your cursor in the **Row Value** field and press F2 (or Fn+F2).
The Array Variable Option dialog box opens.
___ e. In the **\$System Variables\$** part of the list, select **Counter** and click **OK**.
___ f. In the **Column Value** field, enter **1**, and click **OK**.



- ___ g. In the If Variable dialog box, from the **Operator** list, select **Not Equal To(<>)**.



- ___ h. In the **Value** options, choose **Variable**.
 ___ i. Click the **Variable** field, press F2 (or Fn+F2), select **vCheckDeclines**, and click **Insert**.
 ___ j. In the Array Variable Option dialog box, place the cursor in the **Row Value** field, press F2 (or Fn+F2), select **vNextAcctNum**, and click **OK**.
 ___ k. In the **Column Value** field, enter 1 and click **OK**.
 ___ l. Click **Save**.



- ___ 5. Verify that the If condition is added after the Variable Operation.

```

  Comment: Pull check values from file and add to running total for each account then add to result set.
  Read From CSV file: "C:\Labfiles\check_declines\CHECK_DECLINES.csv" Delimiter: "Comma" Header: "No" Trim Leading Spaces
  Start Loop "Each row in a CSV/Text file of Session: Default"
  Comment: Add check values to running total.
  Variable Operation: $vCheckSum($vAcctRow$, 2)$ + $vCheckDeclines($Counter$, 2)$ To $vCheckSum($vAcctRow$, 2)$
IF If $vCheckDeclines($Counter$,1)$ Not Equal To (<>) $vCheckDeclines($vNextAcctNum$,1)$ Then
  Comment: Please enter the conditional commands here
End If

```

6. Add Variable Operation commands to be performed for each account at the final check in the array of check declines.

Variable	Purpose
vCheckSum	Array of all check sums by account
vResultSet	Used to dynamically build the result set to be pasted into the email
vAcctRow	Used to track rows in the vCheckSum array
vTotalAccts	Total tally of all distinct account numbers in CSV file that is used to determine number of loops when retrieving values from the database

- a. Double-click the comment after the **If** command and enter the following text:
 If this is the last check for the account: - Add final check value - Append to result set - Increment Account number - Add to total accounts.
- b. Click **Save**.
7. Add the vCheckSum variable.
- In the **Commands** list, double-click **Variable Operation**.
 - For **Specify Variable**, select **vCheckSum**.
 - For **Select Position**, click the **Row** field, and press F2 (or Fn+F2).
 - Select **vAcctRow** and click **Insert**.
 - In the **Column** field, enter: 1
 - In the **Specify value** field, enter: \$vCheckDeclines(\$Counter\$,1)\$
 - Click **Save**.
8. Add the vResultSet variable.
- In the **Commands** list, double-click **Variable Operation**.
 - For **Specify Variable**, select **vResultSet**.
 - For **Select Operation**, make sure **Assign** is selected.
 - Click the **Select Position** field and press F2 (or Fn+F2).
 - Select **vAcctRow** and click **Insert**.
 - In the **Specify value** field, enter:

$$\$vCheckSum(\$vAcctRow\$,1)\$ \mid \$vCheckSum(\$vAcctRow\$,2)\$$$
 - Click **Save**.
9. Add the vAcctRow variable.
- In the **Commands** list, double-click **Variable Operation**.
 - For **Specify Variable**, select **vAcctRow**.
 - For **Select Operation**, make sure that **Assign** is selected.
 - In the **Specify value** field, enter: \$vAcctRow\$ + 1

- e. Click **Save**.
10. Add the **vTotalAccts** variable.
- In the **Commands** list, double-click **Variable Operation**.
 - For **Specify Variable**, select **vTotalAccts**.
 - For **Select Operation**, make sure that **Assign** is selected.
 - In the **Specify value** field, enter: `$vTotalAccts$ + 1`
 - Click **Save**.
11. Add a comment that the bot increments the account number.
- In the Workbench, make sure that the `End If` command is highlighted.
 - In the **Commands** list, double-click **Comment**.
 - In the Comment window, type:
Increment account number
 - Click **Save**.
12. Add a Variable Operation commands to increment the account number.
- In the **Commands** list, double-click **Variable Operation**.
 - For **Specify Variable**, select **vNextAcctNum**.
 - For **Select Operation**, make sure that **Assign** is selected.
 - In the **Specify value** field, enter: `$vNextAcctNum$ + 1`
 - Click **Save**.
13. Verify that the **vNextAcctNum** Variable Operation command is added before the End Loop line.

```

1   Comment: Pull check values from file and add to running total for each account then add to result set.
2   Read From CSV file: "C:\Labfiles\check_declines\CHECK_DECLINES.csv" Delimiter: "Comma" Header: "No" Trim Leading
3   Start Loop "Each row in a CSV/Text file of Session: Default"
4   Comment: Add check values to running total.
5   Variable Operation: $vCheckSum($vAcctRow$,2)$ + $vCheckDeclines($Counter$,2)$ To $vCheckSum($vAcctRow$,
6   If $vCheckDeclines($Counter$,1)$ Not Equal To (<>) $vCheckDeclines($vNextAcctNum$,1)$ Then
7   Comment: If this is the last check for the account: - Add final check value - Append to result set - Increment Account
8   Variable Operation: $vCheckDeclines($Counter$,1)$ To $vCheckSum($vAcctRow$, 1)$
9   Variable Operation: $vCheckSum($vAcctRow$,1)$ | $vCheckSum($vAcctRow$,2)$ To $vResultSet($vAcctRow$)$
10  Variable Operation: $vAcctRow$ + 1 To $vAcctRow$
11  Variable Operation: $vTotalAccts$ + 1 To $vTotalAccts$
12  End If
13  Comment: Increment account number.
14  Variable Operation: $vNextAcctNum$ + 1 To $vNextAcctNum$
15  End Loop

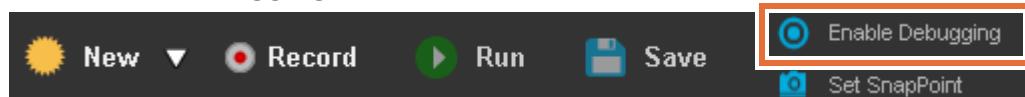
```

14. Save the bot.

2.3. Testing the bot

In this part of the exercise, you test the bot by running it in the debugging mode.

- ___ 1. Click **Enable Debugging** on the toolbar.



- ___ 2. Add the variables to watch.

- ___ a. In the Variable Watch Table, click **Add**.
- ___ b. Select the following variables and click **Add**.

- vAcctRow
- vCheckDeclines
- vCheckSum
- vResultSet
- vTotalAccts

Add Variable(s) In Watch

Select variable(s) to add in watch :

Variable Name	Return Value	Description
Local Variables		
<input type="checkbox"/> my-list-variable	List	
<input type="checkbox"/> Prompt-Assignment	Value	
<input checked="" type="checkbox"/> vAcctRow	Value	
<input type="checkbox"/> vAcctToLookup	Value	
<input checked="" type="checkbox"/> vCheckDeclines	Array	
<input checked="" type="checkbox"/> vCheckSum	Array	
<input type="checkbox"/> vCustLevel	Value	
<input type="checkbox"/> vEmailTable	Value	
<input type="checkbox"/> vNextAcctNum	Value	
<input checked="" type="checkbox"/> vResultSet	List	
<input checked="" type="checkbox"/> vTotalAccts	Value	
System Variables		
<input type="checkbox"/> Year	Date/Time	Returns Year. e.g. if date is 02/23/04, it returns 2004.
<input type="checkbox"/> Month	Date/Time	Returns System Month. e.g. if date is 02/23/04, it returns 2.
<input type="checkbox"/> Day	Date/Time	Returns System Day. e.g. if date is 02/23/04, it returns 23.
<input type="checkbox"/> Date	Date/Time	Returns System Date in mm/dd/yyyy hh:mm:ss format.
<input type="checkbox"/> Hour	Date/Time	Returns System Hour. e.g. if time is 17:33:49, it returns 17.
< Back		
<input type="checkbox"/> Select All		

- ___ 3. On the toolbar, click **Run**.

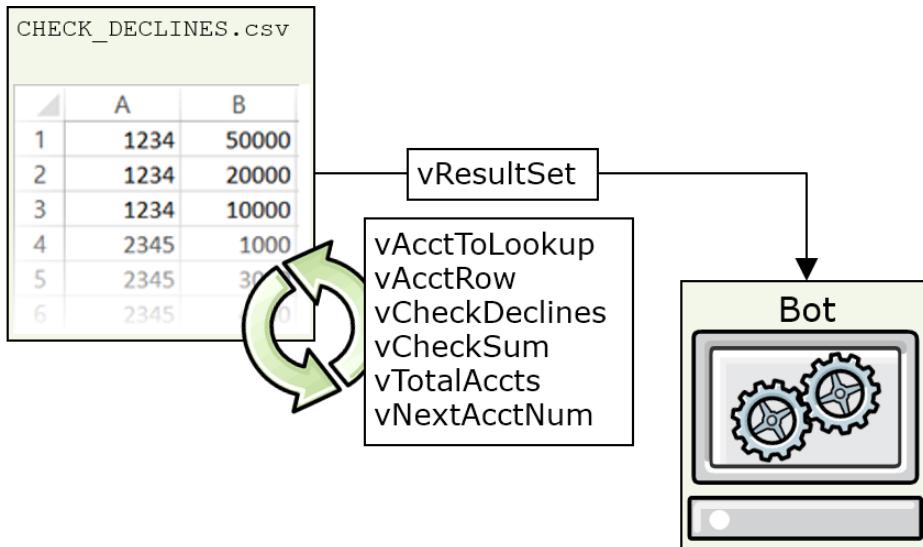
As the bot runs, it populates the watch table with values. You see the runtime window step through each line of code. You might need to move the watch table to view the lines of code that are being processed.

After running, you see the following values in the watch table:

vAcctRow	4
vCheckDeclines	3456
vCheckSum	20000
vResultSet	1234 80000
vTotalAccts	3

Bot tasks:

- Pulls check decline values from the `CHECK_DECLINES.csv` file
- Calculates the sum for each account



- ___ 4. Close the Variable Watch Table.
- ___ 5. On the toolbar, click **Disable Debugging**.



Troubleshooting

If you experience performance problems while running the task in debug mode, try running the task without using debug mode to verify that the bot has no errors. If the Enterprise Client shows an error message, fix the error by reviewing the identified line in the error message and comparing it to the corresponding exercise step.



Information

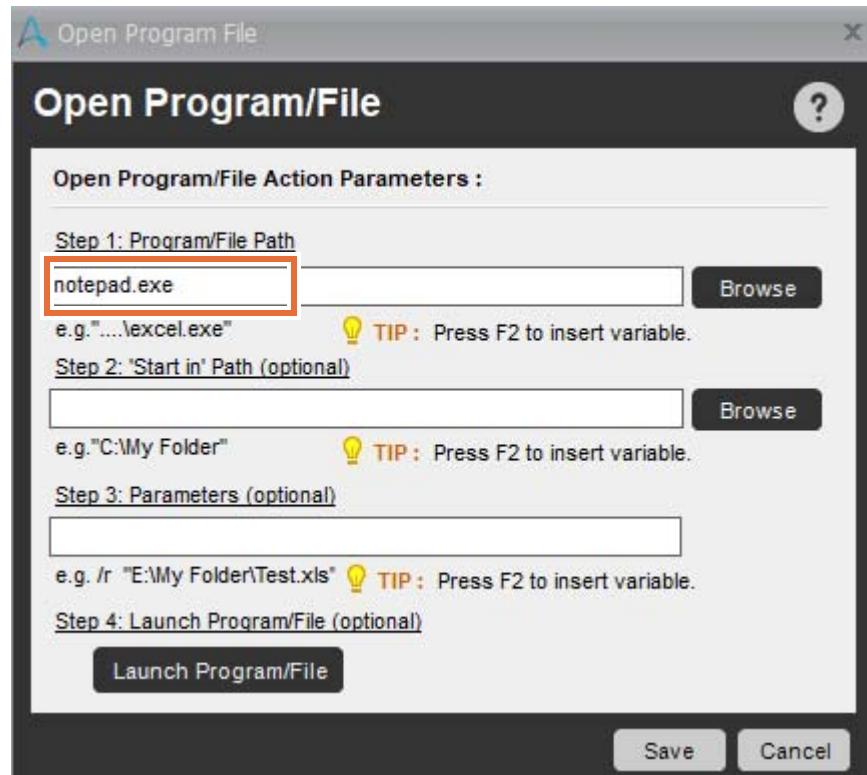
You can toggle the Debug mode by clicking **Enable Debugging/Disable Debugging** at any time throughout the course of this lab.

Section 3. Extracting information from a database

In this section, you extract customer-level information from a database for each account and record that information in Notepad.

3.1. Automating the opening of a text editor

- ___ 1. Add a comment.
 - ___ a. Make sure that the End Loop line is highlighted.
 - ___ b. In the **Commands** list, double-click the **Comment** line and enter the following text:
Pull customer level from database for each account.
 - ___ c. Click **Save**.
- ___ 2. Open Notepad.
 - ___ a. In the **Commands** list, double-click **Open Program/File**.
 - ___ b. In the **Step 1: Program/File Path** field, enter: `notepad.exe`

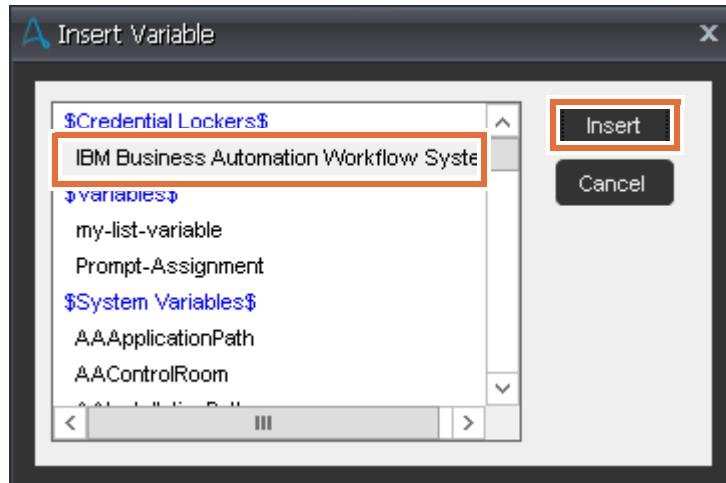


- ___ c. Click **Save**.
- ___ 3. Add a loop.
 - ___ a. In the **Commands** list, expand **Loop** and double-click **Times**.
 - ___ b. Click the **Times** field and press F2 (of Fn+F2).
 - ___ c. Double-click (not single-click) **vTotalAccts** and click **Save**.

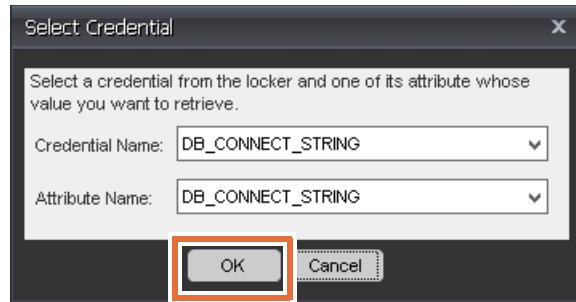
- ___ d. Double-click the **Comment** command that was inserted inside the loop and enter the following text:
Set value of account number to lookup.
- ___ 4. Add a Variable Operation command.
 - ___ a. In the **Commands** list, double-click **Variable Operation**.
 - ___ b. For **Specify Variable**, select **vAcctToLookup**.
 - ___ c. For **Select Operation**, make sure **Assign** is selected.
 - ___ d. In the **Specify value** field, enter: `$vCheckSum($Counter$,1)$`
 - ___ e. Click **Save**.

3.2. Connecting to a database

- ___ 1. Add a comment.
 - ___ a. In the **Commands** list, double-click **Comment** and enter the following text:
Connect to database and pull customer level for each account.
 - ___ b. Click **Save**.
- ___ 2. Connect to the database.
 - ___ a. In the **Commands** list, expand **Database** and double-click **Connect**.
 - ___ b. Place the cursor in the **Connection String** field and press F2 (or Fn+F2).
 - ___ c. In the Insert Variable window, in the **\$Credential Lockers\$** section, select **IBM Business Automation Workflow Systems**, and click **Insert**.



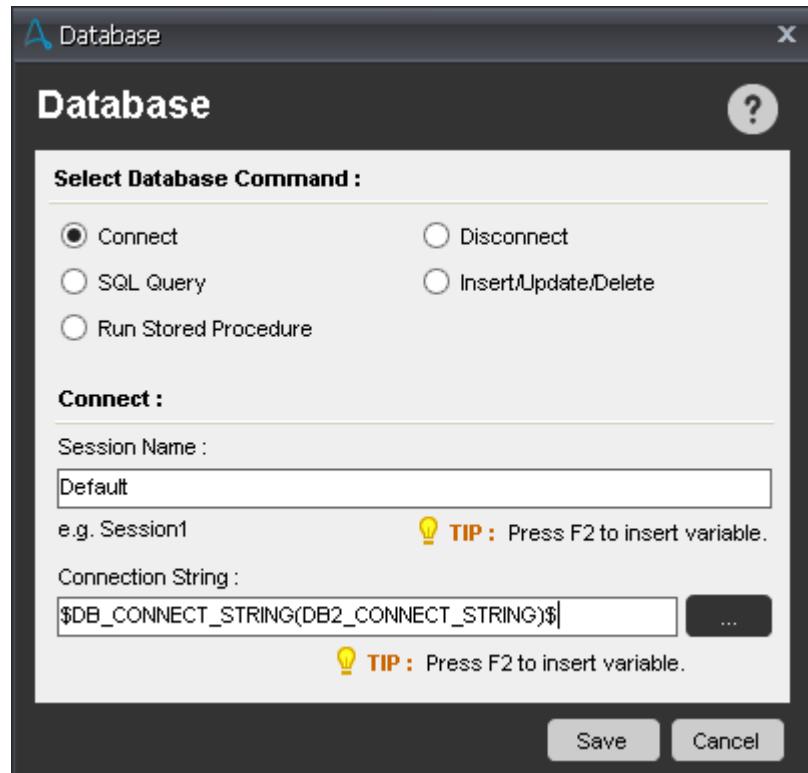
- ___ d. In the Select Credential window, select `DB_CONNECT_STRING` for both the **Credential Name** and **Attribute Name**, and click **OK**.



Information

If you don't see the **\$Credential Lockers\$** category, make sure that you are logged in to the Control Room. If not, you can log in to the Control Room and try this operation again.

- ___ e. Click **Save** to save the database connection command.



- ___ 3. Query the database.
 - ___ a. In the **Commands** list, expand **Database** and double-click **SQL Query**.
 - ___ b. In the **Enter SELECT Statement** field, enter the following SQL command:

```
SELECT * FROM DB2ADMIN.CUST_MEMBER_LEVEL WHERE ACCT_NUM =
'$vAcctToLookup$'
```

 - ___ c. Click **Save**.

**Important**

This command references a schema and table that is already defined.

4. Disconnect from the database.
 - a. In the **Commands** list, expand **Database** and double-click **Disconnect**.
 - b. Click **Save**.
5. Add a loop.
 - a. In the **Commands** list, expand **Loop** and double-click **Each Row In A SQL Query Dataset**.
 - b. Click **Save**.
6. Change the comment.
 - a. Double-click the added **Comment** and replace the text with this comment:

This only loops once for each 'parent' loop. This looping is used to extract \$Dataset Column(2)\$ data (customer level) and add to vCustLevel.
 - b. Click **Save**.
7. Add a Variable Operation command.
 - a. In the **Commands** list, double-click **Variable Operation**.
 - b. For **Specify Variable**, select **vCustLevel**.
 - c. For **Select Operation**, make sure **Assign** is selected.
 - d. In the **Specify value** field, enter: **\$Dataset Column(2)\$**
 - e. Click **Save**.
8. Add a comment after the first End Loop command.
 - a. Highlight the first **End Loop** command, which is before the final **End Loop** command.

```

25  ↗ Start Loop "Each row in SQL Data Set Session: Default"
26  ↗ Comment: This only loops once for each 'parent' loop. This looping is used to extract $Dataset Column(2)$ data (cu
27  ↗ Variable Operation: $Dataset Column(2)$ To $vCustLevel$
28  ↗ End Loop
29  ↗ End Loop

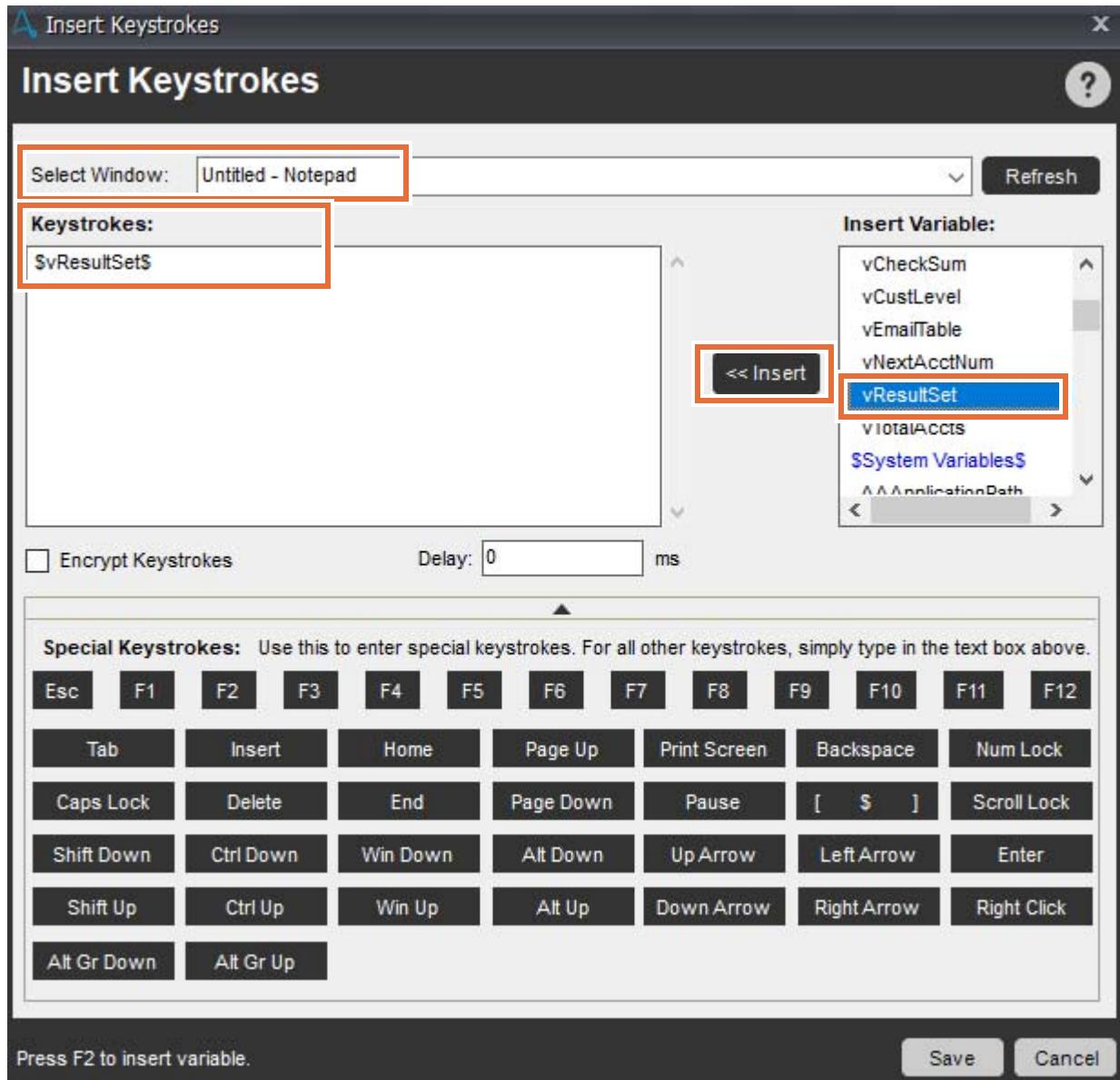
```

- b. In the **Commands** list, double-click the **Comment** command and enter the following text:

Append customer level to result set.
- c. Click **Save**.
9. Add a Variable Operation command
 - a. In the **Commands** list, double-click **Variable Operation**.

- ___ b. For **Specify Variable**, select **vResultSet**.
 - ___ c. For **Select Operation**, make sure **Assign** is selected.
 - ___ d. Place the cursor in the **Select Position** field and press F2 (or Fn+F2), select **Counter** (under **&System Variables\$**), and click **Insert**.
 - ___ e. In the **Specify value** field, enter: \$Counter\$ | \$vResultSet\$ | \$vCustLevel\$
 - ___ f. Click **Save**.
- ___ 10. Add a comment.
- ___ a. In the **Commands** list, double-click **Comment** and enter the following text:
Write out result set to Notepad.
 - ___ b. Click **Save**.
- ___ 11. Add an **Insert Keystrokes** command.
- ___ a. Open Notepad. You can use the Windows Run command (Windows key+R) and type notepad.
 - ___ b. In the **Commands** list, double-click **Insert Keystrokes**.
 - ___ c. From **Select Window**, select **Untitled – Notepad**.
 - ___ d. In the Insert Variable window, select **vResultSet** and click **Insert**.

___ e. Click **Save**.



___ 12. Add another **Insert Keystrokes** command.

- ___ a. In the **Commands** list, double-click **Insert Keystrokes**.
- ___ b. From **Select Window**, select **Untitled – Notepad**.

- ___ c. In the **Special Keystrokes** section, click **Enter**.



You see [ENTER] in the **Keystrokes** field.

- ___ d. Click **Save**.



Note

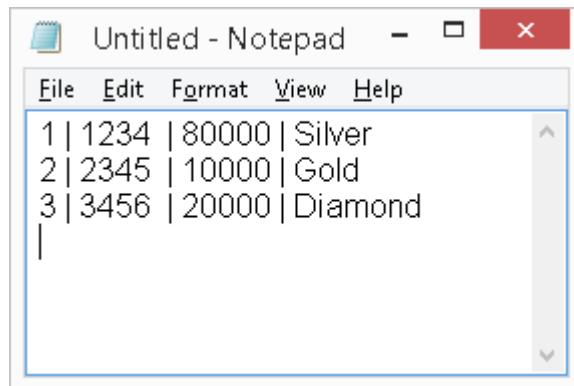
If you do not see **Untitled – Notepad** in the **Select Window** list, you can click **Refresh**.

- ___ 13. Save your bot.

3.3. Running the bot

- ___ 1. Click **Run** on the toolbar.

The bot pastes values to Notepad.

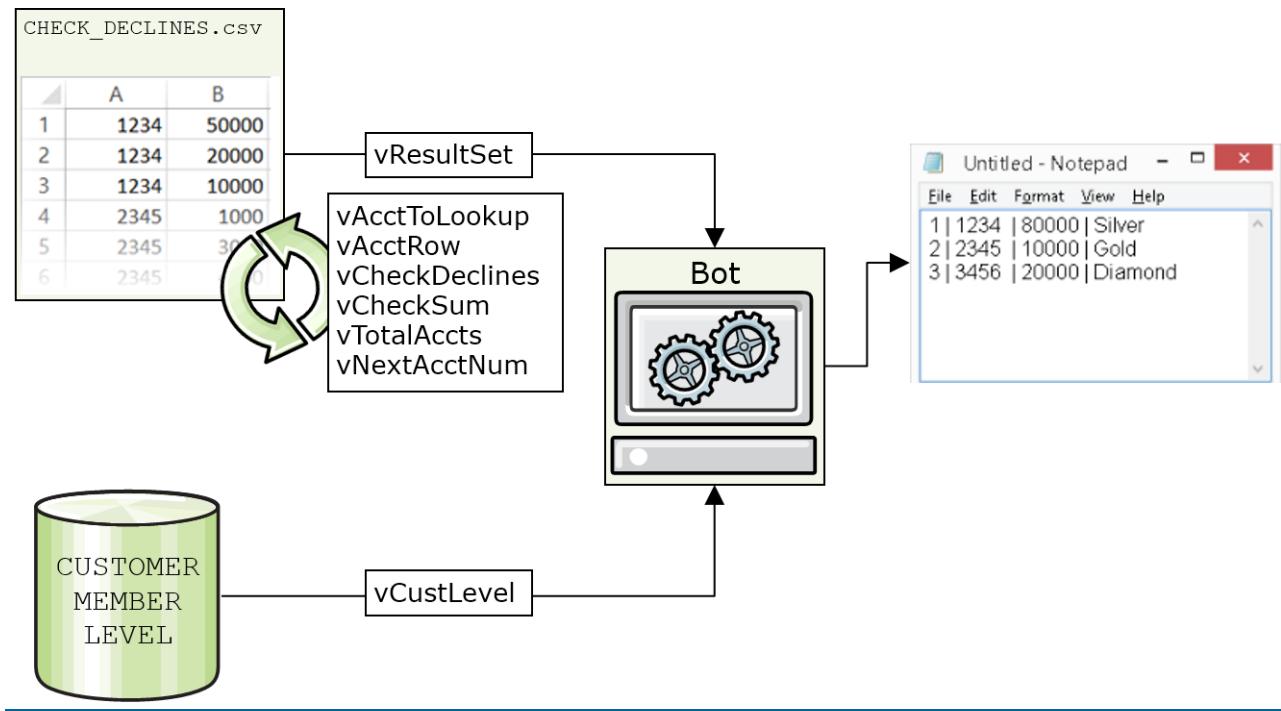


- ___ 2. Close Notepad without saving the changes.

Bot tasks:

- Pulls check decline values from the `CHECK_DECLINES.csv` file and calculates the sum for each account

- Pulls customer membership levels from the database for each account
- Combines the results and pastes them to Notepad



Section 4. Sending results by email

4.1. Copying text to the Clipboard

- ___ 1. Add a comment.
 - ___ a. Make sure that the final `End Loop` command is highlighted.
 - ___ b. In the **Commands** list, double-click **Comment** and enter the following text:
Paste results into Notepad and copy to email then email results to Margin Clerk.
 - ___ c. Click **Save**.
- ___ 2. Add an `Insert Keystrokes` command to copy the results to the Clipboard.
 - ___ a. In the **Commands** list, double-click **Insert Keystrokes**.
 - ___ b. In the **Select Window** list, choose **Untitled – Notepad**.
 - ___ c. Under Special Keystrokes, copy and paste the following text into the field:
`[CTRL DOWN]a[CTRL UP][CTRL DOWN]c[CTRL UP]`
 - ___ d. Click **Save**.
- ___ 3. Assign the contents of the Clipboard to `$vEmailTable$`.
 - ___ a. In the **Commands** list, expand **Clipboard** and double-click **Assign From Clipboard**.
 - ___ b. From the **Assign value of clipboard to a variable** list, select **vEmailTable**.
 - ___ c. Click **Save**.
- ___ 4. Close Notepad.
 - ___ a. In the **Commands** list, expand **Window Actions** and double-click **Close Window**.
 - ___ b. From the **Select Window** list, select **Untitled – Notepad**.
 - ___ c. Click **Save**.

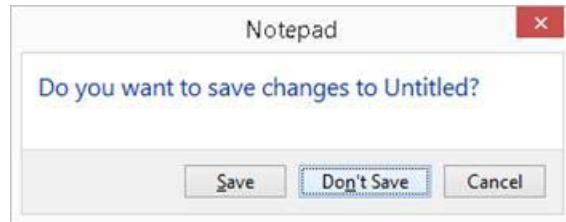
The bot must close Notepad. However, because the Notepad Window contains content, the bot must first respond to the Notepad dialog box that prompts the user to save the content.

4.2. Simulating “Don’t save” in Notepad

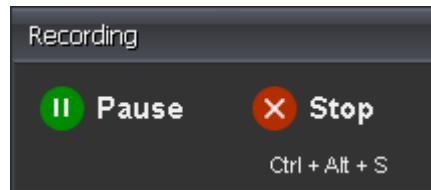
In this section, you simulate the response to the Notepad dialog box by using the Smart Recorder to retrieve the object properties of the **Don’t Save** button. The bot can then respond by clicking **Don’t Save**.

- ___ 1. If Notepad is not open, open it.
- ___ 2. Type a few characters in Notepad and then close Notepad, but do not respond to the Save dialog box.
- ___ 3. Return to the bot code and make sure that the last line is highlighted.
- ___ 4. On the toolbar, click **Record**.

- ___ 5. In the **Select Window** list, select **Notepad**.
- ___ 6. Click **Start**.
- ___ 7. In the Notepad Save window, click **Don't Save**.



- ___ 8. In the Recording window, click **Stop**.



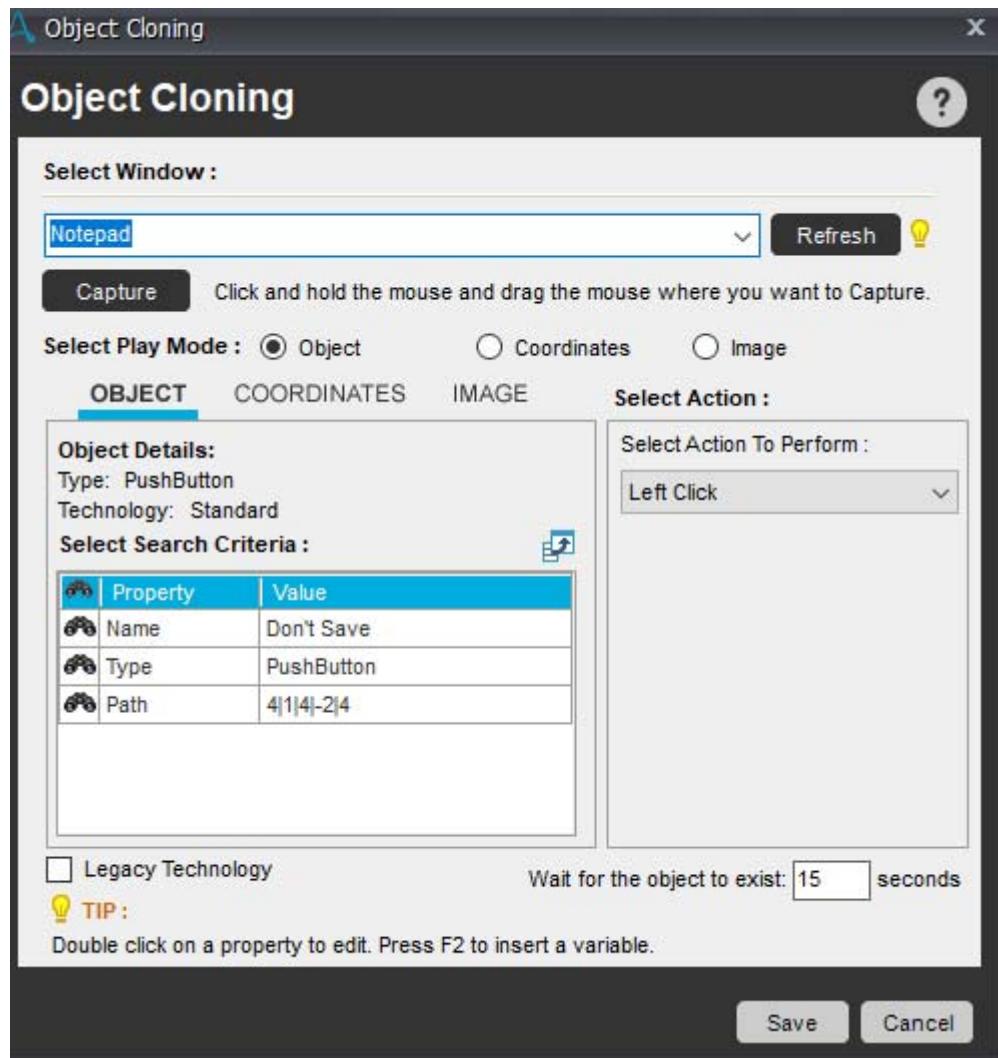
- ___ 9. If an Object Window opens, you can close it.



Information

The Recorder records every mouse and keyboard action you perform. If you perform any more actions than noted, you might end up with superfluous Object Cloning commands.

- ___ 10. Double-click the Object Cloning command that was placed in the bot code to verify it.



- ___ 11. Click **Cancel**.
 ___ 12. Close Notepad.

4.3. Sending email to Margin Clerk with results



Important

An outgoing email server configuration was configured for this exercise so that you can complete these steps.

- ___ 1. Double-click the **Send Email** command in the **Commands** list.

__ 2. Enter the following values:

- **From:** taskbot@sib.com
- **To:** marginClerk@sib.com
- **Subject:** Check Declines Summary
- **Message:** Select **HTML**, and enter the following text into the body of the email. You can copy and paste this text into the message body field.

Dear Margin Clerk,

The Check Declines bot finished processing all check declines for the day.

The results are:

`$vEmailTable$`

For reference purposes, the following levels are the thresholds for the three customer levels:

Silver: 5000

Gold: 10000

Diamond: 20000

__ 3. Click **Save**.



Information

If you need to edit the formatting, you can format the text by using the editing options.

Message : Text HTML 💡 TIP : Press F2 to insert variable.

Arial 3 **B** *I* U **C** **Ln** Insert Picture

Dear Margin Clerk,

The Check Declines Task Bot has finished processing all check declines for the day. Please see results below.

`$vEmailTable$`

For reference purposes, the following levels are the thresholds for

__ 4. Click **Save** on the toolbar to save the bot.

The following screen capture shows the bot code:

```

1  Comment: Pull check values from file and add to running total for each account then add to result set.
2  Read From CSV file: "C:\Labfiles\check_declines\CHECK_DECLINES.csv" Delimiter: "Comma" Header: "No" Trim Lead
3  Start Loop "Each row in a CSV/Text file of Session: Default"
4  Comment: Add check values to running total.
5  Variable Operation: $vCheckSum($vAcctRow$,2$) + $vCheckDeclines($Counter$,2$). To $vCheckSum($vAcctRow$
6  If $vCheckDeclines($Counter$,1$) Not Equal To (<>) $vCheckDeclines($vNextAcctNum$, 1$) Then
7      Comment: If this is the last check for the account: - Add final check value - Append to result set - Increment Accou
8      Variable Operation: $vCheckDeclines($Counter$,1$) To $vCheckSum($vAcctRow$, 1$)
9      Variable Operation: $vCheckSum($vAcctRow$,1$) | $vCheckSum($vAcctRow$,2$) To $vResultSet($vAcctRow$
10     Variable Operation: $vAcctRow$ + 1 To $vAcctRow$
11     Variable Operation: $vTotalAccts$ + 1 To $vTotalAccts$
12 End If
13 Comment: Increment account number.
14 Variable Operation: $vNextAcctNum$ + 1 To $vNextAcctNum$
15 End Loop
16 Comment: Pull customer level from database for each account.
17 Open: "notepad.exe"
18 Start Loop "$vTotalAccts$" Times
19 Comment: Set value of account number to lookup.
20 Variable Operation: $vCheckSum($Counter$,1$) To $vAcctToLookup$
21 Comment: Connect to DB and pull customer level for each account.
22 Connect to "$DB_CONNECT_STRING(DB_CONNECT_STRING)$" Session:'Default'
23 Execute SQL Statement: 'SELECT * FROM DB2ADMIN.CUST_MEMBER_LEVEL WHERE ACCT_NUM = '$vAcctTo
24 Disconnect from database Session:'Default'
25 Start Loop "Each row in SQL Data Set Session: Default"
26 Comment: This only loops once for each 'parent' loop. This looping is used to extract $Dataset Column(2)$ data (c
27 Variable Operation: $Dataset Column(2)$ To $vCustLevel$
28 End Loop
29 Comment: Append customer level to result set.
30 Variable Operation: $Counter$ | $vResultSet$ | $vCustLevel$ To $vResultSet($Counter$)$
31 Comment: Write out result set to Notepad.
32 Keystrokes: $vResultSet$ in "Untitled - Notepad"
33 Keystrokes: [ENTER] in "Untitled - Notepad"
34 End Loop
35 Comment: Paste results into Notepad and copy to email then email results to Margin Clerk.
36 Keystrokes: [CTRL DOWN]a[CTRL UP][CTRL DOWN]c[CTRL UP] in "Untitled - Notepad"
37 Assign value of Clipboard to variable "$vEmailTable$"
38 Close Window: "Untitled - Notepad"
39 Object Cloning: Click On PushButton "Don't Save" in window 'Notepad'; Click Type: Left Click; Source: Window; Play Ty
40 Send Email: Subject "Check Declines Summary "

```

Section 5. Configuring the trigger

To make the bot run on its own, you need to configure either a schedule or a trigger. If you want the bot to run regularly, you can use the schedule option. For example, you can configure a schedule so that the bot runs every weeknight at 4:05 PM after the file is received. You can use the trigger option for the bot to run after a specific event, for example, after a file is received in a particular folder. In this section, you configure a trigger.

For this task, you stop working in the Workbench and return to the main Enterprise Client window.

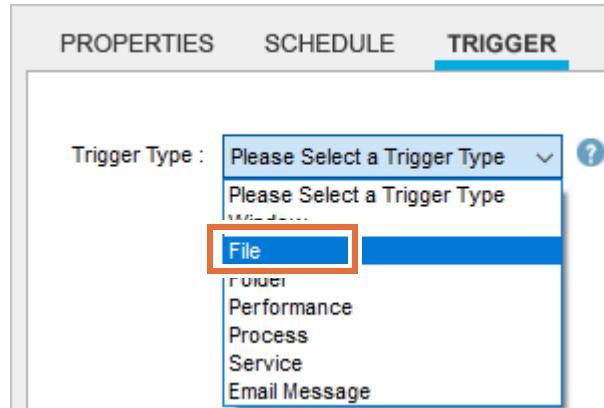
- ___ 1. Move the `CHECK_DECLINES.csv` file.
 - ___ a. Browse to the folder: `C:\labfiles\check_declines`
 - ___ b. Move the `CHECK_DECLINES.csv` file to a temporary location.
- ___ 2. Configure the trigger.
 - ___ a. Return to the main Enterprise Client window (not the Workbench).
 - ___ b. Select the **Check Declines.atmx** bot, and click the **TRIGGER** tab.

Note that your list of task files might be different from this screen capture.

The screenshot shows the Automation Anywhere Enterprise Client interface. The top navigation bar includes File, Edit, View, Tools, Help, and Feedback. Below the navigation is a toolbar with New, Record, Run, Edit, and ROI buttons. The main window is divided into sections: AUTOMATE, MANAGE, and DEFINE. The AUTOMATE section contains a sidebar with Tasks, My Tasks (expanded to show Sample Tasks, My Exes, My Scripts, My Docs), MetaBots, and a search bar. The MANAGE section has tabs for Properties, Schedule, and Trigger (the latter is highlighted with a red box). The DEFINE section is currently empty. The central area displays the 'My Tasks' list, which includes 'File Name' columns for various automation files: Account Opening.atmx, Check Declines.atmx (highlighted with a red box), helloworld.atmx, and Transactions.atmx. The Task Properties for 'Check Declines.atmx' are shown in the bottom right, with the 'Trigger' tab selected. The 'General' section contains the following details:

Repeat	File Name :	Check Declines.atmx
Speed	Created at :	04/26/2018 12:51:46
Notification	Status :	Complete
Hotkey	Last Run Time :	04/26/2018 16:26:03
Security	Priority (for queuing) :	Low
	Timeout (in minutes) :	0
<input type="checkbox"/> Enable this task to run with other similar files or windows		

- ___ c. For **Trigger Type**, select **File**.



- ___ d. In the **File Name** field, paste the following path:

C:\labfiles\check_declines\CHECK_DECLINES.csv

- ___ e. For **Action**, select **When file is created**.

- ___ f. Click **Save**.

A screenshot showing the trigger configuration dialog and a preview table. The dialog contains fields for 'Trigger Type' (set to 'File'), 'File Name' (set to 'C:\labfiles\check_declines\CHECK_DECLINES.csv'), and 'Action' (set to 'When file is created'). Below the dialog are 'Save' and 'Cancel' buttons. To the right, a preview table shows the configuration: 'Trigger Type' is 'File' and 'Action' is 'When 'C:\labfiles\check_declines\CHECK_DECLINES.csv' is created'. The table has a header row and two data rows.

Trigger Type	Action
File	When 'C:\labfiles\check_declines\CHECK_DECLINES.csv' is created

Section 6. Running the bot and confirming email delivery

After running the bot, you check that the email was sent with the correct information. In this section, you take on the role of the Margin Clerk to confirm the information that is sent by the bot.



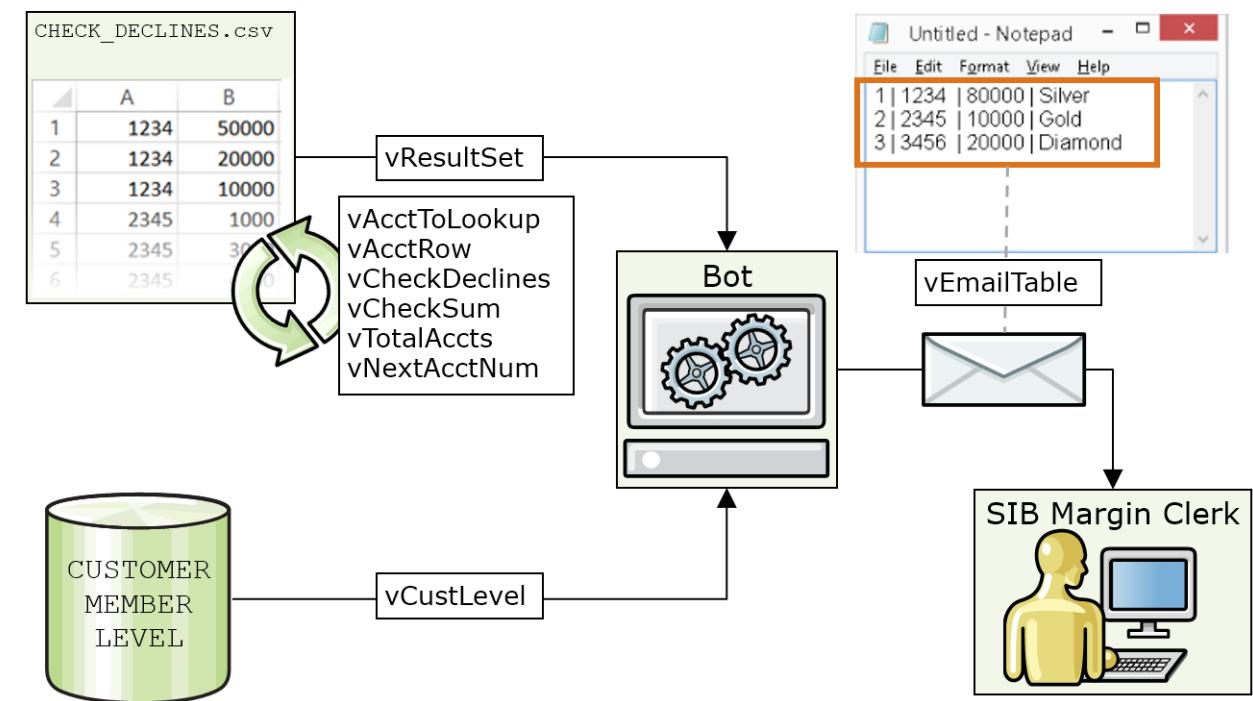
Information

When running bots interactively, refrain from making keystrokes or mouse movements during its execution.

- 1. Trigger the bot by moving the `CHECK_DECLINES.csv` file from its temporary location to the directory: `C:\labfiles\check_declines`
The bot starts running.

Bot tasks:

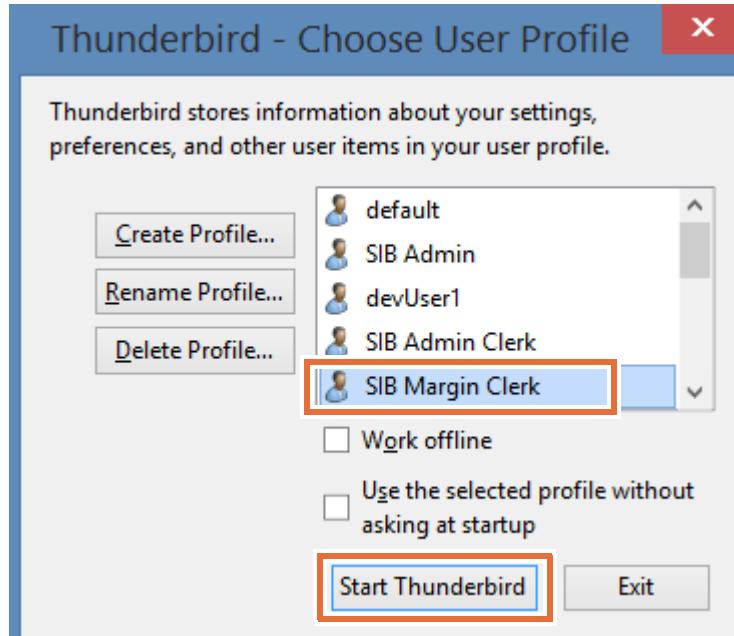
- Pulls check decline values from the `CHECK_DECLINES.csv` file and calculates the sum for each account
- Pulls customer membership levels from the database for each account
- Combines the results and pastes them to Notepad
- Copies the results from Notepad into a variable
- Emails the results to the Margin Clerk



- ___ 2. Sign in to the Mozilla Thunderbird email client as the Margin Clerk.
 - ___ a. On the desktop, double-click the **Mozilla Thunderbird** shortcut.



- ___ b. Select the SIB Margin Clerk and click **Start Thunderbird**



- ___ c. Sign in with these credentials:
 - **User name:** SIB Margin Clerk
 - **Password:** marginClerk
- ___ 3. Check for the email from the Check Declines bot.
 - ___ a. In the left pane, click **marginClerk@sib.com > Inbox** to open the Inbox.
 - ___ b. In the Inbox pane, click the **Check Declines Summary** email to open it.

You should see the following message:

Dear Margin Clerk,

The Check Declines Task Bot finished processing all check declines for the day.

The results are:

1		1234		80000		Silver
2		2345		10000		Gold
3		3456		20000		Diamond

For reference purposes, the following levels are the thresholds for the three customer levels:

Silver: 5000

Gold: 10000

Diamond: 20000

Section 7. *Optional:* Viewing the solution file

If you were unable to complete and run the bot successfully, you can open the solution file for this exercise to see how the bot should be coded.

- __ 1. Go to C:\labfiles\check_declines\solution, right-click the Check Declines solution.atmx file and click **Edit**.
The file opens in the Workbench.
- __ 2. To run the bot, click **Run** on the Workbench toolbar.



Information

If you want to load the solution file from the Enterprise Client, move the Transactions solution.atmx file to the following directory: C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks

The Enterprise Client **My Tasks** view automatically accesses files in that folder.

From the **My Tasks** list, you can either run the bot by double-clicking it or you can view it by right-clicking the task and clicking **Edit**.

End of exercise

Exercise review and wrap-up

In this exercise, you started by creating a task and creating a set of variables. You then configured the bot to read values from a CSV file, cross-referenced the extracted CSV values against data from a database, and keep a running sum of check totals. Finally, you configured a bot trigger, and verified the content of the email that the bot sends to the SIB margin clerk.

Exercise 6. Creating a bot to evaluate data from a PDF and send an email

Estimated time

02:00

Overview

In this exercise, you implement conditional logic and work with email triggers. You learn how to download attachments and extract data from a PDF and validate content.

Objectives

After completing this exercise, you should be able to:

- Automate the download of email attachments
- Extract data from a PDF by using custom form fields
- Implement conditional (If/Else) logic in a bot
- Configure an email trigger to run the bot

Introduction

In this exercise, you create a bot that saves PDF attachments from an email message, and extracts data from the PDF. You then configure the bot to evaluate the extracted data, and send an email based on conditional (If/Else) logic.

Scenario: Currently, SIB enters trades manually on behalf of MyBank because the two systems are not integrated yet. During the integration transition, MyBank forwards manual trades to a SIB trade reporter. When a trade receipt is received, the trade reporter verifies whether all required information is on the trade receipt. If all of the required information is available in the trade receipt, the trade reporter emails the supervisor to inform them that they received a trade.

The bot should monitor the email distribution list, verify that the required information is present, and send appropriate notifications.

As a bot developer, you must develop a bot that automates these tasks:

- Connect to the email server and monitor the distribution list for trade receipt email messages
- Download and save the trade receipt PDF attachment
- Extract the trade data from the PDF and verify that required fields are present

- Based on the condition of missing information, send appropriate email notifications
-

This exercise includes the following sections:

- [Section 1, "Creating a bot and adding custom variables"](#)
- [Section 2, "Extracting trade receipts from email and extracting values from PDF"](#)
- [Section 3, "Trimming variables and determining length"](#)
- [Section 4, "Evaluating data and sending email responses"](#)
- [Section 5, "Automating clean-up of emails and files"](#)
- [Section 6, "Configuring a trigger"](#)
- [Section 7, "Running the bot"](#)
- [Section 8, "Optional. Viewing the solution file"](#)

Requirements

This exercise requires the computer lab environment that was developed for this course.

Section 1. Creating a bot and adding custom variables

In this section, you create a task and define the custom variables that you use in the bot.

1.1. Creating a task

- __ 1. Go to the Enterprise Client.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

- __ 2. In the Enterprise Client, create a task with the Workbench, by clicking **New** on the toolbar and selecting **Workbench** when prompted for a tool option.

1.2. Creating the variables for the bot

First, you create the variables to contain the data that is extracted from the trade receipt PDF document. Then, you create variables to contain the length of each variable. The length variables are used to determine whether data is missing from the trade receipt.

- __ 1. In the Workbench, open the **Variable Manager**.

- __ 2. Create the `vParty1` variable.

- __ a. In the Variable Manager pane, click **Add**.
- __ b. In the **Name** field, type: `vParty1`
- __ c. Click **Save**.
- __ d. When prompted about the null value, click **Yes**.

- __ 3. Repeat [Step 2](#) to create the remaining data variables:

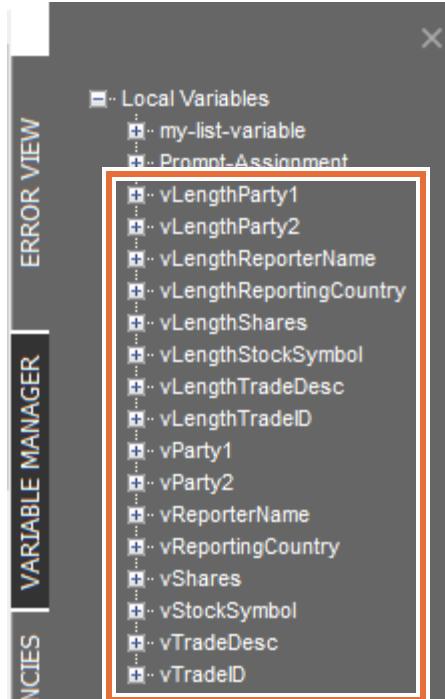
- `vParty2`
- `vReporterName`
- `vReportingCountry`
- `vShares`
- `vStockSymbol`
- `vTradeDesc`
- `vTradeID`

- __ 4. Repeat [Step 2](#) to create the length variables:

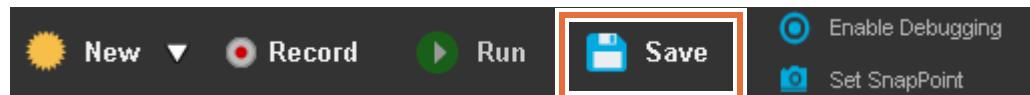
- `vLengthParty1`
- `vLengthParty2`
- `vLengthReporterName`
- `vLengthReportingCountry`
- `vLengthShares`
- `vLengthStockSymbol`
- `vLengthTradeDesc`
- `vLengthTradeID`

1.3. Saving your work

- 1. Review the list of variables in the Variable Manager to make sure that you added all 16 custom variables (prefixed with “v”).



- 2. Save your work by clicking **Save** on the toolbar.



- 3. In the **Filename** field, enter **Trade Booking** and click **Save**.



Reminder

Save your work often.

Section 2. Extracting trade receipts from email and extracting values from PDF

In this section, you configure a loop that iterates through emails in the `tradeList@sib.com` email account, and download PDF attachments from email messages that are in the account. You also configure the bot to read and extract data from the trade receipt PDF file.

2.1. Adding comments to your code

- ___ 1. In the **Commands** list, double-click the **Comment** command to enter this text.

=====

By using the equal sign (=) to separate your comments, your bot code is easier to read.

- ___ 2. Click **Save**.

- ___ 3. Add the following 3 comments by adding 3 more **Comment** commands:

- Pull check values from file, add them to running total for each account, then add to result set.
- =====
- Loop emails extracting attachments from the TradeReceiptsDL mailbox.

```

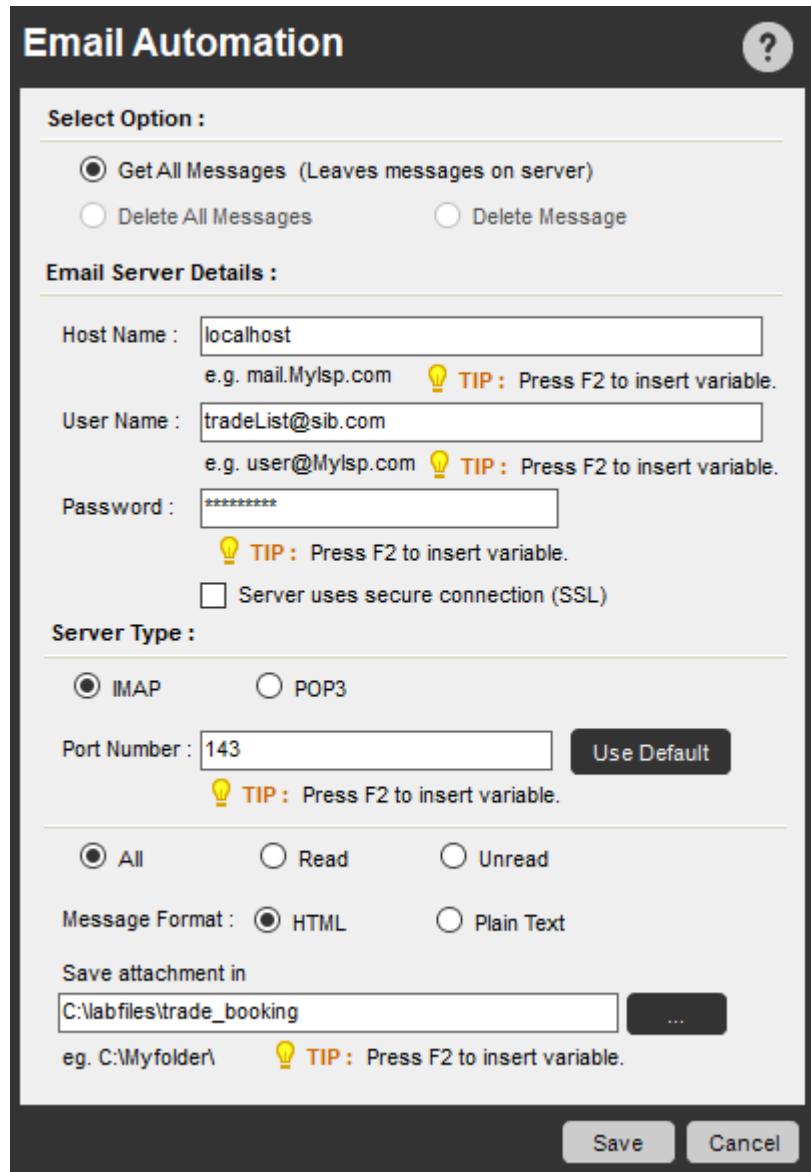
1  [green] Comment: =====
2  [green] Comment: Pull check values from file, add them to running total for each account, then add to result set.
3  [green] Comment: =====
4  [green] Comment: Loop emails extracting attachments from the TradeReceiptsDL mailbox.

```

2.2. Iterating through emails

- ___ 1. Add a **Loop** command to iterate through emails.
 - ___ a. In the **Commands** list, double-click the **Loop** command to see the list of loop commands and double-click **Each Email Message On Mail Server**.
 - ___ b. Enter the following values in the corresponding fields
 - **Host Name:** localhost
 - **User Name:** tradeList@sib.com
 - **Password:** tradeList
 - **Port Number:** 143
 - **Save attachment in:** C:\labfiles\trade_booking

___ c. Click **Save**.



- ___ 2. Update the **Comment** command and move it after the **End Loop** command.
- ___ a. Double-click the added **Comment** line and replace the text with the following text:
Extract custom-designed Form Fields from PDF.
- ___ b. Drag the **Comment** line and place it after the **End Loop** command.



Information

You can rearrange commands by selecting a line and dragging it. The command will be placed after the highlighted line in the Workbench.

For example, in this screen capture, the **Comment** line is dragged from line 6 in the code. The **End Loop** line is highlighted, so when the mouse is released, the **Comment** line is dropped at line 7, and the **End Loop** command moves to line 6.

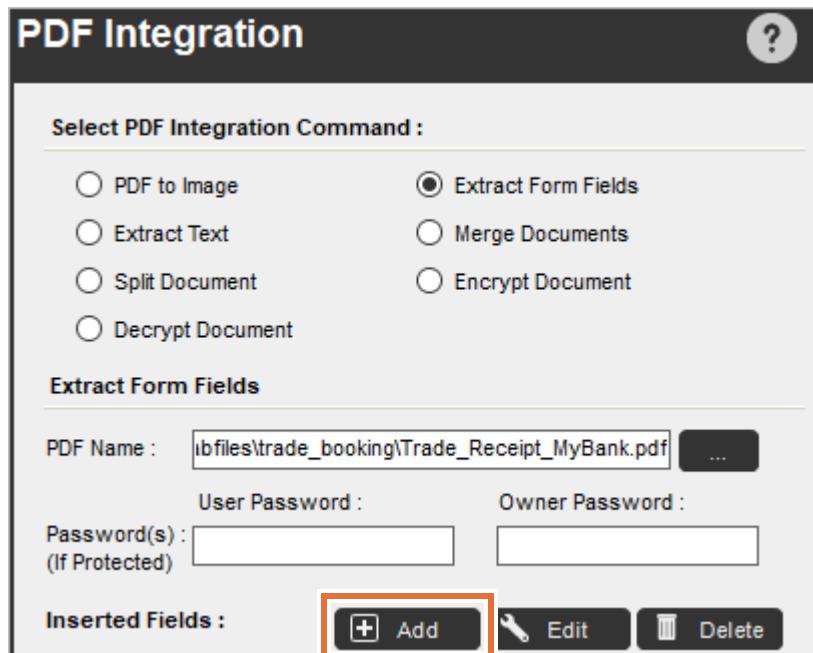
```

3 Comment: =====
4 Comment: Loop through emails extracting attachments from the TradeReceiptsDL mailbox.
5 Start Loop "Each message on server: localhost, User Name: tradeList@sib.com, ServerType: IMAP, Mes
6 Comment: Extract custom-designed Form Fields from PDF.
7 End Loop
8 Comment: Extract custom-designed Form Fields from PDF.

```

2.3. Extracting information from a PDF form

- ___ 1. Configure the **PDF Integration** command.
 - ___ a. In the **Commands** list, double-click the **PDF Integration** command to see the list of commands and double-click **Extract Form Fields**.
 - ___ b. In the **PDF Name** field, click the Ellipses (...), go to the C:\labfiles\trade_booking directory, and double-click the **Trade_Receipt_MyBank.pdf** file.
 - ___ c. In the **Inserted Fields** section, click **Add**.



- ___ 2. Define the custom form fields to be extracted from PDF by capturing each of the values for the fields in the Trade Information section.

The fields are:

- TRADE ID
- TRADE STOCK SYMBOL
- NUMBER OF SHARES
- REPORTING COUNTRY
- TRADE DESCRIPTION
- TRADE REPORTER NAME
- PARTY 1
- PARTY 2

TRADE INFORMATION	
TRADE ID	1234
TRADE STOCK SYMBOL	IBM
NUMBER OF SHARES	10
REPORTING COUNTRY	U.S.
TRADE DESCRIPTION	Please buy 10 shares
TRADE REPORTER NAME	Hans Wegman
PARTY 1	MvBank
PARTY 2	SIB



Information

To create a custom form field, use your mouse to outline the area on the page that you want to capture.

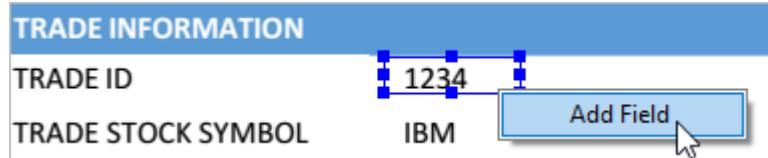
Using your mouse, drag an outline around an area that is large enough to accommodate the largest value possible for that field. The area that you outline determines the number of characters that can be read.

For this training, the bot works even when some data is cut off.

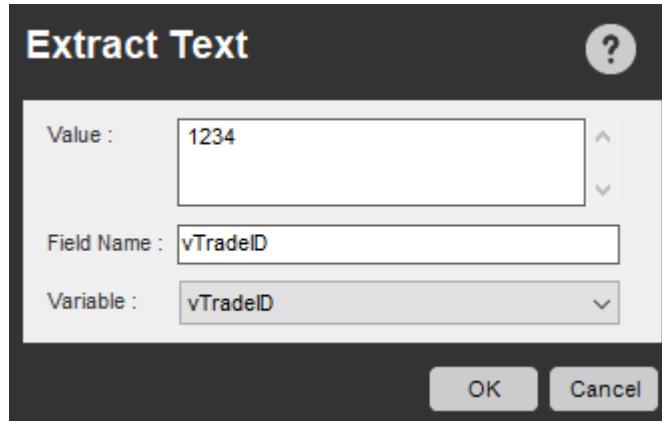
- ___ a. For the **TRADE ID** field, place your mouse to the upper left of **1234** and drag the outline around **1234** so that it is wide enough to accommodate some additional characters.



- ___ b. After the outline is created, right-click the highlighted value and click **Add Field**.



- ___ c. In the **Field Name** field of the **Extract Text** dialog box, enter: **vTradeID**
 ___ d. In the **Variable** list, choose **vTradeID**, and then click **OK**.



The extracted fields appear in the **Inserted Fields** list.

Field Name	Field Type	Field Value	Select Var
vTradeID	Custom	1234	vTradeID



Hint

When outlining custom form fields for PDF integration, keep in mind these limitations:

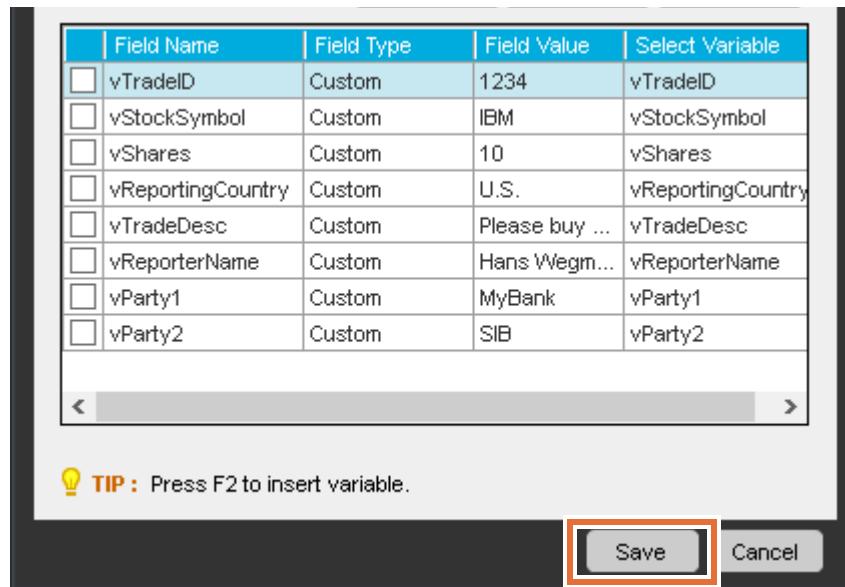
- You can outline only one value at a time.
- The outline does not display while moving or dragging it.
- If you need to adjust the outline of a field, you can hover your mouse over the outline and either move it or resize it.
- After you add a field to the **Inserted Fields** list, you can reuse the outline for your next field by hovering your mouse on it and moving or resizing it.
- You can change a field name after you add it to the **Inserted Fields** list. Select the field in the list and click **Edit**. However, if you need to change the outline, you can delete the field and add it again.

-
- 3. Repeat [Step 2](#) using the following information for each custom form field.

Stock Certificate Field	Field Name / Variable
TRADE STOCK SYMBOL	vStockSymbol
NUMBER OF SHARES	vShares
REPORTING COUNTRY	vReportingCountry
TRADE DESCRIPTION	vTradeDesc
TRADE REPORTER NAME	vReporterName
PARTY 1	vParty1
PARTY 2	vParty2

- 4. After you outline and define all fields, click **Save**.

- 5. Check that the PDF Integration dialog box correctly lists all the inserted fields, and click **Save**.



- 6. Save your work.

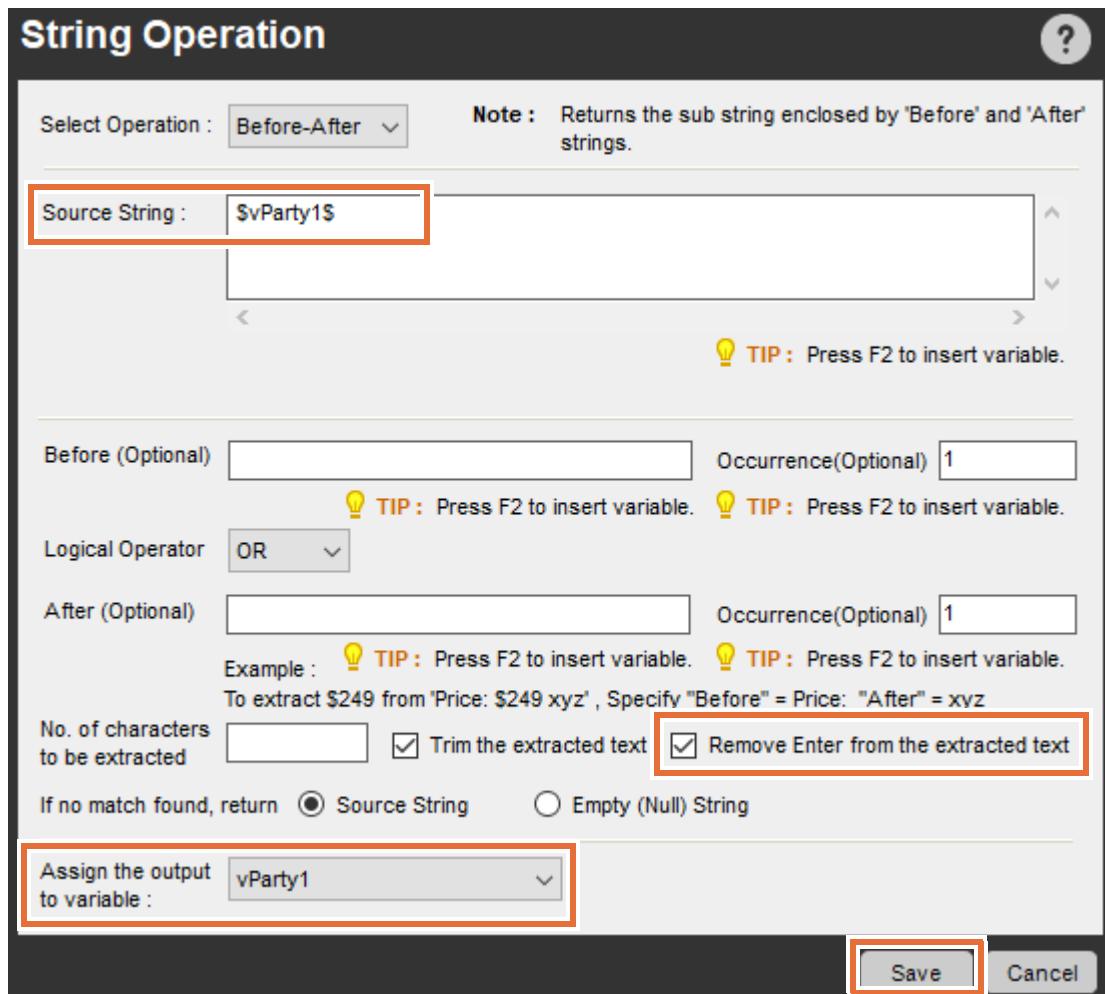
Section 3. Trimming variables and determining length

When extracting values from the PDF, the Enter key is included as part of the value. In this section, you use the **String Operation** command to remove the Enter key from the extracted text. After the variables are trimmed, the next step is to determine how many characters each variable contains.

3.1. Trimming variables

- ___ 1. Add the following 4 comments by adding 4 Comment commands.
 - =====
 - Trim all variables and determine length.
 - =====
 - Trim all variables.
- ___ 2. In the **Commands** list, double-click **String Operation** to see the list of commands and double-click **Before-After**.
- ___ 3. Define the String Operation command to trim the vParty1 variable.
 - ___ a. In the **Source String** field, place the cursor and press F2 (or Fn+F2) to access the variable list.
 - ___ b. In the Insert Variable dialog box, select **vParty1** and click **Insert** to close the dialog box.
 - ___ c. In the lower part of the String Operation dialog box, select the **Remove Enter from the extracted text** check box.
 - ___ d. From the **Assign the output to variable** list, select **vParty1**.

- __ e. Click Save.



- __ 4. Repeat [Step 2](#) and [Step 3](#) for each variable to be extracted from the PDF.

- vParty2
- vReporterName
- vReportingCountry
- vShares
- vStockSymbol
- vTradeDesc
- vTradeID

3.2. Determining the length of each variable

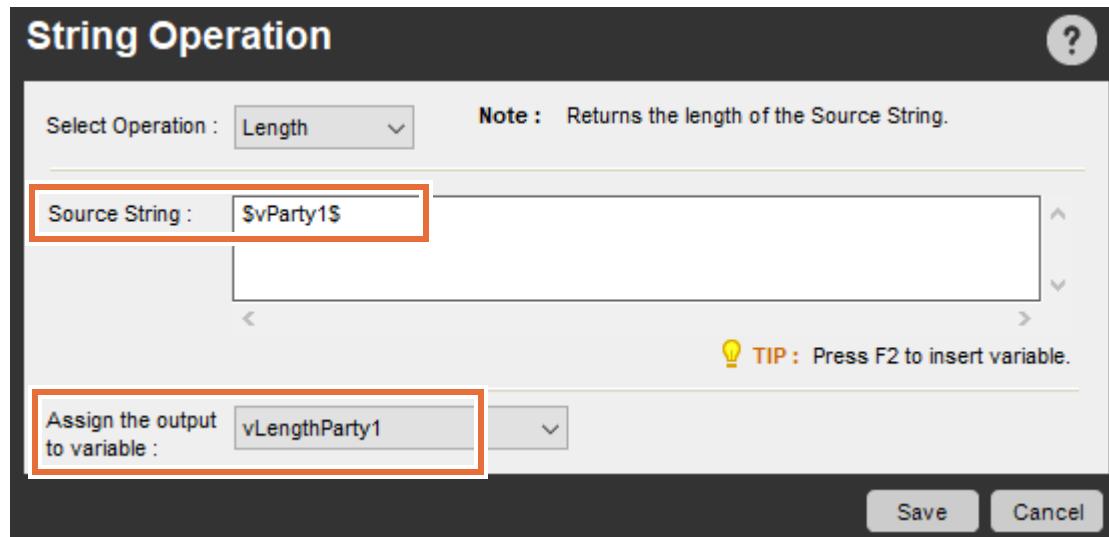
In this section, you evaluate the length of the value of each variable. The length variables are used to determine whether data is missing from the trade receipt.

- __ 1. Add a comment line with the following text:

Get length of variables.

- __ 2. In the **Commands** list, double-click **String Operation**, and then double-click **Length**.

- ___ 3. Define the String Operation command to determine the length of the vParty1 variable
- In the **Source String** field, place the cursor and press F2 (or Fn+F2) to access the variable list.
 - In the Insert Variable dialog box, select **vParty1** and click **Insert** to close the dialog box.
 - In the **Assign the output to variable** field, select **vLengthParty1**.
 - Click **Save**.



- ___ 4. Repeat these steps for each variable in which length is evaluated.

Source String	Assign the output to variable:
\$vParty2\$	vLengthParty2
\$vReporterName\$	vLengthReporterName
\$vReportingCountry\$	vLengthReportingCountry
\$vShares\$	vLengthShares
\$vStockSymbol\$	vLengthStockSymbol
\$vTradeDesc\$	vLengthTradeDesc
\$vTradeID\$	vLengthTradeID

- ___ 5. Save your work.

The following screen capture shows the bot code.

```

1  Comment: =====
2  Comment: Pull Trade Receipts from email, save to local folder, and extract values from PDF.
3  Comment: =====
4  Comment: Loop through emails extracting attachments from the TradeReceiptsDL mailbox.
5  Start Loop "Each message on server: localhost, User Name: tradeList@sib.com, ServerType: IMAP, Message Format: HTML"
6  End Loop
7  Comment: Extract custom-designed Form Fields from PDF.
8  PDF Integration: Extract Form Fields from "C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf"
9  Comment: =====
10 Comment: Trim all variables and determine length.
11 Comment: =====
12 Comment: Trim all variables.
13 String Operation: Before-After "$vParty1$" and assign output to $vParty1$
14 String Operation: Before-After "$vParty2$" and assign output to $vParty2$
15 String Operation: Before-After "$vReporterName$" and assign output to $vReporterName$
16 String Operation: Before-After "$vReportingCountry$" and assign output to $vReportingCountry$
17 String Operation: Before-After "$vShares$" and assign output to $vShares$
18 String Operation: Before-After "$vStockSymbol$" and assign output to $vStockSymbol$
19 String Operation: Before-After "$vTradeDesc$" and assign output to $vTradeDesc$
20 String Operation: Before-After "$vTradeID$" and assign output to $vTradeID$
21 Comment: Get length of variables.
22 String Operation: Get length of "$vParty1$" and assign output to $vLengthParty1$
23 String Operation: Get length of "$vParty2$" and assign output to $vLengthParty2$
24 String Operation: Get length of "$vReporterName$" and assign output to $vLengthReporterName$
25 String Operation: Get length of "$vReportingCountry$" and assign output to $vLengthReportingCountry$
26 String Operation: Get length of "$vShares$" and assign output to $vLengthShares$
27 String Operation: Get length of "$vStockSymbol$" and assign output to $vLengthStockSymbol$
28 String Operation: Get length of "$vTradeDesc$" and assign output to $vLengthTradeDesc$
29 String Operation: Get length of "$vTradeID$" and assign output to $vLengthTradeID$
```

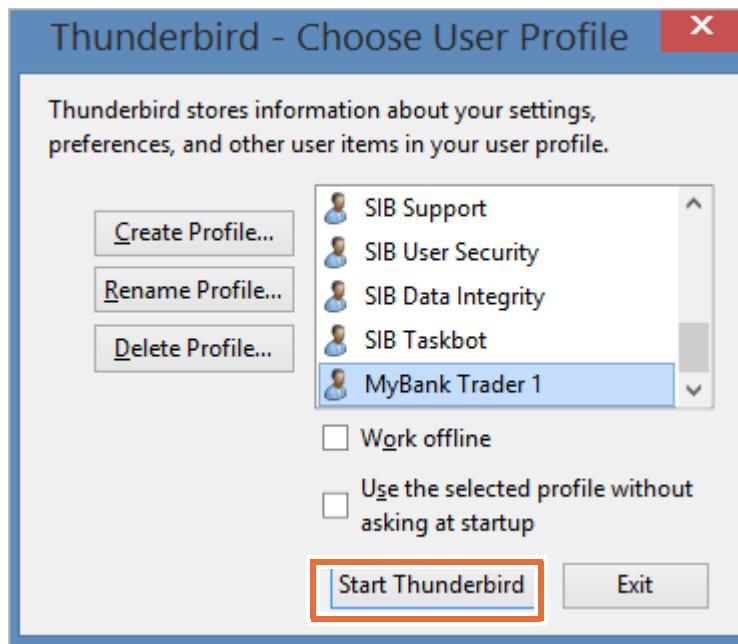
3.3. Preparing your environment to test the bot

In this section, you run the bot to verify that it behaves correctly. First, you prepare your environment for this test by deleting the current `Trade_Receipt_MyBank.pdf` file. You then send an email with a new trade receipt, which should start the trigger to run the bot.

- ___ 1. In Windows Explorer, go to the `C:\labfiles\trade_booking` directory and delete the `Trade_Receipt_MyBank.pdf` file.
- ___ 2. Open the Thunderbird email client as MyBank Trader 1.
 - ___ a. On the desktop, double-click the **Mozilla Thunderbird** shortcut.

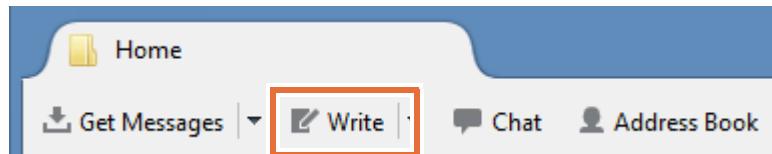


- __ b. Select the **MyBank Trader 1** user name, click **Start Thunderbird**.

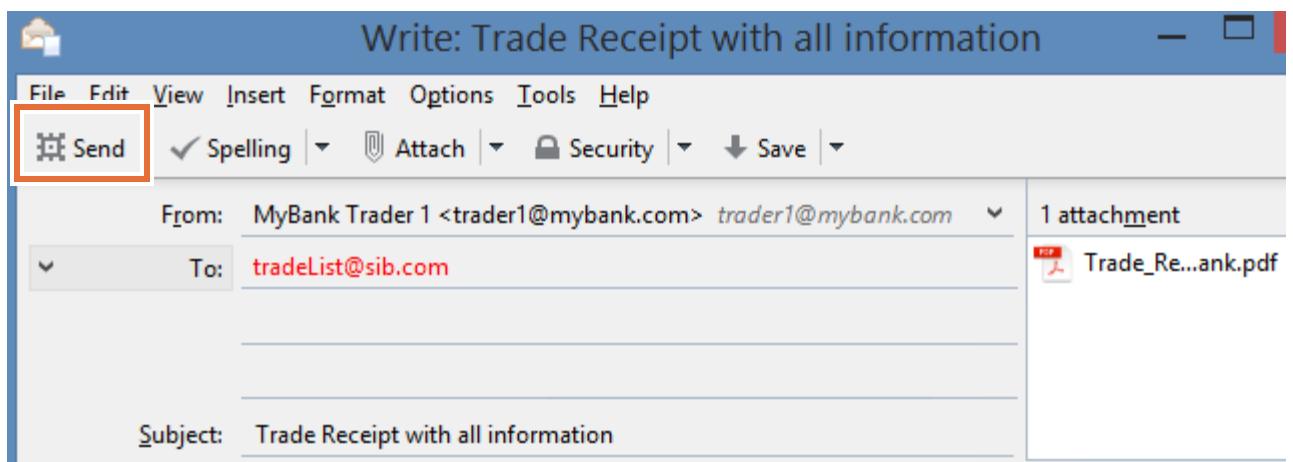


- __ c. When prompted for a password, enter: trader1
- __ 3. Send an email with the Trade Receipt to the `tradeList@sib.com` distribution list from MyBank Trader 1.

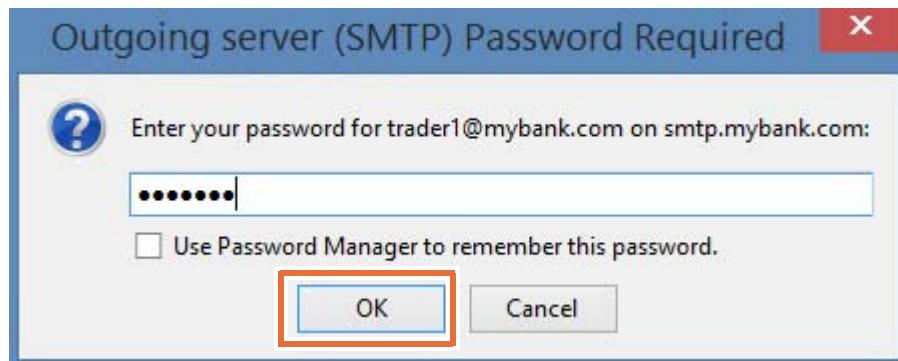
- __ a. On the Thunderbird toolbar, click **Write**.



- __ b. In the **To** field, enter: `tradeList@sib.com`
- __ c. In the **Subject** field, enter: Trade Receipt with all information
- __ d. On the email toolbar, click **Attach**, and go to the `C:\labfiles\trade_booking\trade_receipts_scenarios\all_information\Trade_Receipt_MyBank.pdf` file, and then click **Open**.
- __ e. On the email toolbar, click **Send**.



- ___ f. When prompted for a password, enter `trader1` and click **OK**.

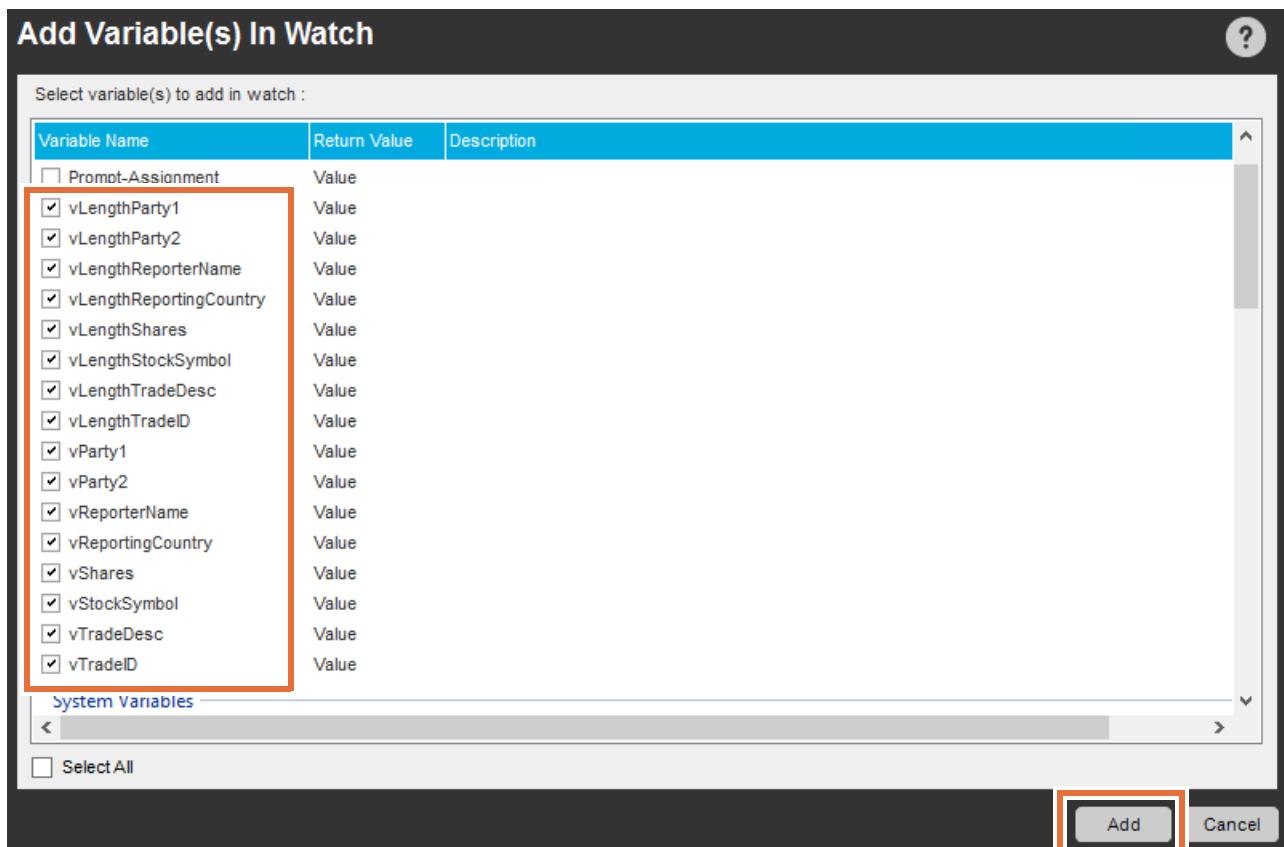


- ___ 4. Close **Thunderbird** and return to the **Workbench** window.

3.4. Configuring the Workbench for debugging

In this section, you enable debugging mode to see whether the variables populate correctly during the bot test.

- ___ 1. In the Workbench window, click **Enable Debugging** on the toolbar.
- ___ 2. Add variables to the **Variable Watch Table**.
 - ___ a. In the Variable Watch Table, click **Add**.
 - ___ b. Select all the custom variables (prefixed with "v") and click **Add**.



- ___ 3. On the toolbar, click **Run**.



Troubleshooting

When you run bots in debug mode, do not use the mouse or keyboard during execution.

4. Verify that the values that are assigned in the Variable Watch Table match the following values:

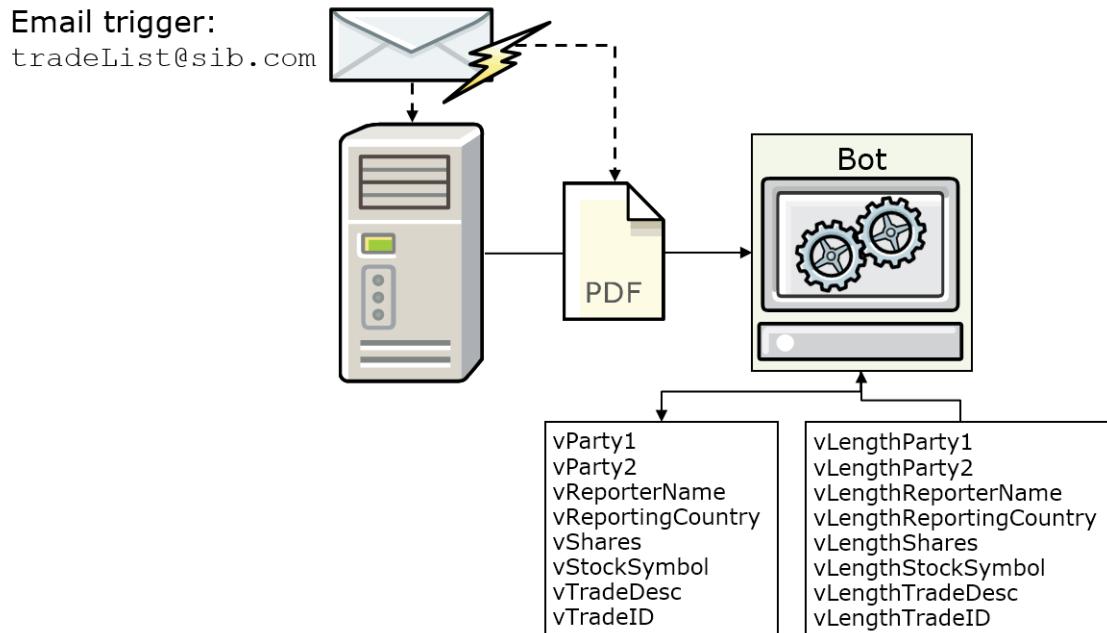
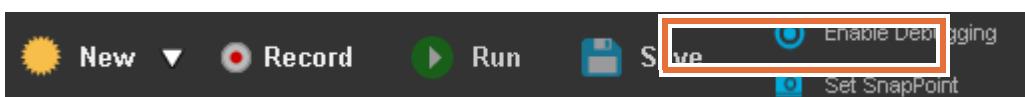
Variable	Value
vParty1	MyBank
vParty2	SIB
vReporterName	Hans Wegman
vReportingCountry	U.S.
vShares	10
vStockSymbol	IBM
vTradeDesc	Please buy 10 shares
vTradeID	1234
vLengthParty1	6
vLengthParty2	3
vLengthReporterName	11
vLengthReportingCountry	4
vLengthShares	2
vLengthStockSymbol	3
vLengthTradeDesc	20
vLengthTradeID	4

Variable(s) Watch Table	
Name	Value
vLengthParty1	6
vLengthParty2	3
vLengthReporterName	11
vLengthReportingCountry	4
vLengthShares	2
vLengthStockSymbol	3
vLengthTradeDesc	20
vLengthTradeID	4
vParty1	MyBank
vParty2	SIB
vReporterName	Hans Weiman
vReportingCountry	U.S.
vShares	10
vStockSymbol	IBM
vTradeDesc	Please buy 10 shares
vTradeID	1234

Select All

Bot actions:

- Connect to the email server and monitor the trade distribution list for trade receipt messages
- Download trade receipt PDF attachments, and extract all data from the trade receipt PDF
- Determine the length of each variable

5. Click **Disable Debugging**.

Section 4. Evaluating data and sending email responses

In this section, you configure the bot to evaluate the data. Based on what the bot finds, two conditions are possible:

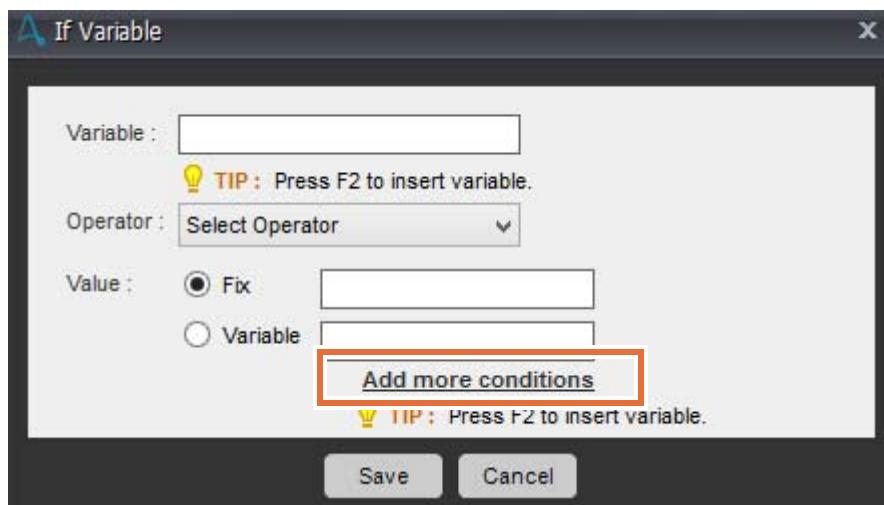
1. The bot determines that the trade receipt is missing data.
2. The bot determines that the trade receipt is not missing data.

In either case, the bot sends an email. You configure the bot to send emails to different recipients depending on the condition:

1. If the bot finds that the trade receipt is missing data, it sends an email to the trade reporter to alert them that the trade data is incomplete.
2. If the bot finds that the trade receipt is complete, it sends an email to the trade supervisor to notify them that the trade was received.

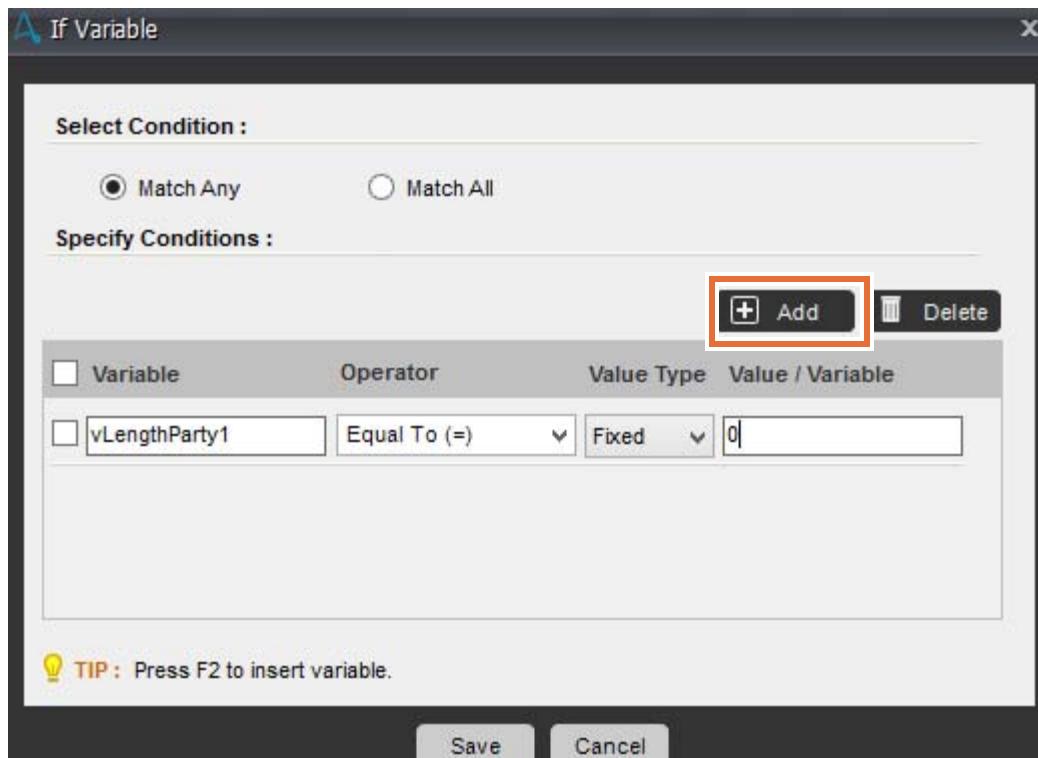
4.1. Evaluating the length of variables

- ___ 1. Add the following 4 comments by adding 4 **Comment** commands.
 - =====
 - Evaluate data and send email based on one of two situations.
 - =====
 - Evaluate length of each variable.
- ___ 2. In the **Commands** list, double-click **If/Else** and double-click **Variable**.
- ___ 3. In the If dialog box, for the **If Condition** field, click **Edit**.
- ___ 4. In the If Variable dialog box, click the **Add more conditions** link.



- ___ 5. In the If Variable dialog box, define conditions to test each of the length variables.
 - ___ a. Place the cursor in the **Variable** field, and press F2.
 - ___ b. In the Insert Variable list, choose **vLengthParty1** and click **Insert**.
 - ___ c. In the Operator list, choose **Equal To (=)**.

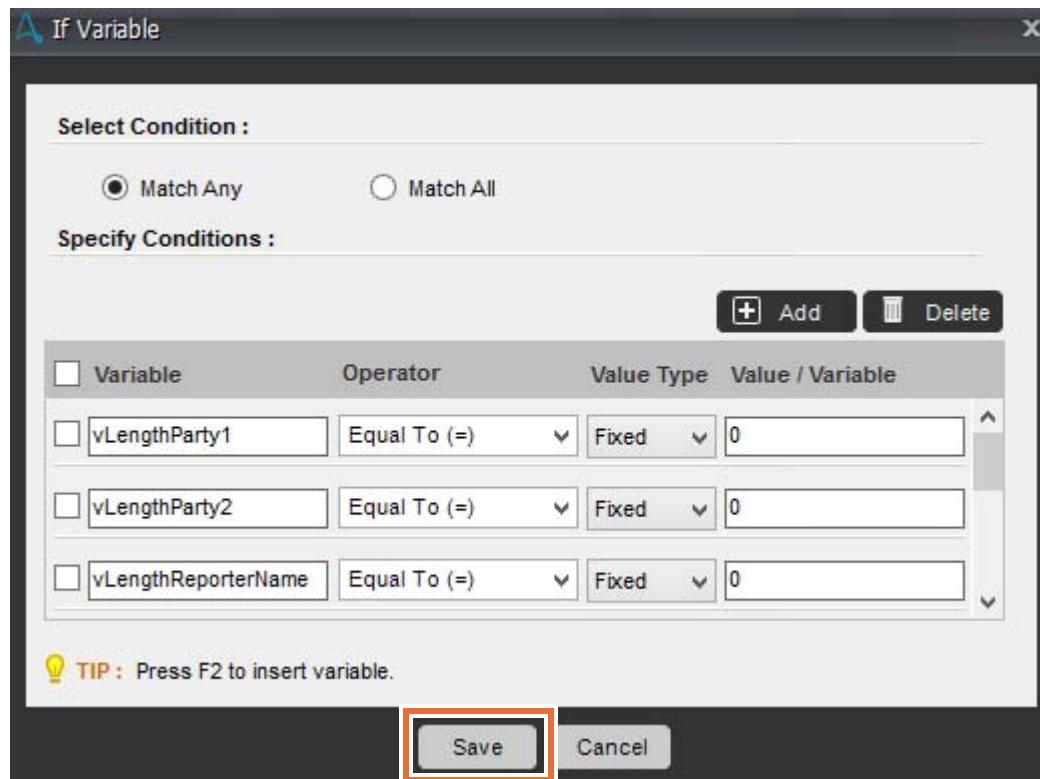
- ___ d. Leave **Value Type** as **Fixed**.
- ___ e. Place the cursor in the **Value / Variable** field and type 0.
- ___ f. Click **Add**.



- ___ 6. Repeat [Step 5](#) for each of the following variables:

- vLengthParty2
- vLengthReporterName
- vLengthReportingCountry
- vLengthShares
- vLengthStockSymbol
- vLengthTradeDesc
- vLengthTradeID

- ___ 7. Click **Save**.



4.2. Creating an email response for missing data

In this section, you configure the bot to respond to the first condition: data is missing from the trade receipt.

- ___ 1. Edit the comment to describe the email response when the **If** test shows missing data.
 - ___ a. Double-click the **Comment** command that is placed after the **If/Else** command.
 - ___ b. Enter this text:
Scenario: Missing Data and click **Save** (replacing the default text).
- ___ 2. In the **Commands** list, double-click **Send Email**.

3. In the **Select Email Parameters** section, enter the following values:

- **From:** taskbot@sib.com
- **To:** tradeReporter@sib.com
- **Subject:** Trade: \$vTradeID\$ Details
- **Attachment:** C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf
- **Message:** Select **HTML** and enter the following text for the message body. Note that you can copy the contents and paste this message into the body of the email.

Dear Trade Reporter,

The taskbot finished evaluating data for Trade: \$vTradeID\$ and determined that all required fields are **NOT** present.

The Trade details are as follows:

Trade Stock Symbol: \$vStockSymbol\$

Number of Shares: \$vShares\$

Trade Reporter Name: \$vReporterName\$

Reporting Country: \$vReportingCountry\$

Party1: \$vParty1\$

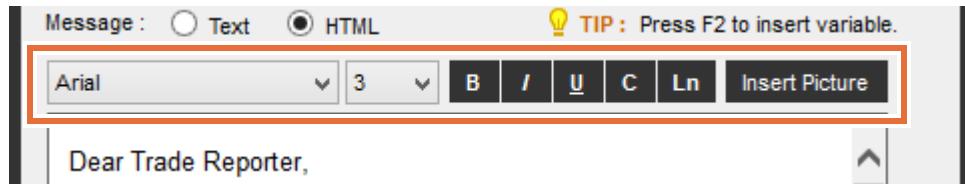
Party2: \$vParty2\$

Trade Description: \$vTradeDesc\$



Information

If you lose some text formatting when you copy and paste to the **Message** field, you can use the text editing tools to fix the format.



Or, you can select the **Text** option to avoid formatting issues.



4. Click **Save**.

4.3. Creating an email response when all the data is present

In this section, you configure the bot to handle the second condition: all data is included in the trade receipt.

- ___ 1. In the **Commands** list, drag the **Else** command from the **If/Else** list to the **Send Email** command in the Workbench.
- ___ 2. Add a comment.
 - ___ a. Double-click the **Comment** command and enter the following text:
Scenario: All Data Present
 - ___ b. Click **Save**.
- ___ 3. In the **Commands** list, double-click the **Send Email** command and enter the following values for the email parameters:
 - **From:** taskbot@sib.com
 - **To:** tradeSupervisor@sib.com
 - **Subject:** Trade: \$vTradeID\$ Details
 - **Attachment:** C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf
 - **Message:** Select **HTML** and enter the following text for the message body. You can copy the contents and paste this message into the body of the email.
Dear Trade Reporter Supervisor,

The task bot finished evaluating the data for Trade: **\$vTradeID\$** and determined that all required fields are present.

The trade details are as follows:

```
Trade Stock Symbol: $vStockSymbol$  

Number of Shares: $vShares$  

Trade Reporter Name: $vReporterName$  

Reporting Country: $vReportingCountry$  

Party1: $vParty1$  

Party2: $vParty2$  

-----  

Trade Description: $vTradeDesc$
```

- ___ 4. Click **Save**.

Section 5. Automating clean-up of emails and files

In this section, you add the final commands to the bot. These commands delete email messages from the tradeList@sib.com email account, and delete the downloaded PDF file from the C:\labfiles\trade_booking folder.

- ___ 1. Make sure that the **End If** statement is highlighted and add the following comments:
 - =====
 - Cleanup: Delete all emails and files in folder.
 - =====
- ___ 2. Automate the deletion of all emails that are in the tradeList@sib.com account inbox.
 - ___ a. In the **Commands** list, double-click **Email Automation** and double-click **Delete All Messages**.
 - ___ b. In the Email Automation dialog box, enter the following values:
 - **Host Name:** localhost
 - **User Name:** tradeList@sib.com
 - **Password:** tradeList
 - **Server Type:** IMAP
 - **Port Number:** 143
 - ___ c. Click **Save**.
- ___ 3. Automate the deletion of the trade receipt file.
 - ___ a. In the **Commands** list, double-click **Files/Folders** and double-click **Delete Files**.
 - ___ b. In the **File(s) Name** field of the File/Folders dialog box, enter:
C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf
 - ___ c. Click **Save**.
- ___ 4. Save your work.

The following screen capture shows the completed bot code.

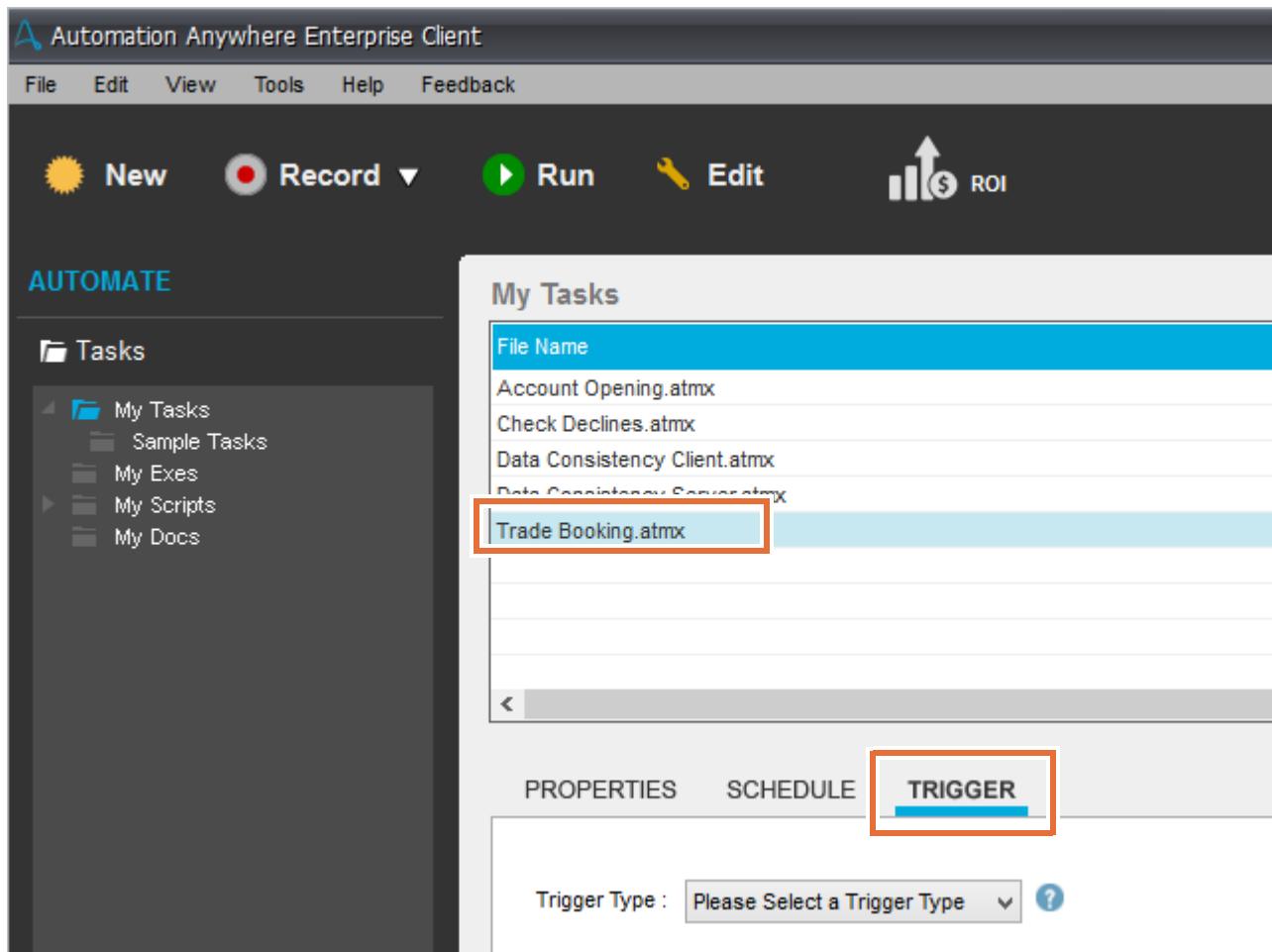
```

1  Comment: =====
2  Comment: Pull check values from file, add them to running total for each account, then add to result set.
3  Comment: =====
4  Comment: Loop emails extracting attachments from the TradeReceiptsDL mailbox.
5  ↗ Start Loop "Each message on server: localhost, User Name: tradeList@sib.com, ServerType: IMAP, Message
6  ✖ End Loop
7  Comment: Extract custom-designed Form Fields from PDF.
8  PDF Integration: Extract Form Fields from "C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf"
9  Comment: =====
10 Comment: Trim all variables and determine length.
11 Comment: =====
12 Comment: Trim all variables.
13 ⏚ String Operation: Before-After "$vParty1$" and assign output to $vParty1$#
14 ⏚ String Operation: Before-After "$vParty2$" and assign output to $vParty2$#
15 ⏚ String Operation: Before-After "$vReporterName$" and assign output to $vReporterName$#
16 ⏚ String Operation: Before-After "$vReportingCountry$" and assign output to $vReportingCountry$#
17 ⏚ String Operation: Before-After "$vShares$" and assign output to $vShares$#
18 ⏚ String Operation: Before-After "$vStockSymbol$" and assign output to $vStockSymbol$#
19 ⏚ String Operation: Before-After "$vTradeDesc$" and assign output to $vTradeDesc$#
20 ⏚ String Operation: Before-After "$vTradeID$" and assign output to $vTradeID$#
21 Comment: Get length of variables.
22 ⏚ String Operation: Get length of "$vParty1$" and assign output to $vLengthParty1$#
23 ⏚ String Operation: Get length of "$vParty2$" and assign output to $vLengthParty2$#
24 ⏚ String Operation: Get length of "$vReporterName$" and assign output to $vLengthReporterName$#
25 ⏚ String Operation: Get length of "$vReportingCountry$" and assign output to $vLengthReportingCountry$#
26 ⏚ String Operation: Get length of "$vShares$" and assign output to $vLengthShares$#
27 ⏚ String Operation: Get length of "$vStockSymbol$" and assign output to $vLengthStockSymbol$#
28 ⏚ String Operation: Get length of "$vTradeDesc$" and assign output to $vLengthTradeDesc$#
29 ⏚ String Operation: Get length of "$vTradeID$" and assign output to $vLengthTradeID$#
30 Comment: =====
31 Comment: Evaluate data and send email based on one of two situations.
32 Comment: =====
33 Comment: Evaluate length of each variable.
34 IF If $vLengthParty1$ Equal To (=) "0" OR $vLengthParty2$ Equal To (=) "0" OR $vLengthReporterName$ Equal
35 Comment: Scenario: Missing Data and click Save (replacing the default text).
36 ✉ Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
37 ELSE Else
38 Comment: Scenario: All Data Present
39 ✉ Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
40 ✖ End If
41 Comment: =====
42 Comment: Cleanup: Delete all emails and files in folder.
43 Comment: =====
44 ✖ Delete All Messages on server: 'localhost'
45 ✈ Delete Files "C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf"
```

Section 6. Configuring a trigger

Next, you configure a trigger to make this bot run when an email arrives in the `tradeList@sib.com` inbox. The bot looks for an email with an attachment.

- 1. Minimize the Workbench window and return to the Enterprise Client window to locate your `Trade Booking.atmx` bot and click the **TRIGGER** tab.



- 2. Enter the following values in the corresponding fields:

- **Trigger Type:** Select **Email Message**
- **Action:** Select **When new email arrives**
- **Host:** localhost
- **Port:** 143
- **Poll every:** 1 mins
- **User Name:** tradeList@sib.com
- **Password:** tradeList

3. Click Save.

PROPERTIES SCHEDULE TRIGGER

Trigger Type : Email Message ?

Action : When new email arrives

Host : localhost

Port : 143 Poll every 1 mins

User Name : tradeList@sib.com

Password : *****

Server uses secure connection (SSL)

Save Cancel

Trigger Type	Action
<input checked="" type="checkbox"/> Email Message	When new en

Section 7. Running the bot

In this section, you test the bot to make sure that it runs when the `tradeList@sib.com` email account receives a message with a PDF attachment.



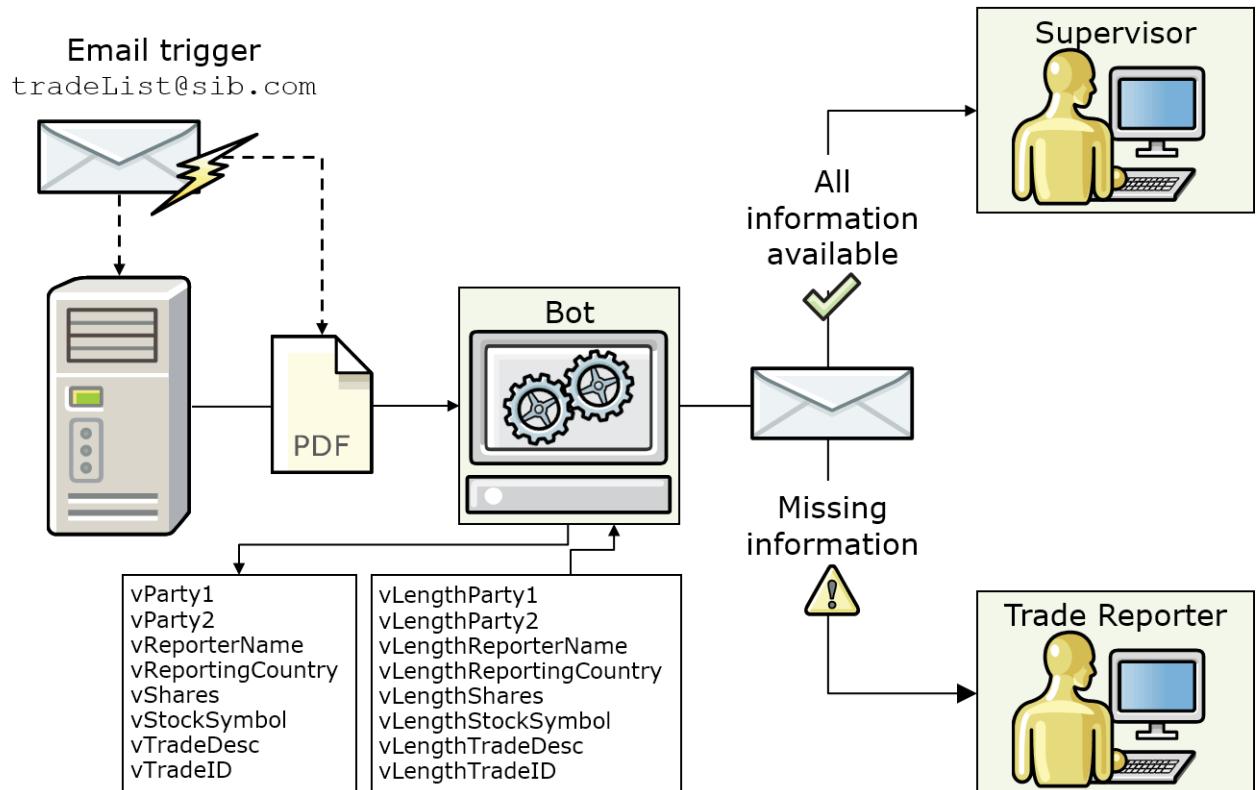
Information

When you run bots in the Workbench, you must not use the keyboard or mouse during execution. However, when a trigger causes a bot to run, the bot runs in the background so keystrokes and mouse movements do not affect execution.

Note: The bot should trigger within in a minute.

Bot actions:

- Connect to the email server and monitor the trade distribution list for trade receipt messages
- Download trade receipt PDF attachments, and extract all data from the trade receipt PDF
- Verifies whether all required fields are present
- Send an email according to these conditions:
 - If all required information is available, the bot emails the trade reporter supervisor
 - If any of the required information is missing, the bot emails the trade reporter to notify them that information is missing.

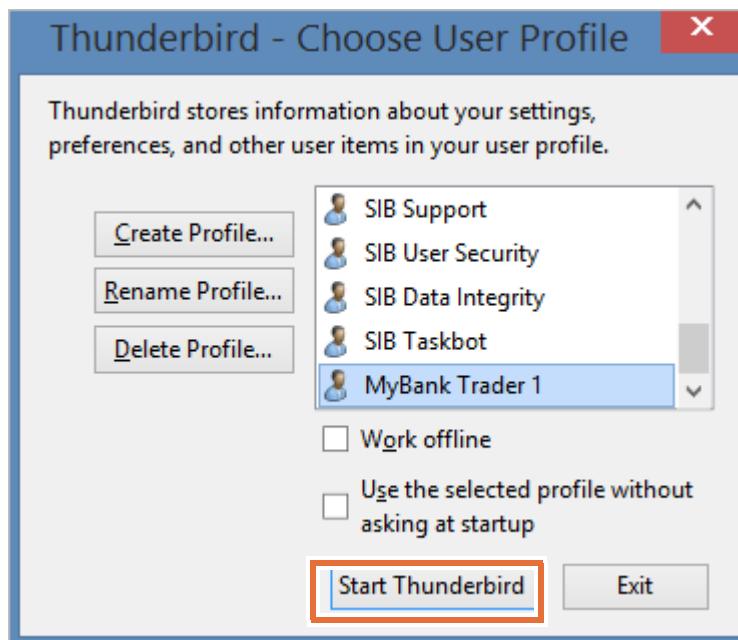


7.1. Testing the email response when all data is present

- ___ 1. In Windows Explorer, go to the C:\labfiles\trade_booking folder and delete the Trade_Receipt_MyBank.pdf file.
- ___ 2. Send an email with the Trade Receipt to the **tradeList@sib.com** distribution list from the MyBank Trader 1 account.
 - ___ a. On the desktop, double-click the **Mozilla Thunderbird** shortcut.

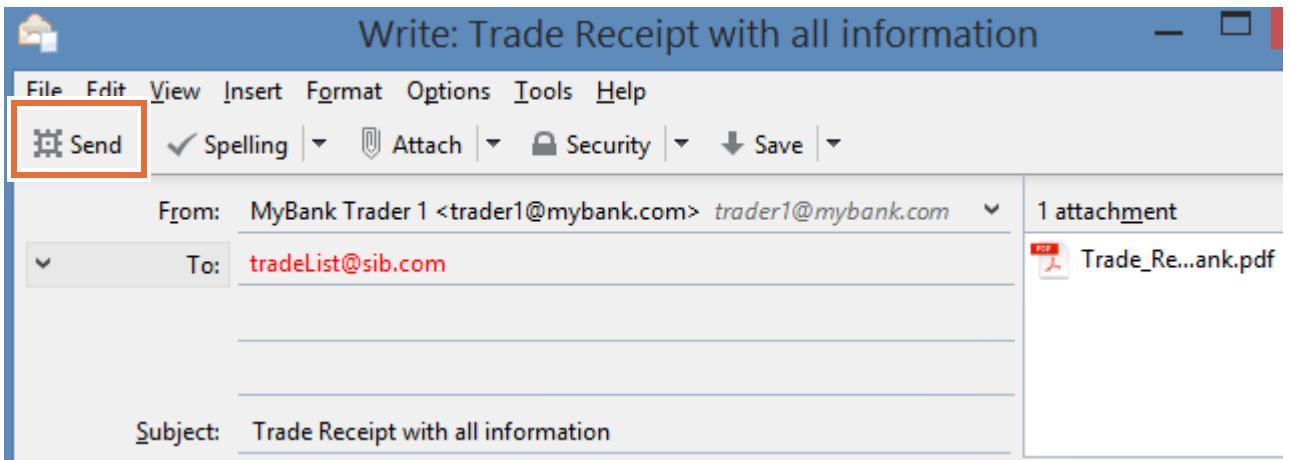


- ___ b. Select **MyBank Trader 1**, and click **Start Thunderbird**.

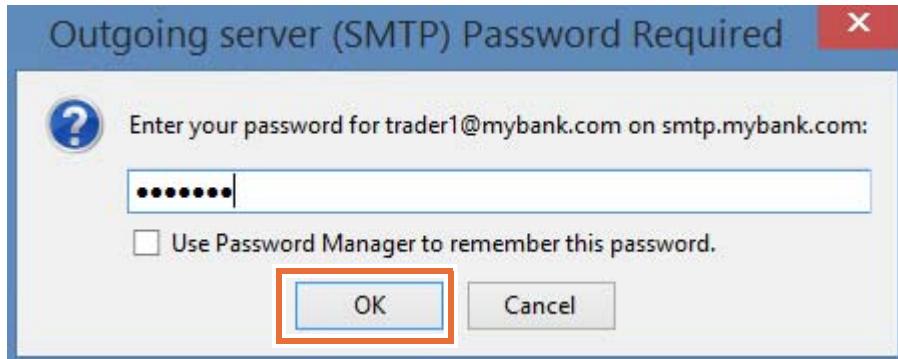


- ___ c. When prompted for a password, enter: trader1
- ___ 3. Send a trade receipt with all the correct information to **tradeList@sib.com**.
 - ___ a. On the Thunderbird toolbar, click **Write**.
 - ___ b. In the **To** field, enter: **tradeList@sib.com**
 - ___ c. In the **Subject** field, enter: **Trade Receipt with all information**
 - ___ d. On the email toolbar, click **Attach**, and go to the **C:\labfiles\trade_booking\trade_receipts_scenarios\all_information\Trade_Receipt_MyBank.pdf** file, and then click **Open**.

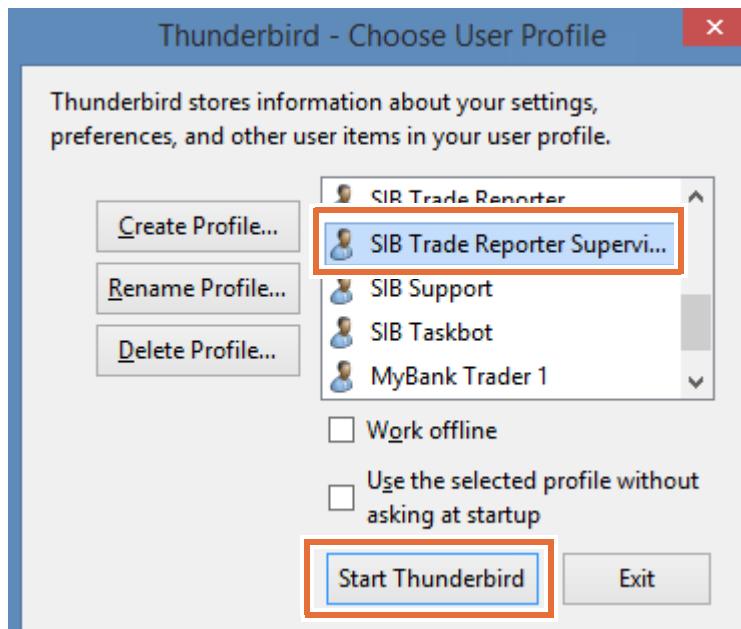
- ___ e. On the email toolbar, click **Send**.



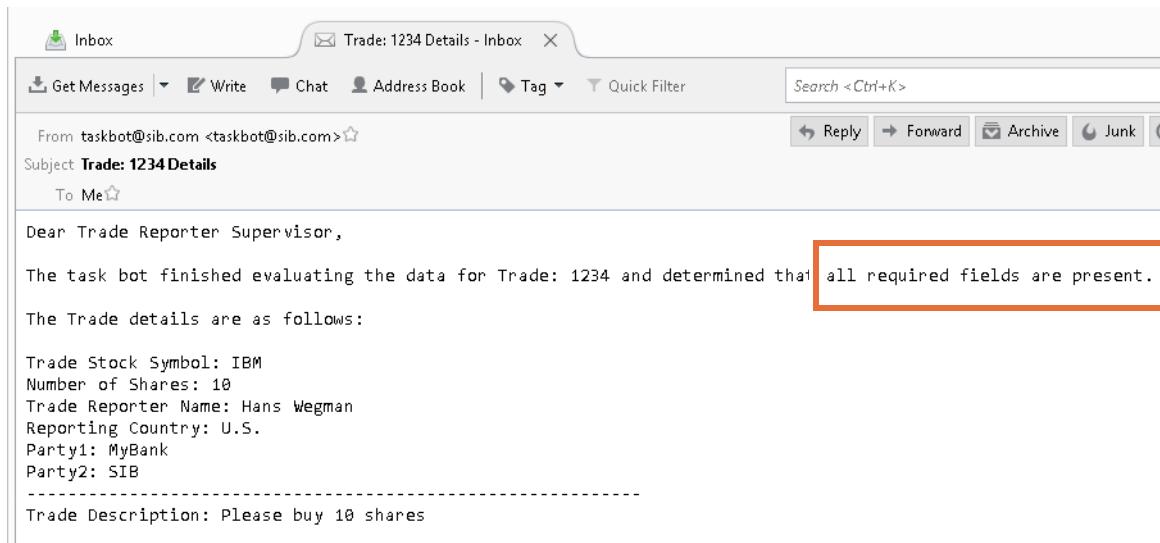
- ___ f. When prompted for a password, enter `trader1` and click **OK**.



- ___ 4. Close **Thunderbird**.
 ___ 5. Confirm that the SIB Trade Reporter Supervisor account received the notification email.
 ___ a. Restart Thunderbird by clicking the shortcut on the desktop.
 ___ b. Select the **SIB Trade Reporter Supervisor** user name, click **Start Thunderbird**.



- c. When prompted for a password, enter: tradeSupervisor
- d. In the Inbox, double-click the email with the subject **Trade: 1234 Details** to open it.
- e. Verify that the information in the email is correct.

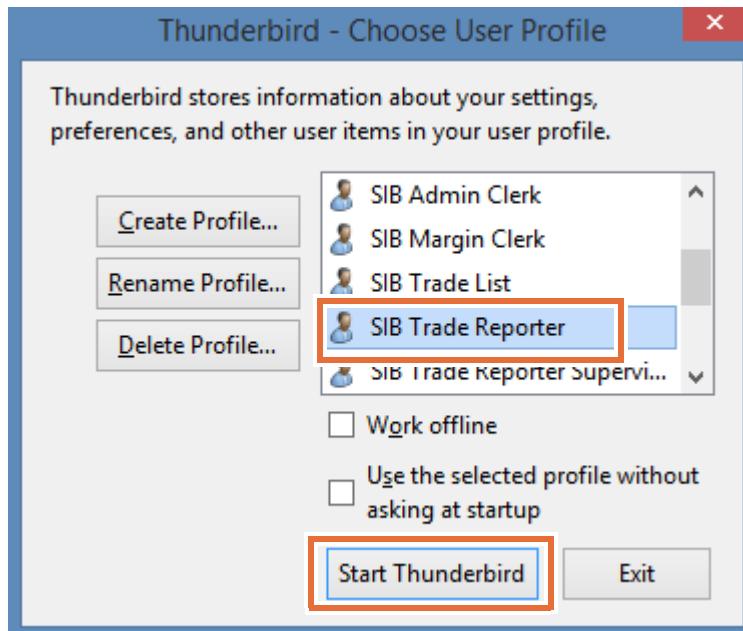


- 6. Close **Thunderbird**.

7.2. Testing the email response when all data is missing

- 1. Restart Thunderbird as **MyBank Trader 1** and use **trader1** as the password.
- 2. Send a trade receipt with missing information to **tradeList@sib.com**.
 - a. On the Thunderbird toolbar, click **Write**.
 - b. In the **To** field, enter: **tradeList@sib.com**
 - c. In the **Subject** field, enter: **Trade Receipt** with missing information
 - d. On the email toolbar, click **Attach**, and go to the **C:\labfiles\trade_booking\trade_receipts_scenarios\missing_information\Trade_Receipt_MyBank.pdf** file, and then click **Open**.
 - e. On the email toolbar, click **Send**.
 - f. When prompted for a password, enter **trader1** and click **OK**.
- 3. Close **Thunderbird**.

- ___ 4. Confirm that the SIB Trade Reporter email account received the notification email.
- ___ a. Restart Thunderbird as **SIB Trade Reporter**.



- ___ b. When prompted for a password, enter: tradeReporter
- ___ c. In the Inbox, double-click the email with subject **Trade: 1234 Details**.
- ___ d. Verify that the information in the email is correct.

From: taskBot@sib.com <taskBot@sib.com> ★ Reply Forward Archive Junk Delete 1

Subject: **Trade: 1234 Details**

To: Me ★

Dear Trade Reporter,

Task Bot has finished evaluating data for Trade: **1234** and has determined all required fields are **NOT** present.

Trade Stock Symbol: **IBM**
 Number of Shares: **10**
 Trade Reporter Name: **Hans Weiman**
 Reporting Country: **U.S.**
 Party1:
 Party2: **SIB**

Trade Description: **Please buy 10 shares**

Thank you from your friendly neighborhood Task Bot

- ___ 5. Close **Thunderbird**.



Troubleshooting

If you run into errors while running your bot, you can check the log file in the following directory:

C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Trade Booking.log

Section 8. *Optional.* Viewing the solution file

If you were unable to complete and run the bot successfully, you can open the solution file for this exercise to see how the bot should be coded.

- ___ 1. Go to C:\labfiles\trade_booking\solution, right-click the Trade Booking solution.atmx file and click **Edit**.
The file opens in the Workbench.
 - ___ 2. To run the bot, you can either run in Debugging mode or you must first configure the Trigger as described in [Section 6, "Configuring a trigger"](#)[Section 6, "Configuring a trigger,"](#) on page 6-27.
-



Information

If you want to load the solution file from the Enterprise Client, move the Trade Booking solution.atmx file to the following directory: C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks

The Enterprise Client **My Tasks** view automatically accesses files in that folder.

From the **My Tasks** list, you can either run the bot by double-clicking it or you can view it by right-clicking the task and clicking **Edit**.

End of exercise

Exercise review and wrap-up

In this exercise, you configured a bot that loops through emails in an email account and downloads the PDF attachments from the emails. You configured conditional logic to determine whether data was missing from the PDF and to send an appropriate email response based on the result. You configured the bot to delete the downloaded attachments and the emails in the monitored email account. You also worked with an email trigger and tested the conditional logic in the bot.

Exercise 7. Creating an interactive bot to check values in disparate systems

Estimated time

01:30

Overview

In this exercise, you learn how to add interactive components, such as text entry windows, to a bot. You also learn how to use the Web Recorder to work with web page data.

Objectives

After completing this exercise, you should be able to:

- Create interactive prompts in bots
- Use the Web Recorder to extract web page data

Introduction

Scenario: During the SIB and MyBank system integration, management wants to make sure that the company data for each account uses the same format across their systems and documents. Inconsistent data formats can cause issues when customer accounts updates are committed to the different data sources.

Currently, account managers must check three places: the SIB intranet customer list, a SIB Windows application that is built on Microsoft Access, and customer data that is retrieved from a MyBank database. SIB management wants to automate the account lookup process so the account manager can identify differences more quickly.

As a bot developer, you must develop a bot that automates these tasks:

- Prompt the user to enter an account number
- Use the account number to search the SIB intranet customer list
- If the account is found:
 - Retrieve the data from the three data sources
 - Display the three versions of the data in a message window
- If the account is not found:
 - Display a message that the account does not exist

- Stop running
-

This exercise includes these sections:

- [Section 1, "Starting a new task and defining local variables"](#)
- [Section 2, "Creating an interactive prompt and checking information from a web page"](#)
- [Section 3, "Checking information from a Microsoft Access-based Windows application"](#)
- [Section 4, "Extracting the company value from an Excel spreadsheet"](#)
- [Section 5, "Optional. Viewing the solution file"](#)

Requirements

This exercise requires the computer lab environment that was built for this course.

Section 1. Starting a new task and defining local variables

In this section, you create a task and define the local variables that the bot uses.

- 1. Go to the Enterprise Client.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

- 2. In the Enterprise Client, create a task with the Workbench, by clicking **New** on the toolbar and selecting **Workbench** when prompted for a tool option.
- 3. In the Workbench, open the Variable Manager, and add the following variables:

Type	Name	Value
Value	vAccountNumber	Leave this field blank
Value	vAccountCheck	Leave this field blank
Value	vCompany1	Leave this field blank
Value	vCompany2	Leave this field blank
Value	vCompany3	Leave this field blank
Value	vFound	false

- 4. Save your work, and name the task: Data Consistency

Section 2. Creating an interactive prompt and checking information from a web page

In this section, you add the command for the first bot task: prompting the user to enter an account number. The bot searches the list of SIB customers to determine whether the account number exists in the SIB system. If the account number is not found, the bot shows a message that alerts the user.

2.1. Creating an interactive prompt

- ___ 1. Add the following comment to the Workbench:

Prompt the user to enter an account number and assign it to the vAccountNumber variable

- ___ 2. Add a **Prompt For Value** command that asks the user to enter an account number.

- ___ a. In the **Commands** list, double-click **Prompt** and add the **Prompt For Value** command to the Actions List.
- ___ b. In the Prompt window, click **Select window**, and select **Taskbar**.
- ___ c. In the **Please enter custom prompt message** field, enter:
Enter the customer account number:
- ___ d. Select **Assign the value to an existing variable**.
- ___ e. Click **Select Variable**, and select **vAccountNumber**.

This action takes the value that the user enters, and assigns it to the vAccountNumber local variable. This variable is used by the bot to locate customer data across different applications.

- ___ f. Click **Save**.

2.2. Checking the customer list web page for account information

- ___ 1. Add the following comment to the Workbench. Make sure that it is after the **Prompt** command.

Source: SIB Intranet

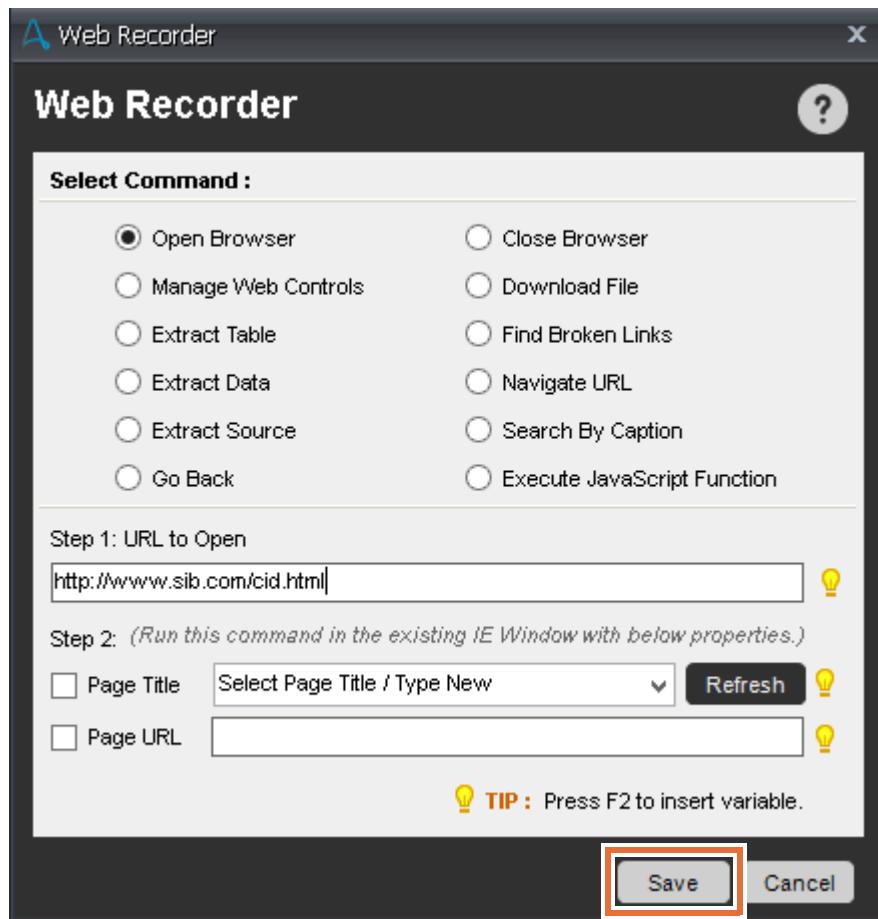
- ___ 2. Add another comment to the Workbench:

Use the Web Recorder to check the SIB customer list for the account number that is entered by the user

- ___ 3. Add the Open Browser Web Recorder command after the comment line.

- ___ a. In the **Commands** list, double-click the **Web Recorder** section to expand it.
- ___ b. From the Web Recorder section, add the Open Browser command after the Comment.
- ___ c. In the **URL to Open** field, enter: <http://www.sib.com/cid.html>
- ___ d. Leave the options in the Step 2 section of the Web Recorder window clear.

- ___ e. Click Save.



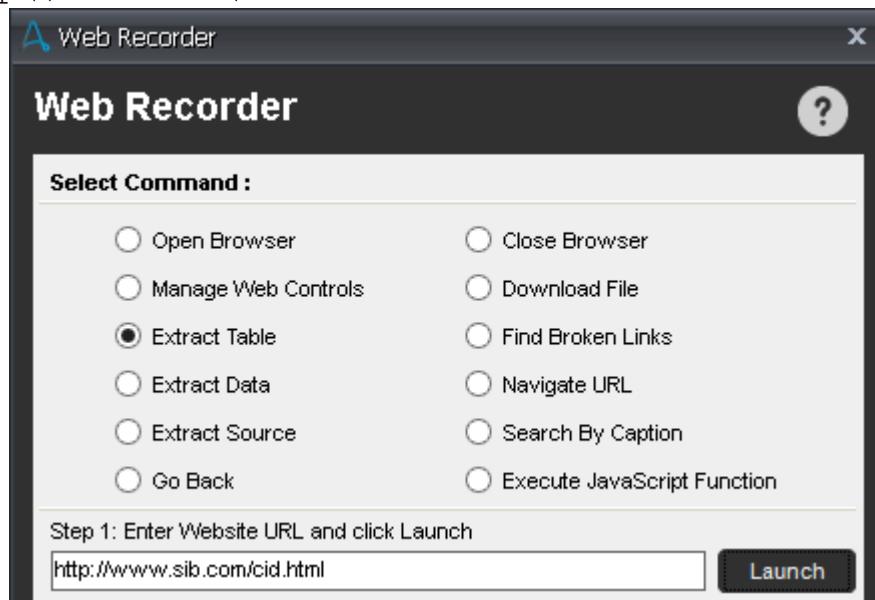
Information

The Web Recorder commands open Internet Explorer as the web browser.

- ___ 4. Add a command to extract data from the table on the SIB Customers web page, and find whether the inputted account number is in the customer list.
- ___ a. From the Web Recorder section of the **Commands** list, add the Extract Table command after the Open Browser command.

- ___ b. In Step 1 of the Web Recorder window, in the **Enter Website URL** field, enter the following URL, and click **Launch**:

<http://www.sib.com/cid.html>



- ___ c. In Step 2 of the Web Recorder window, click **Capture Table**.



The Enterprise Client goes to the [sib.com](http://www.sib.com) Customers web page, and shows the following message:

Click on the table you want to extract from the page. Press 'Esc' to cancel and continue recording.

Click on the table you want to extract from the page. Press 'Esc' to cancel and continue recording.

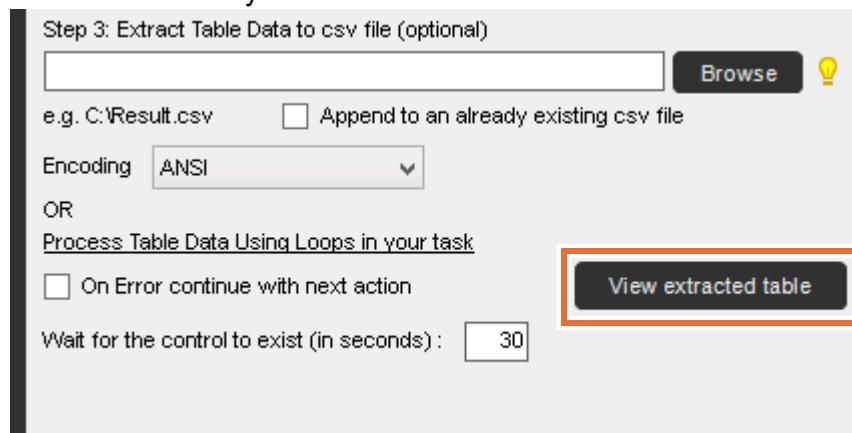
Account Number	Company	First Name	Last Name	Phone Number
1234	MyBank	Meredith	Chu	650-555
2345	Miniloan	James	Doe	202-555

- ___ d. In the web page, click the customer table to select it.

The Enterprise Client returns to the Web Recorder window. The table is identified by the Enterprise Client as table 1.

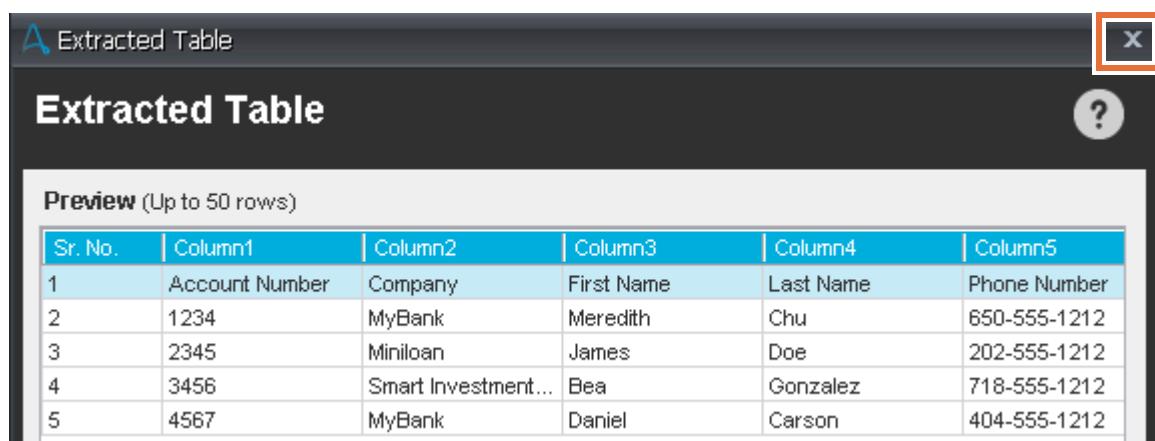


- ___ e. In Step 3 of the Web Recorder window, click **View extracted table** to confirm that the Web Recorder correctly read the table.

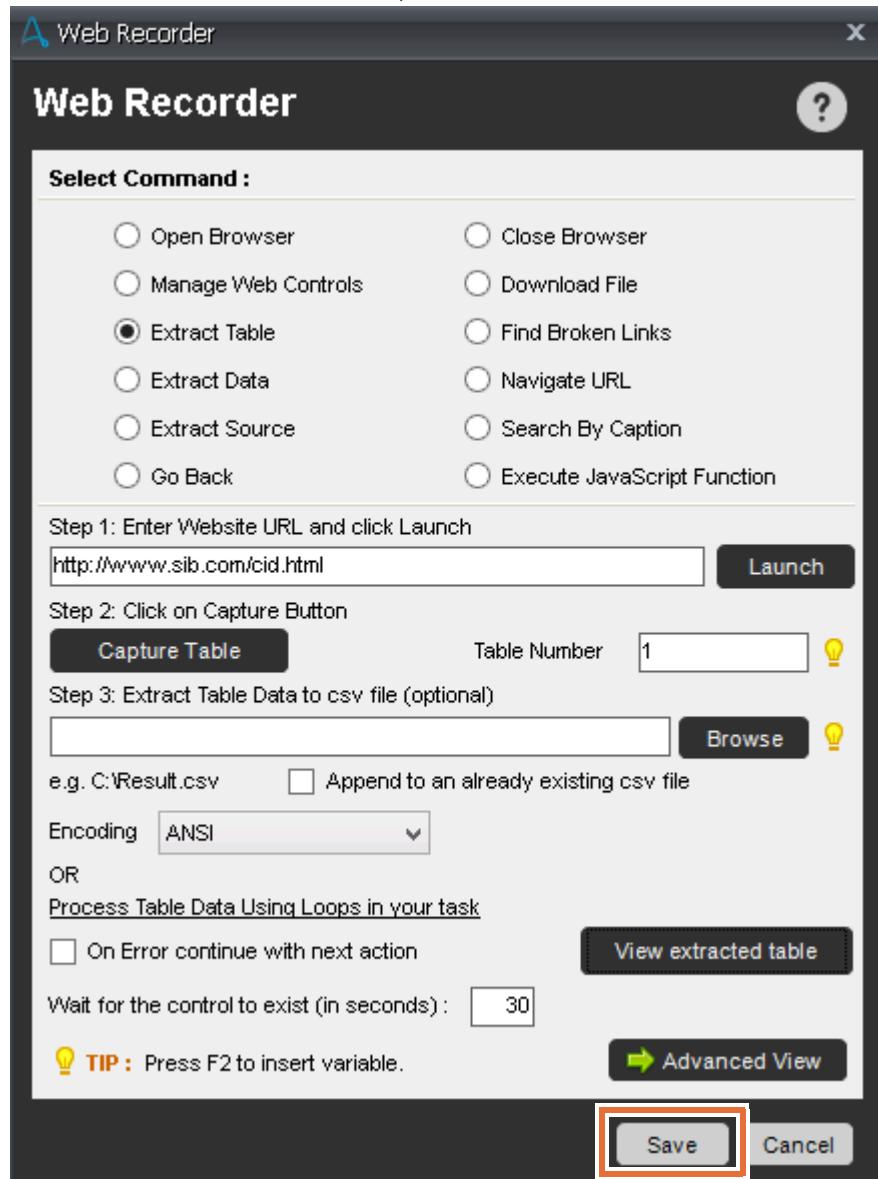


The table opens in a separate window.

- ___ f. When you are finished reviewing the table, close the window.



- ___ g. Back in the Web Recorder window, click **Save**.



Next, you set up a loop to iterate through the table data and check whether the account number that was entered by the user is in the table data.

- ___ 5. Add the following comment after the `Extract Table` command.

Loop through the extracted table data to look for the value in vAccountNumber

- ___ 6. Add a loop command to iterate through each row in the web page table.

- ___ a. In the **Commands** list, go to the **Loop** section and expand it.
- ___ b. From the **Loop** section, add `Each Row In An Internet Explorer Table` after the comment you added in Step 5.
- ___ c. In the Loop window, leave the settings at their default values, and click **Save**.

- ___ d. In the default comment that is added with the `Loop` command, notice that you use the `$TableColumn$` system variable to handle table data.

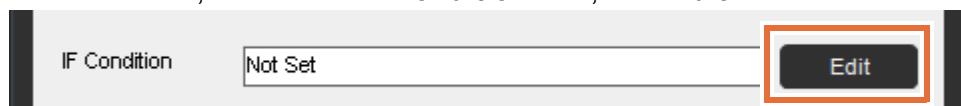


- ___ 7. Replace the default loop comment with the following comment:

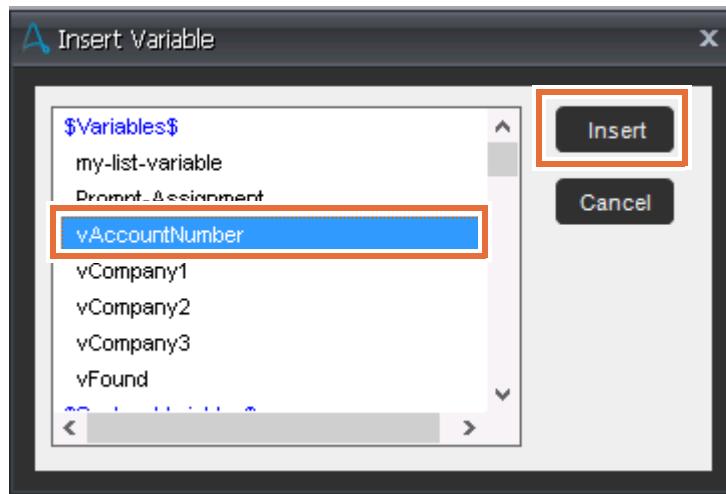
Use an `if` condition to determine whether the value in `vAccountNumber` is in the table data

- ___ 8. Add an `If` command to check the value of `vAccountNumber` against the table data.

- ___ a. In the **Commands** list, go to the **If/Else** section and expand it.
- ___ b. From the **If/Else** section, add the **Variable** command after the comment that you entered in Step 7.
- ___ c. In the If window, next to the **IF Condition** field, click **Edit**.



- ___ d. In the If Variable window, click the **Variable** field and press F2 (or Fn+F2).
- ___ e. In the Insert Variable window, in the **\$Variables\$** section, select `vAccountNumber`, and click **Insert**.



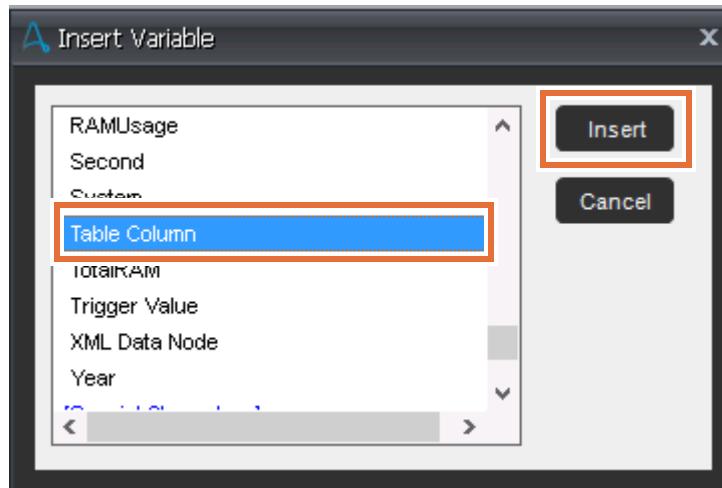
- ___ f. From the **Operator** list, select **Equal To (=)**.

- ___ g. For **Value**, select **Variable**.

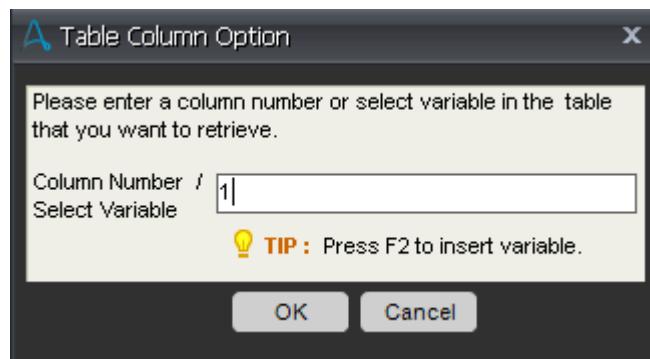


- ___ h. Click the **Value > Variable** field, and press F2 (or Fn+F2).

- ___ i. In the Insert Variable window, scroll to the **\$SystemVariables\$** section, select **Table Column**, and click **Insert**.

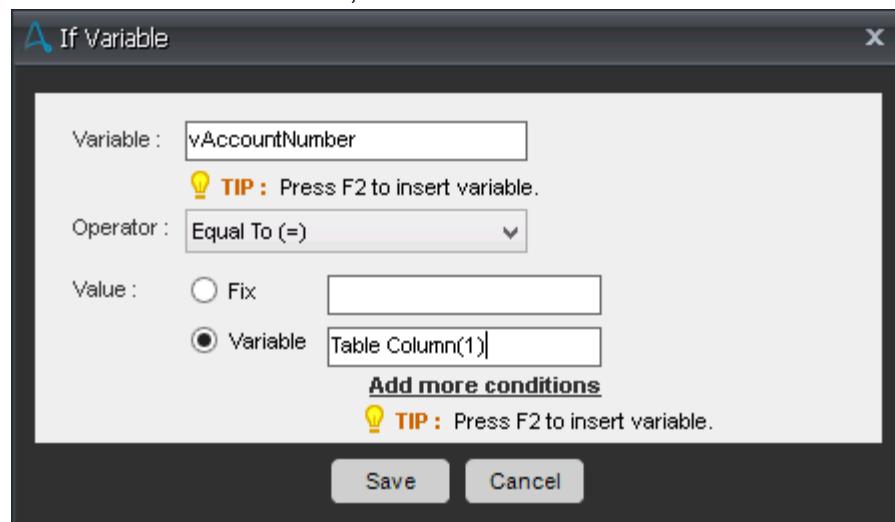


- ___ j. In the Table Column Option window, enter 1 and click **OK**.



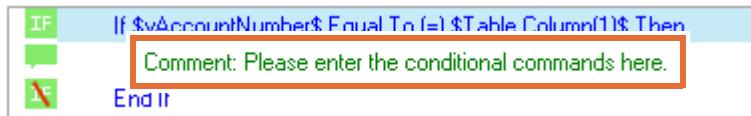
This action identifies the contents of table column 1, which represent the account number for each customer.

- ___ k. Back in the If Variable window, click **Save**.

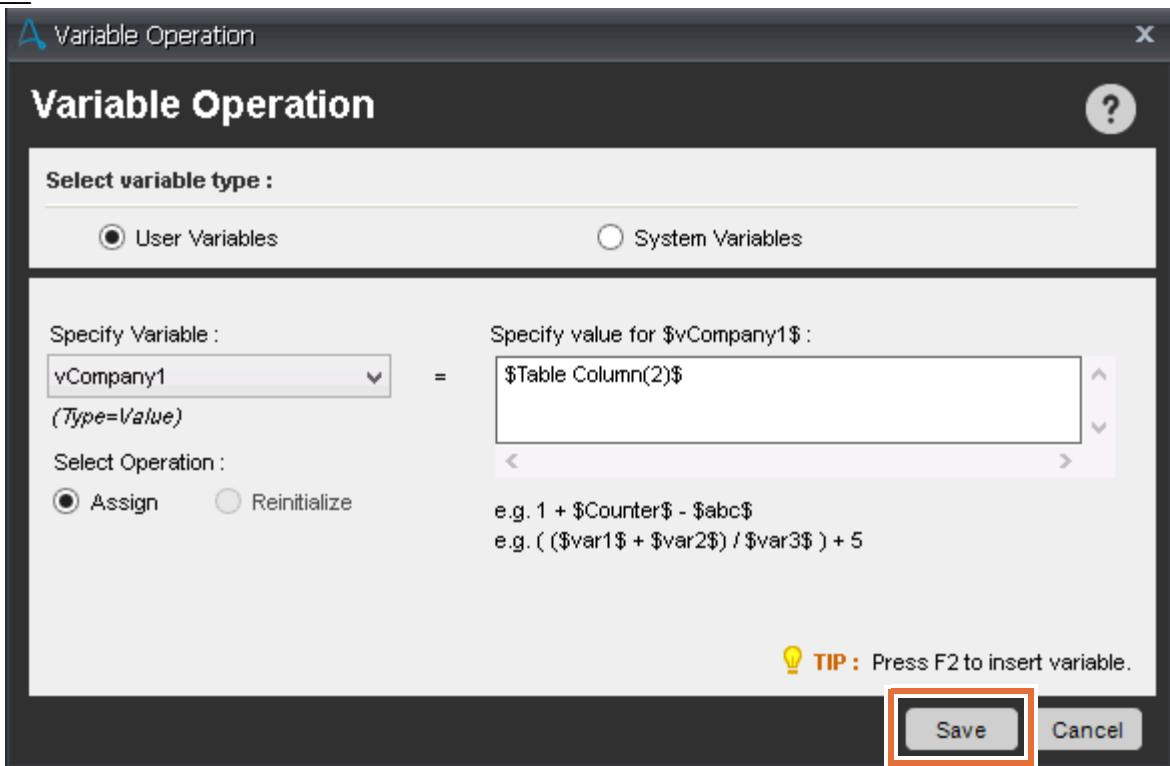


Note that adding the **IF** command adds a default comment and an **End IF** command.

- ___ 9. Edit the `IF` command comment to describe the actions that are taken when the condition is met.
- ___ a. In the Workbench, double-click the `IF` command comment to open the Comment window.



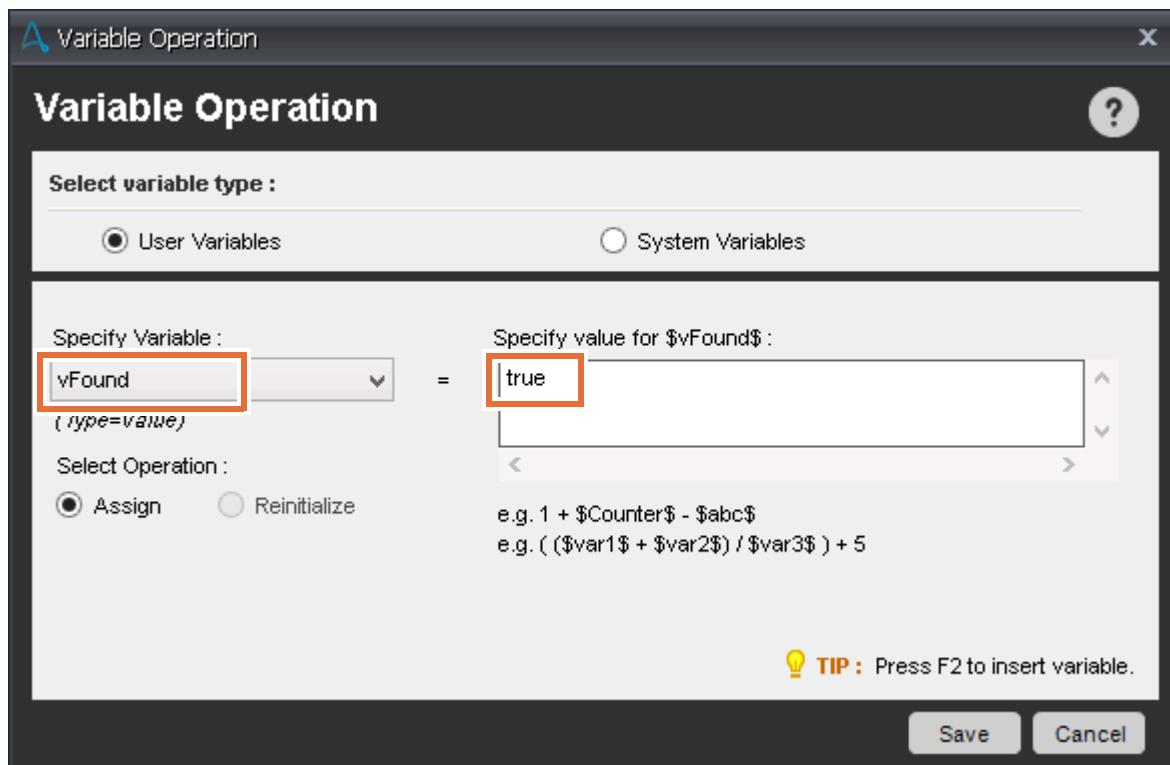
- ___ b. Delete the existing comment text, and type:
If the value in vAccountNumber is found, assign table column(2) value to vCompany1 and update value in vFound to true
- ___ c. Click **Save**.
- ___ 10. Add a Variable Operation command to assign the value in `$Table Column(2)$` to `$vCompany1$`.
- ___ a. From the **Commands** list, add a Variable Operation command to the Workbench. Make sure it is after the comment that you updated in [Step 9](#).
- ___ b. In the Variable Operation window, click **Select variable** and select **vCompany1**.
- ___ c. In the **Specify value for \$vCompany1\$** field, enter: `$Table Column(2)$`
- ___ d. Click **Save**.



- ___ 11. Add a Variable Operation command to update the value in `vFound` to `true`.
- ___ a. From the **Commands** list, add a Variable Operation command after the Variable Operation command that you defined in [Step 10](#).

___ b. In the Variable Operation window, enter the following settings:

- **Specify Variable:** vFound
- **Specify value:** true



___ c. Click **Save**.

The If conditions for the loop are now complete, and the loop is also complete.

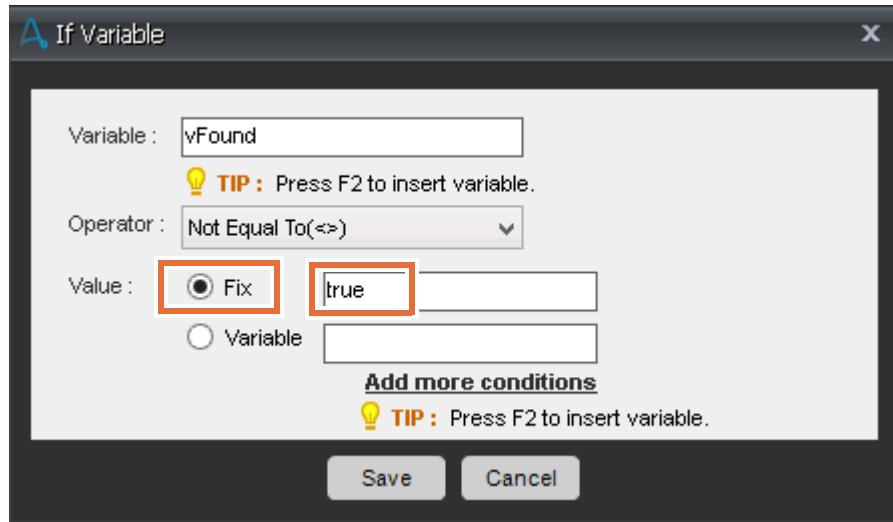
2.3. Notifying the user that the account is not found

Next, you create an interactive window that notifies the user when the account number is not found. If the vFound variable does not evaluate to true, a message opens.

- ___ 1. Add the following comment to the Workbench. Make sure it is after the End Loop command.
If an account number match is not found, inform the user and stop the task
- ___ 2. Add an If command that determines whether vFound does not equal true.
 - ___ a. In the **Commands** list, go to the **If/Else** section, and add a **Variable** command to the Workbench. The If window opens.
 - ___ b. Next to the **IF Condition** field, click **Edit**.

___ c. In the If Variable window, enter the following settings.

- Variable: **vFound**
- Operator: **Not Equal To (<>)**
- Value:
 - o Select **Fix**.
 - o Enter: **true**



___ d. Click **Save**.

___ 3. Edit the default If command comment to state:

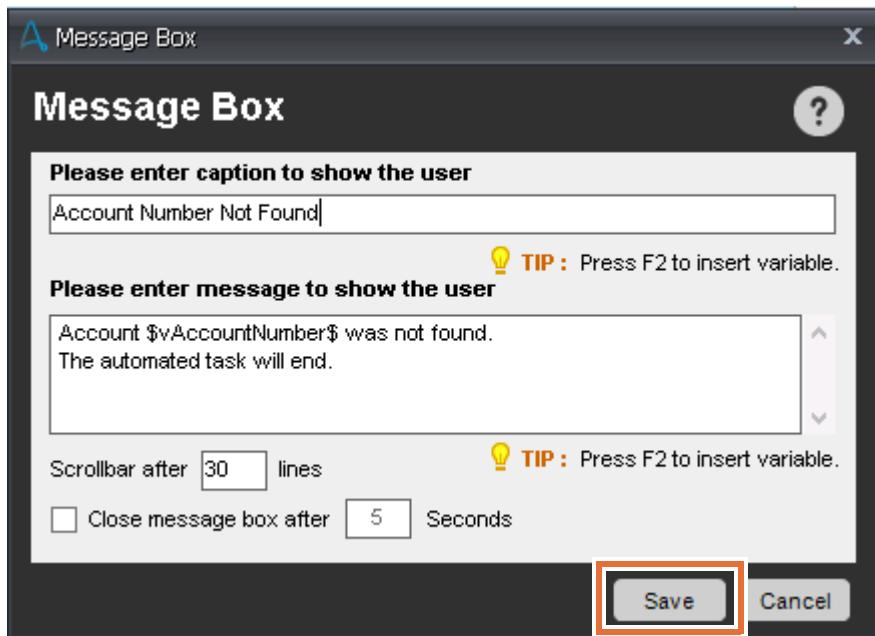
Show a message that tells the user that the account number was not found, and stop the task

___ 4. Add a Message Box command that notifies the user that the account number was not found.

- ___ a. From the **Commands** list, add a Message Box command to the Workbench after the comment that you updated in Step 3.
- ___ b. In the Message Box window, change the value in the **Please enter caption to show the user** field to: Account Number Not Found
- ___ c. In the **Please enter message to show the user** field, enter the following message:

Account \$vAccountNumber\$ was not found.
The automated task will end.

- ___ d. Click **Save**.



- ___ 5. From the **Commands** list, add a Close Browser **Web Recorder** command to the Workbench. Make sure that this command is after the Message Box command.
- ___ 6. Add a Stop Task Task command.
 - ___ a. In the **Commands** list, expand the Task section.
 - ___ b. From the Task section, add Stop Task to the Workbench. It should be after the Close Browser command and before the End If command.
- ___ 7. Add a Close Browser **Web Recorder** command after End If line.
 - ___ a. From the **Commands** list, go to the Web Browser section and double-click the Close Browser command.
 - ___ b. Click **Save**.
- ___ 8. Save your work.

2.4. Notifying the user that the account is found

Next, you create an interactive window that notifies the user when the account number is correctly found. When the vFound variable evaluates to `true`, the message window opens and lists the data source. Currently, only the first company data source is displayed. As you continue building this bot, you add to this Message Box command so that all three data sources are displayed.

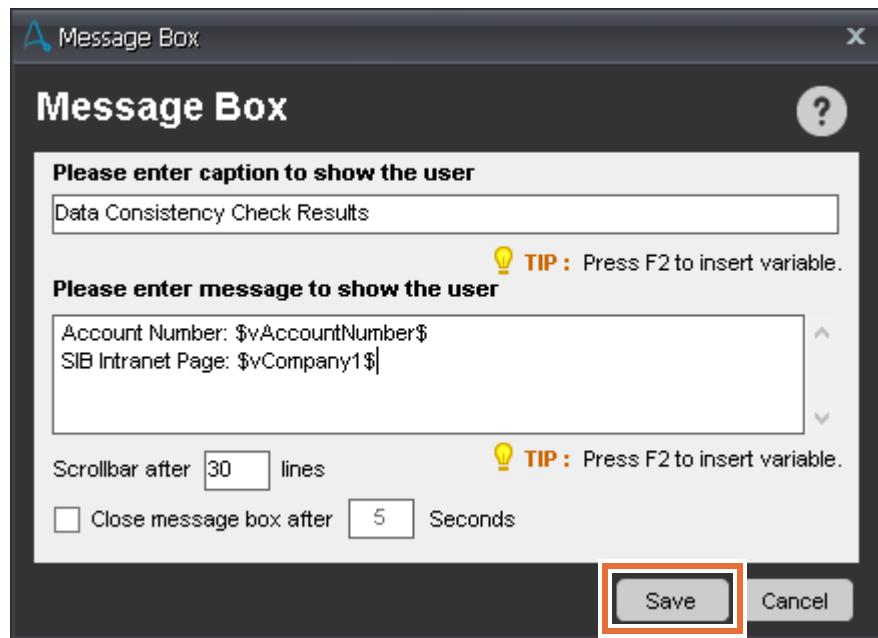
- ___ 1. Add a message box to display the account number and the company data source.
 - ___ a. From the **Commands** list, add a Message Box command.
 - ___ b. In the **Please enter caption to show the user** field, enter:

Data Consistency Check Results

- ___ c. In the **Please enter message to show the user** field, enter:

Account Number: \$vAccountNumber\$
SIB Intranet Page: \$vCompany1\$

- ___ d. Click **Save**.

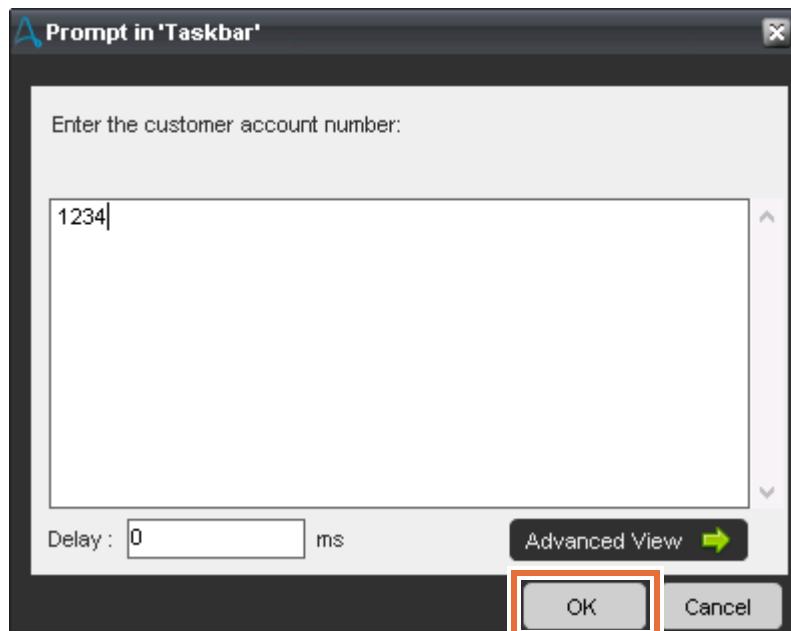


- ___ 2. Save your work.

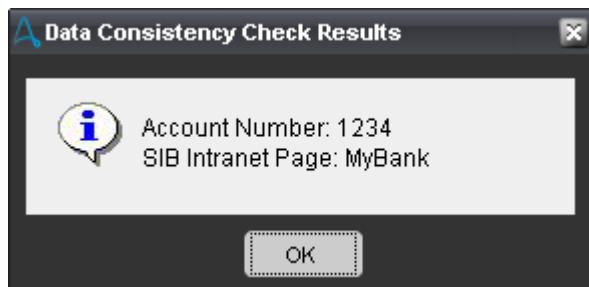
2.5. Testing the bot

Next, you run the bot to confirm that the correct message is displayed for a valid account number and an invalid account number.

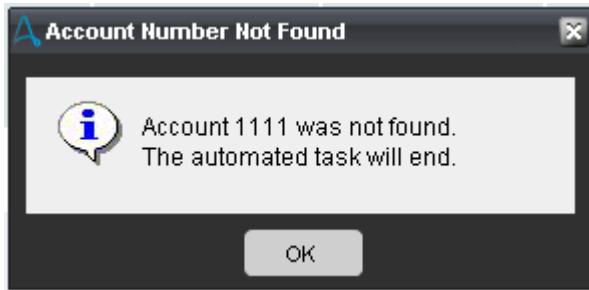
- ___ 1. On the toolbar, click **Run**.
___ 2. In the **Enter the customer account number** field, enter: 1234 and click **OK**.



- ___ 3. Confirm that the message shows that the account was found and lists SIB Intranet Page: MyBank as the company data source.



- ___ 4. Click **OK** to close the window.
- ___ 5. On the toolbar, click **Run** again.
- ___ 6. In the “Prompt in ‘Taskbar’” window, enter 1111 and click **OK**.
- ___ 7. Confirm that the message displays Account Number 1111 as not found, and click **OK** to close the window.



Section 3. Checking information from a Microsoft Access-based Windows application

In this section, you add instructions to the bot for checking the company data in an Access-based Windows application. This application keeps a list of customers and their contact information.



Note

The Microsoft Access file is in: C:\labfiles\data_consistency

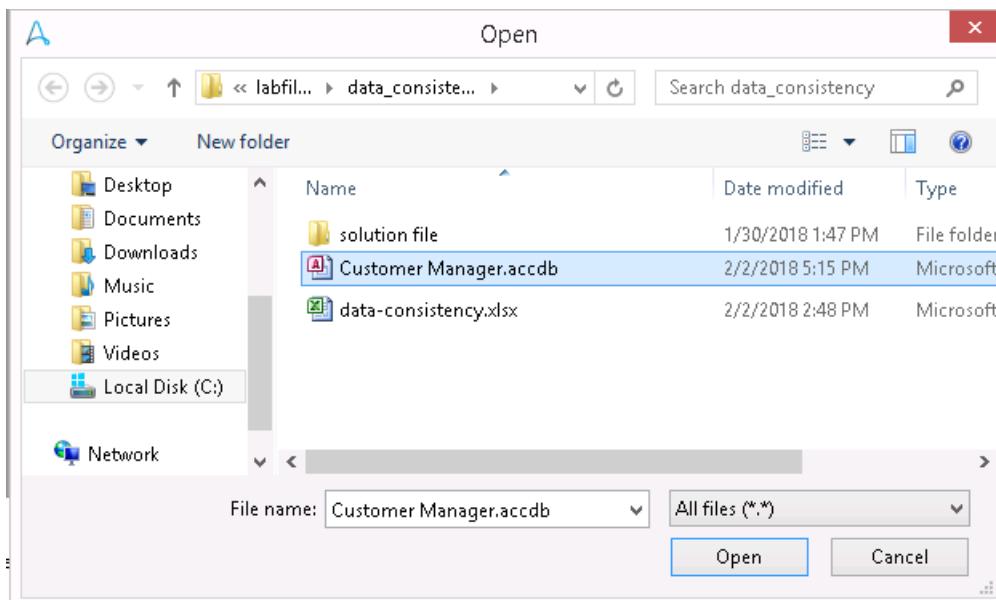
3.1. Opening the Microsoft Access application and logging in

- ___ 1. Add the following comment after the Close Window command for the SIB Intranet browser.
Source: Windows application (Access)
- ___ 2. Add another comment:
Open the Customer Manager application
- ___ 3. Add a command to open Customer Manager.accdb.
 - ___ a. From the **Commands** list, add the Open Program/File command to the Workbench. Make sure that it is after the comment that you entered in Step 2.
 - ___ b. In the Open Program/File window, in the **Step 1: Program/File Path** field, enter:
C:\labfiles\data_consistency\Customer Manager.accdb

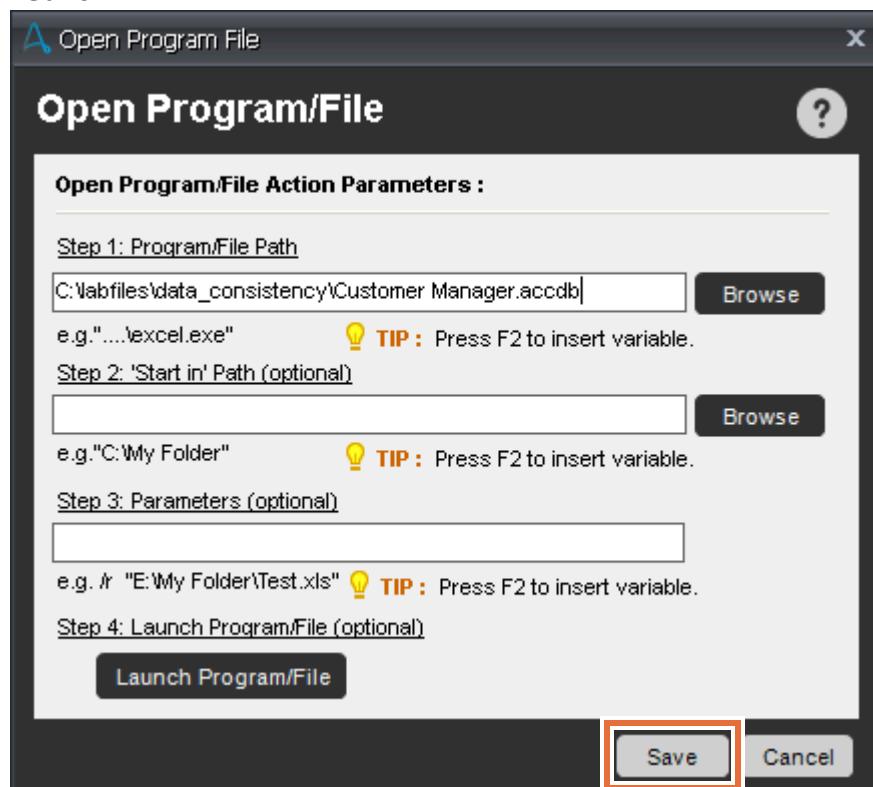
**Note**

You can also enter the file path by:

- Clicking **Browse** next to the field.
- In the Open window, going to the C:\labfiles\data_consistency folder
- Selecting the Customer Manager.accdb file.
- Clicking **Open**.

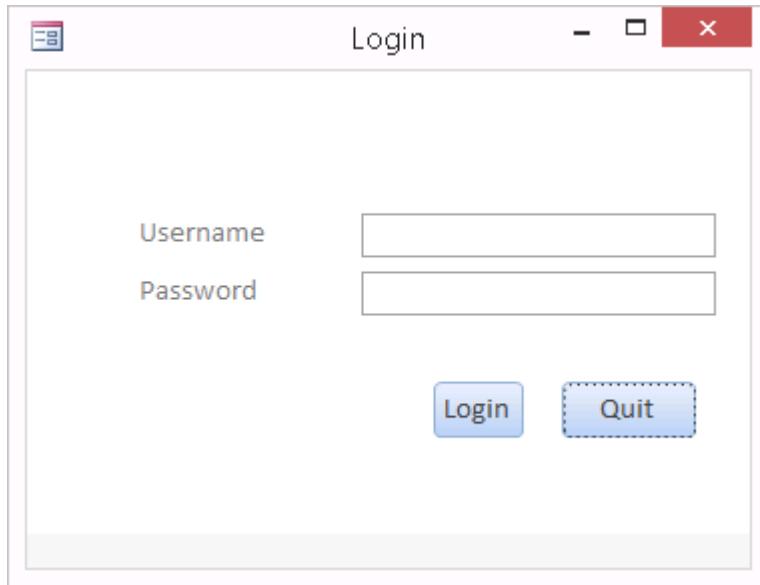


___ c. Click **Save**.



- ___ 4. In Windows Explorer, go to C:\labfiles\data_consistency, and double-click **Contact Manager.accdb** to open the application.

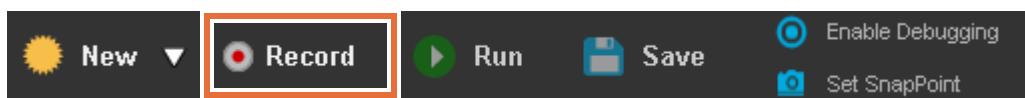
The Login window opens. You can minimize the window, but you need to keep it open during this exercise.



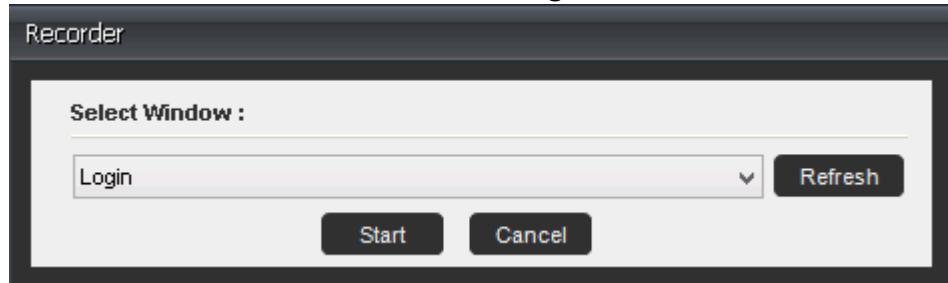
Reminder

This login screen emulates an application login process. You can minimize the window, but keep it open so you can select it for automation tasks.

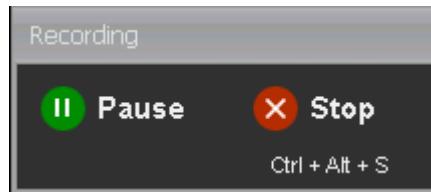
- ___ 5. Add a Resize Window command to move the login window to a specific location.
- ___ a. In the **Commands** list, expand **Windows Actions**, and double-click the **Resize Window** command.
 - ___ b. In the Window Actions window, enter the following settings and click **Save**:
 - **Select Window:** Login
 - **Top:** 400
 - **Left:** 600
 - **Height:** 295
 - **Width:** 380
- ___ 6. Add the following comment to the Workbench:
- Use the Smart Recorder and Object Cloning command to log in to the application
- ___ 7. Use the Smart Recorder to capture the application login credentials.
- ___ a. In the Workbench toolbar, click **Record** to start the Smart Recorder.



- ___ b. In the Recorder window, make sure that **Login** is selected, and click **Start**.



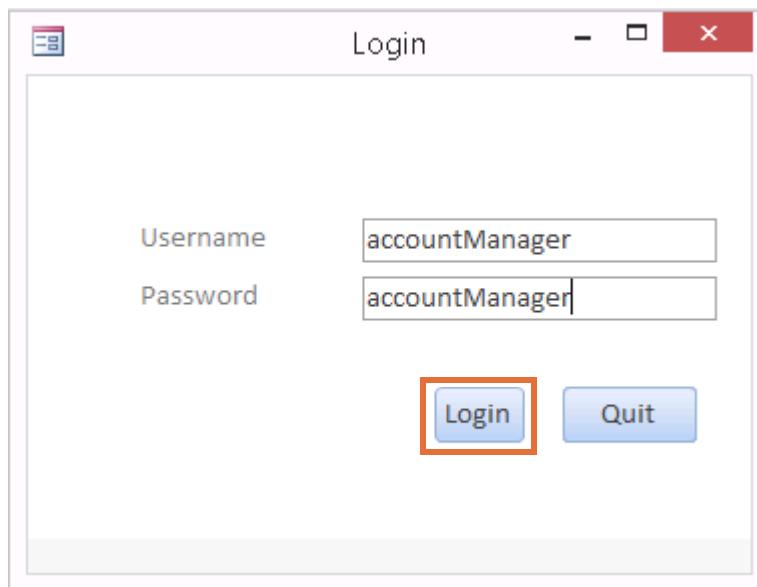
The Login window is brought to the foreground, and the Recording window is in the lower-right section of the monitor.



Reminder

You can move the Recording window to a different location on your display. The Enterprise Client remembers the new location of the window, and continues to open it at the new location.

- ___ c. Click the **Username** field, and enter: accountManager
- ___ d. Press Tab to go to the **Password** field, and type: accountManager
- ___ e. Click **Login**.



- ___ f. In the Recording window, click **Stop**.
- ___ g. Make sure that you leave the Contact Details window open.

These two `Object Cloning` commands are added to the Workbench:

- Object Cloning: Set Text of TextBox "Username" in window 'Login'; Value: "accountManager[TAB]account [SHIFT DOWN]m[SHIFT UP]anager"; Source: Window; Play Type: Object
- Object Cloning: Click On PushButton "Login" in window 'Login'; Click Type: Left Click; Source: Window; Play Type: Object

```
27   Object Cloning: Set Text of TextBox "Username" in window 'Login'; Value: "accountManager[TAB]account[SHIFT DOWN]m[SHIFT UP]anager"
28   Object Cloning: Click On PushButton "Login" in window 'Login'; Click Type: Left Click; Source: Window; Play Type: Object
```



Troubleshooting

When you use the Smart Recorder, do not worry if you capture unneeded Object Cloning commands in the Workbench. You can delete them.

3.2. Searching for the account number and retrieving the Company field value

In this section, you configure the bot to search for the value in vAccountNumber.

Because this application opens by default on the last record in the Access database, you must first check to see whether the account number in the current record matches the value in vAccountNumber.

If the account numbers do not match, the bot retrieves the correct record from the application by entering the value in vAccountNumber in the **Go to** field. In either case, it retrieves the company name from that record.

— 1. Add the following comment to the Workbench:

Check whether the account number in the current record matches vAccountNumber

— 2. Add another comment to the Workbench:

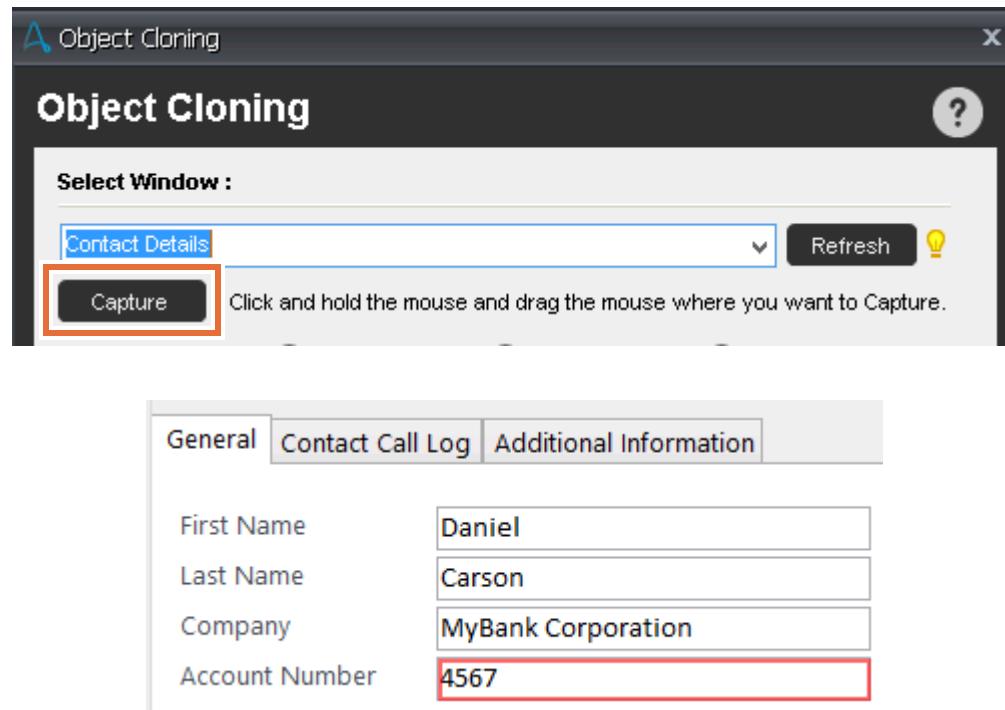
If the value in vAccountNumber does not match the account in the record, search for the account number in the application

— 3. Add an Object Cloning command to copy the value of the **Account Number** field.

— a. From the **Commands** list, add an Object Cloning command to the Workbench.

— b. In the Object Cloning window, in the Select Window section, select **Contact Details**.

- ___ c. Click the **Capture** button (*do not release the left-click mouse button*), and drag the mouse so that the **Account Number** field is selected.



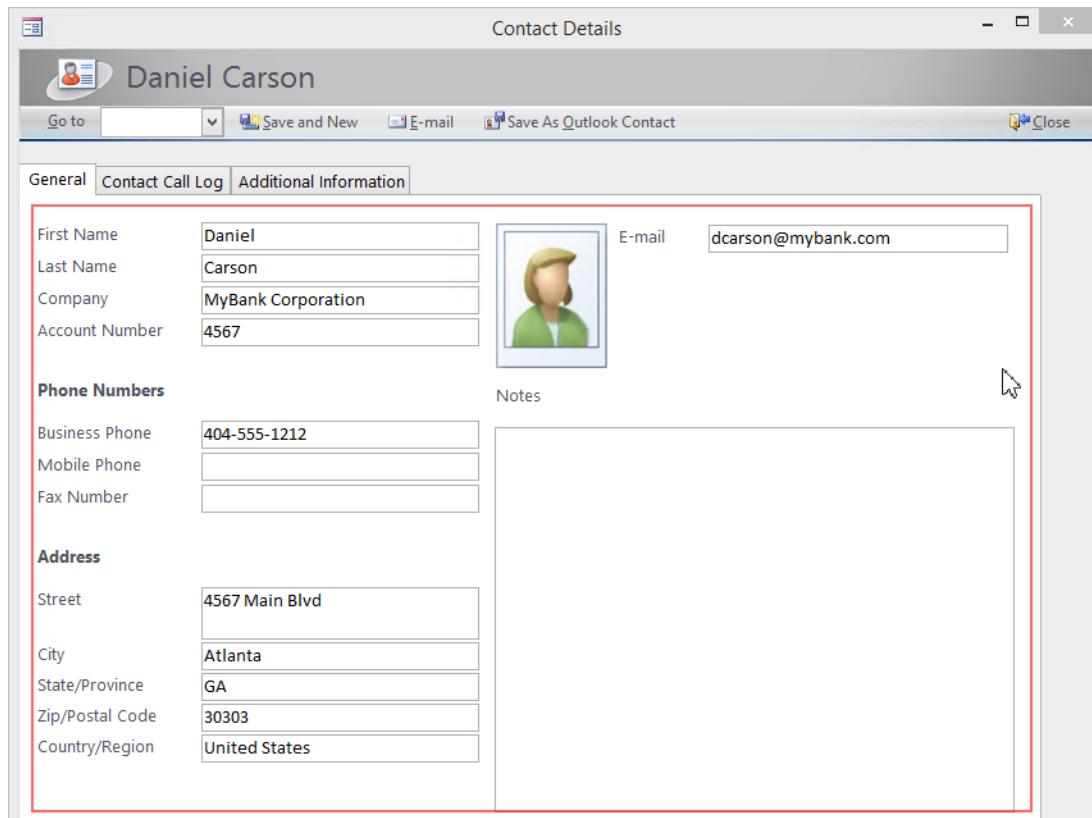
- ___ d. When the **Account Number** field is highlighted, release the mouse button to capture the field.



Information

When you click **Capture**, your left-click mouse button becomes a “record on” control. **Do not release** the mouse until your pointer selects the element you want. When you release the mouse button, the recorder is turned off and captures whatever UI element the mouse pointer was on. To make sure that you capture the correct UI element (in this case, the **Account Number** field), continue to hold the left-click mouse button until you are certain that the UI element is selected.

The Capture button opens the Contact Details window, and a red outline highlights the UI element that the mouse pointer is on. The outline might be on one of the window elements. For example, in the following screen capture, the outline highlights the contact details for the account.



While continuing to press the left mouse button, you can move the mouse to highlight the correct field. When you release the mouse button, the highlighted field is captured.

If you failed to capture the field you wanted, click **Capture** again, **do not release the mouse**, and retake the capture.

Back in the Object Cloning window, the Object Details section shows the details of the **Account Number** field:

- Type: TextBox
- Path: 4|7|1|7

Property	Value
Type	TextBox
Path	4 7 1 7

- ___ e. In the Object Details section, click **Expand Search Criteria**.

Property	Value
Type	TextBox
Path	4 7 1 7

- ___ f. In the Search Criteria window, click the binoculars column box for the **Name** property to toggle the binoculars on.

Property	Value
Name	Account Number



Reminder

If the binoculars are on for a specific object property, it is included in the search criteria for that object. When the bot runs, it searches for a text box with the **Name** property **Account Number**.

- ___ g. In the Search Criteria window, click **OK** to close it.

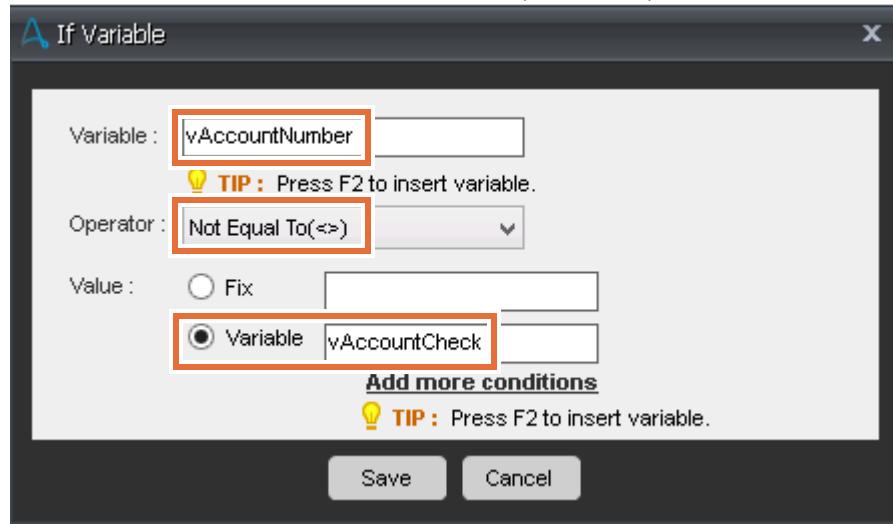
The **Name** property is now listed in the Object Details section.

Property	Value
Name	Account Number
Type	TextBox
Path	4 7 1 7

- ___ h. In the Select Action section, select **Get Property**, and define the following settings:
 - Select Property: **Value**
 - Assign To Variable: **vAccountCheck**
- ___ i. Click **Save**.

Property	Value
Name	Account Number
Type	TextBox
Path	4 7 1 7

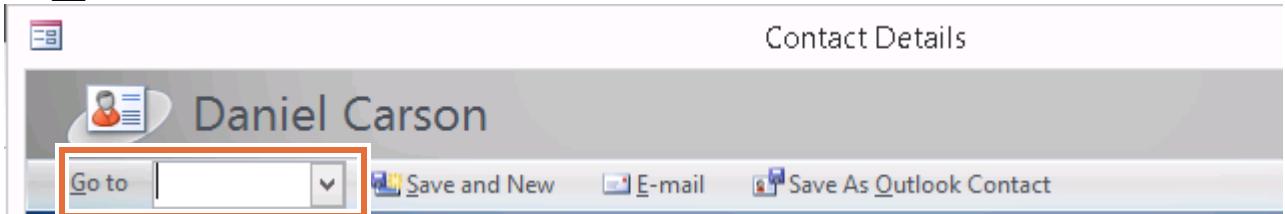
- ___ 4. Add an If command to check whether the value of vAccountNumber and the value of vAccountCheck do not match.
- ___ a. In the **Commands** list, go to the **If/Else** section, and double-click the **Variable** command.
 - ___ b. In the If window, click the **Edit** button to define the If condition.
 - In the If Variable window, define the following settings:
 - **Variable:** Press F2 (or Fn+F2) and insert **vAccountNumber**
 - **Operator:** Select **Not Equal To (<>)**
 - **Value:** Select **Variable**, and then press F2 (or Fn+F2) to insert **vAccountCheck**



- ___ c. Click **Save**.
- ___ 5. In the **If** statement, edit the default comment to the following statement:
If the value of vAccountCheck does not match the value of vAccountNumber,
search the application
- ___ 6. Use the Smart Recorder to capture the Contact Details window **Go to** field. Make sure that this command is inside the If statement.
 - ___ a. In the Workbench toolbar, click **Record**.
 - ___ b. In the Recorder window, select **Contact Details** for the window, and click **Start**.



- ___ c. In the Contact Details window, click the **Go to** field.



- ___ d. In the Recording window, click **Stop**.

- ___ 7. In the Workbench, look for the Object Cloning: Click on ComboBox "Go to" command and make sure that line is highlighted.

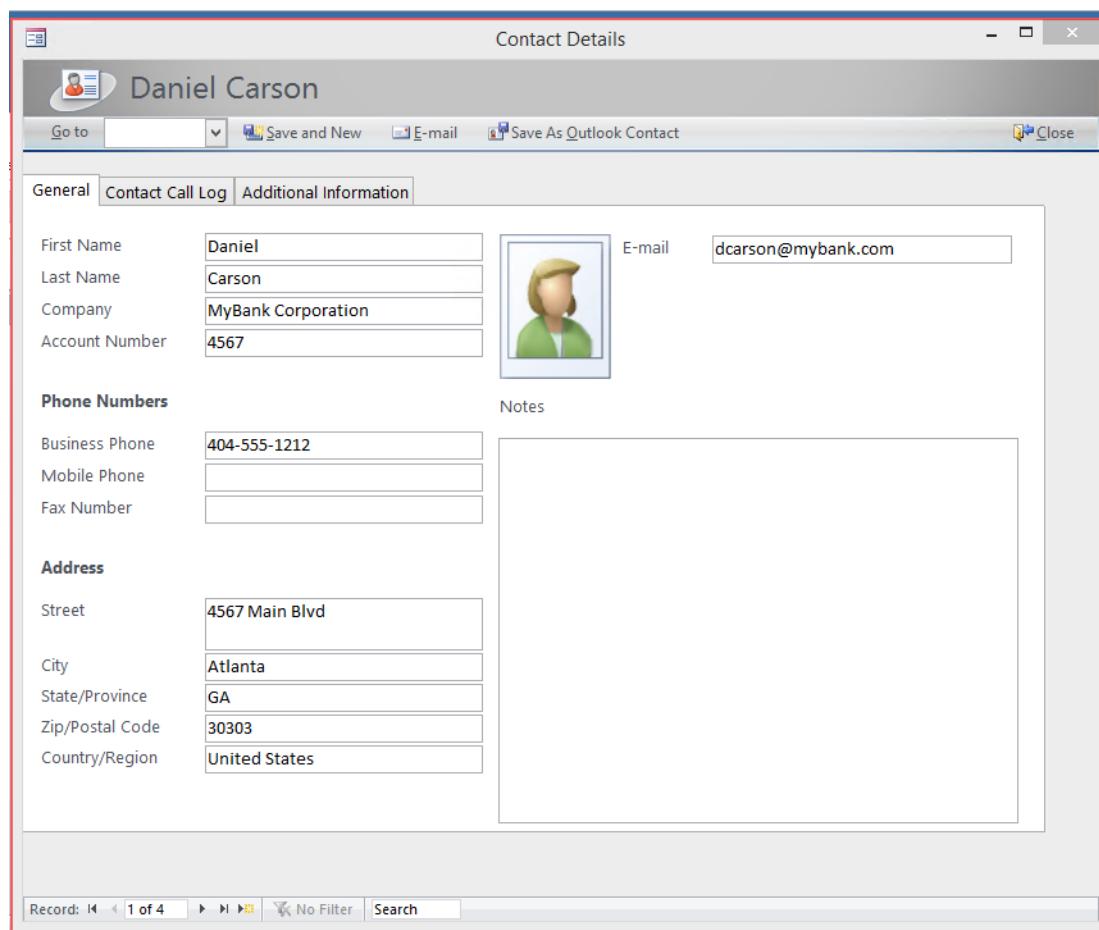
| 29 Object Cloning: Click On ComboBox "Go to" in window 'Contact Details'; Click Type: Left Click; Source: Window; Play 1

- ___ 8. Add an Object Cloning command that enters the vAccountNumber variable and presses Enter.

- ___ a. From the **Commands** list, double-click the **Object Cloning** command.

- ___ b. In the Object Cloning window, click **Select Window**, and select **Contact Details**.

- ___ c. Click **Capture**, continue pressing the left-click button, and drag the mouse so that the entire Contact Details window is highlighted.



- ___ d. When the entire Contact Details window is highlighted, release the left-click mouse button.

In the Object Cloning window, you should see that the Object Details section now includes information about the Contact Details window:

- Name: Contact Details
- Type: Client
- Path: 4

The screenshot shows the 'OBJECT' tab selected in the top navigation bar. Below it, the 'COORDINATES' and 'IMAGE' tabs are visible. The main area displays the 'Object Details:' section, which includes fields for 'Type: Client' and 'Technology: Standard'. Below this is a 'Select Search Criteria:' section with a search icon. A table below lists three items: 'Name' (Value: Contact Details), 'Type' (Value: Client), and 'Path' (Value: 4).

Property	Value
Name	Contact Details
Type	Client
Path	4



Note

It might take you a few tries before you capture the entire window. If you capture a part of the Contact Details window by mistake, click **Capture** and try again.

-
- ___ e. In the **Select Action** section, click **Select Action To Perform**, and select **Set Text**.
 - ___ f. In the **Text To Set** field, enter: \$vAccountNumber\$ [ENTER]
-

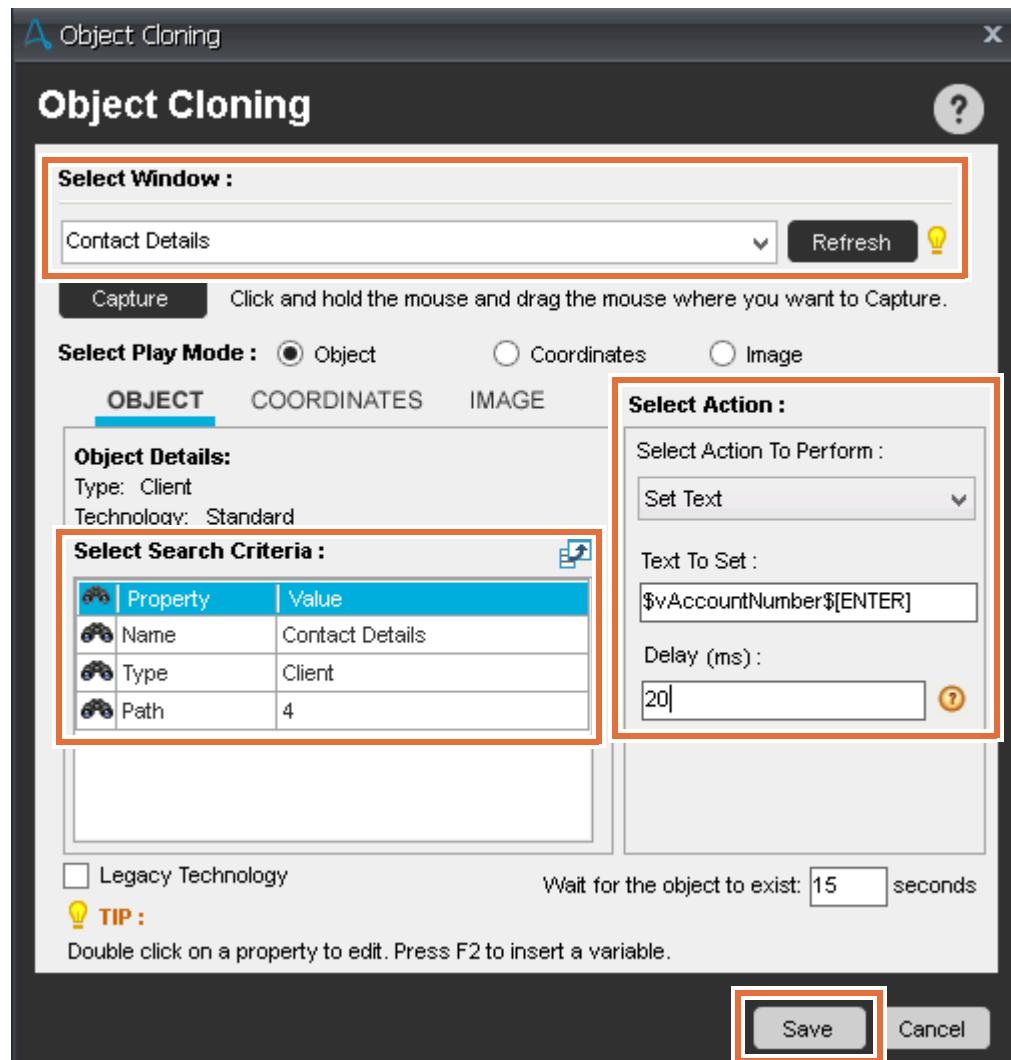


Information

You can press F2 (or Fn+F2) to insert the vAccountNumber variable. However, you must type [ENTER] directly into the field.

-
- ___ g. In the **Delay (ms)** field, enter: 20

- h. Click **Save**.



- i. Make sure that the Object Cloning: Set Text of Client "Contact Details" command is before the End If command.

```
38   Object Cloning: Set Text of Client "Contact Details" in window 'Contact Details'; Value: "$vAccountNumber${ENTER}
39   End If
```

- 9. Add the following comment after the End If command.

Copy the Company value and assign it to the vCompany2 variable

- 10. After the comment, add an **Object Cloning** command to copy the value in the Contact Details **Company** field.

**Note**

The previous Object Cloning command extracted the **Value** property from the field and assigned it to a variable.

In this step, the bot uses a different method to extract the value from the field: by mimicking the keyboard actions Ctrl+C to copy the value to the system Clipboard.

- ___ a. From the **Commands** list, add an Object Cloning command to the Workbench, and select the **Contact Details** window.
- ___ b. In the **Select Window** field, click **Contact Details**.
- ___ c. Click **Capture**, and while pressing the left-click mouse button, drag the mouse so that the **Company** field is highlighted.

The screenshot shows a software interface with a tab bar at the top containing 'General', 'Contact Call Log', and 'Additional Information'. Below the tabs, there are four input fields: 'First Name' (Daniel), 'Last Name' (Carson), 'Company' (MyBank Corporation), and 'Account Number' (4567). The 'Company' field is highlighted with a red border, indicating it is selected for capture.

- ___ d. When the **Company** field is highlighted, release the mouse button.

The Object Details section should have the following information:

- **Property:** Value
- **Type:** TextBox
- **Path:** 4|-7|1|5

The screenshot shows the 'OBJECT' tab selected in the Object Details panel. It displays the following information:

- Object Details:**
 - Type: TextBox
 - Technology: Standard
- Select Search Criteria:** A table with columns 'Property' and 'Value'.

Property	Value
Type	TextBox
Path	4 -7 1 5

- ___ e. In the Select Action section, select **Set Text**.
- ___ f. In the **Text To Set** field, enter: [CTRL DOWN]c[CTRL UP]



Information

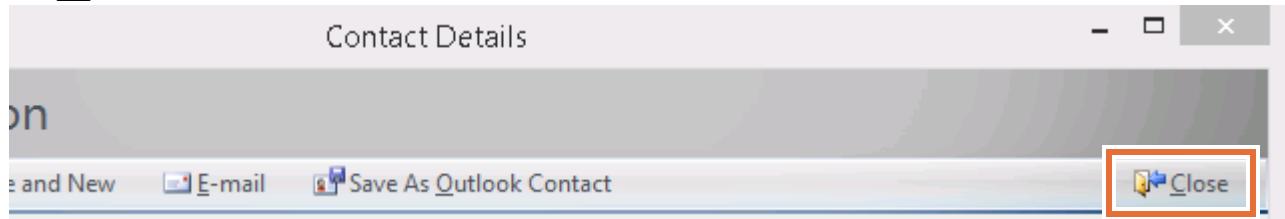
This value tells the bot to perform a Ctrl+C copy function by mimicking the actions of pressing Ctrl+C, and then releasing Ctrl.

- ___ g. In the **Delay (ms)** field, enter: 20
- ___ h. Click **Save**.

The Object Cloning: Set Text of TextBox "Company" command is now in the Workbench.

33 Object Cloning: Set Text of TextBox "Company" in window 'Contact Details'; Value:"[CTRL DOWN]c[CTRL UP]"; Source:

- ___ 11. Use the Smart Recorder to record the action of closing the Contact Details window.
 - ___ a. In the Workbench toolbar, click **Record**, select **Contact Details**, and click **Start**.
 - ___ b. In the Contact Details window, click **Close**.



- ___ c. In the Recording window, click **Stop**.

The Object Cloning: Click On PushButton "Close" command is now in the Workbench.

33 Object Cloning: Click On PushButton "Close" in window 'Contact Details'; Click Type: Left Click; Source: Window; Play T

- ___ 12. Add a Variable Operation command to assign the copied value from the Contact Details **Company** field to the **vCompany2** local variable.

You use the **\$Clipboard\$** system variable as part of the variable operation.

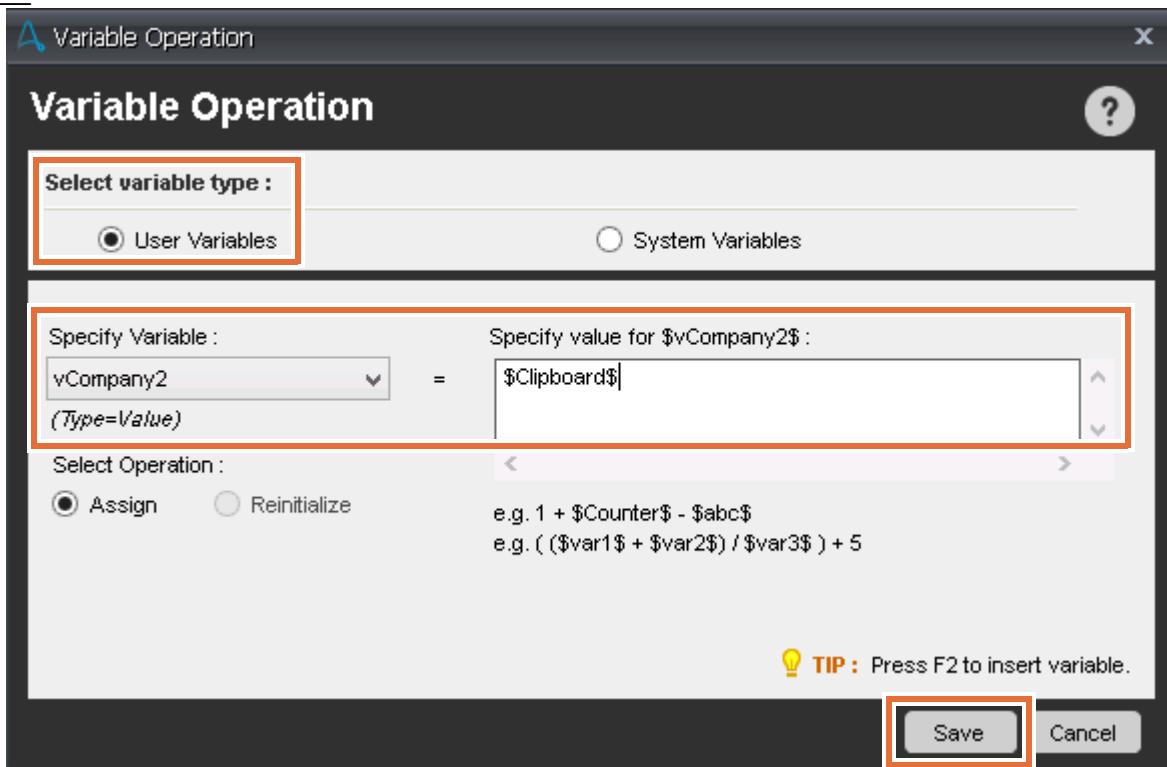
- ___ a. From the **Commands** list, add a Variable Operation command after the Object Cloning command that you added in the previous step.
- ___ b. In the Variable Operation window, enter the following settings:
 - Select variable type: **User Variables**
 - Specify Variable: **vCompany2**
 - Specify value for **\$vCompany2\$**: **\$Clipboard\$**



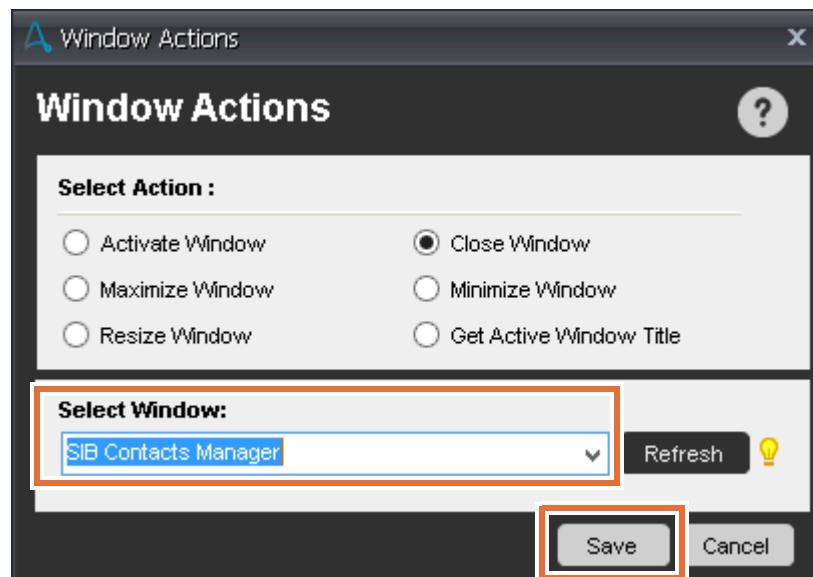
Note

You can use F2 (or Fn+F2) to insert **\$Clipboard\$** into the field. It is in the **\$System Variables\$** section.

- c. Click **Save**.



13. Add a command to close the SIB Contacts Manager Microsoft Access window.
- From the **Commands** list, go to the Windows Actions section, and add a Close Window command to the Workbench.
 - In the Window Actions window, select **SIB Contacts Manager**, and click **Save**.



14. Save your work.
15. Close the SIB Contacts Manager window.

3.3. Adding vCompany2 to the Message Box

Next, you add vCompany2 to the message that is displayed by the Message Box command.

- 1. Double-click the line of the code to edit the **Data Consistency Check Results** Message Box.



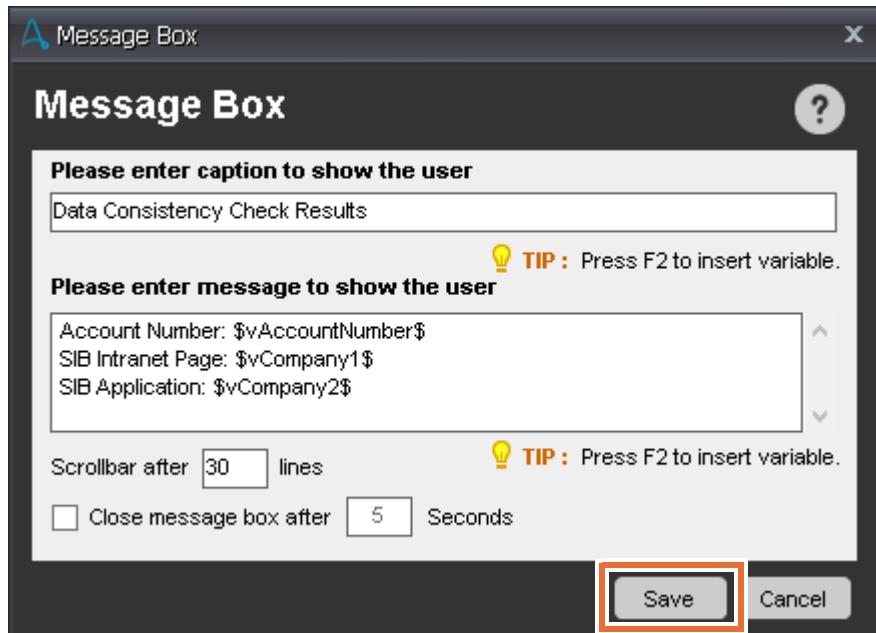
Note

If **Data Consistency Check Results** Message Box is not the last line, then move the line to the last line of the code.

- 2. In the Please enter message to show the user field, append the following line to the existing lines:

SIB Application: \$vCompany2\$

- 3. Click **Save**.



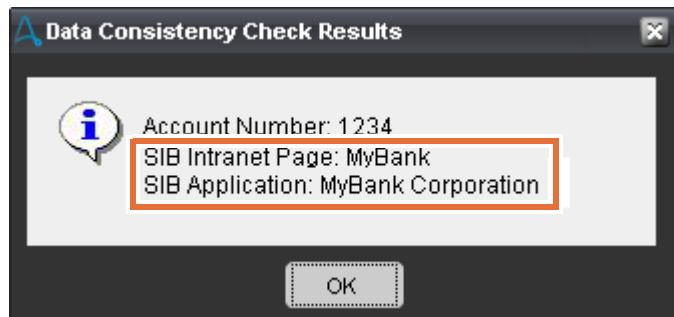
- 4. Save your work.

3.4. Testing the bot

Next, you run the bot with a valid account number. You expect the bot to find the account in two company data sources. The message should display both results.

- 1. On the toolbar, click **Run**.
- 2. In the **Enter the customer account number** field, enter 1234 and click **OK**.

- ___ 3. Confirm that the resulting message lists SIB Intranet Page and SIB Application.



- ___ 4. Click **OK** to close the message.

Section 4. Extracting the company value from an Excel spreadsheet

In this section, you search for the account number in an Excel spreadsheet, and extract its associated company value. You then assign the company value to the `vCompany3` variable.

4.1. Opening the Excel spreadsheet and searching for the account number

1. Add the following comment:

Source: Excel

2. Add another comment:

Open the Excel file and search for the account number

3. Add an Excel command to open the `mybank_accounts.xlsx` spreadsheet.

a. In the **Commands** list, expand the Excel section if needed, and add an `Open Spreadsheet` command to the Workbench.

b. In the Excel window, keep the **Session Name** at Default.

c. In the **Spreadsheet Path** field, enter the following path:

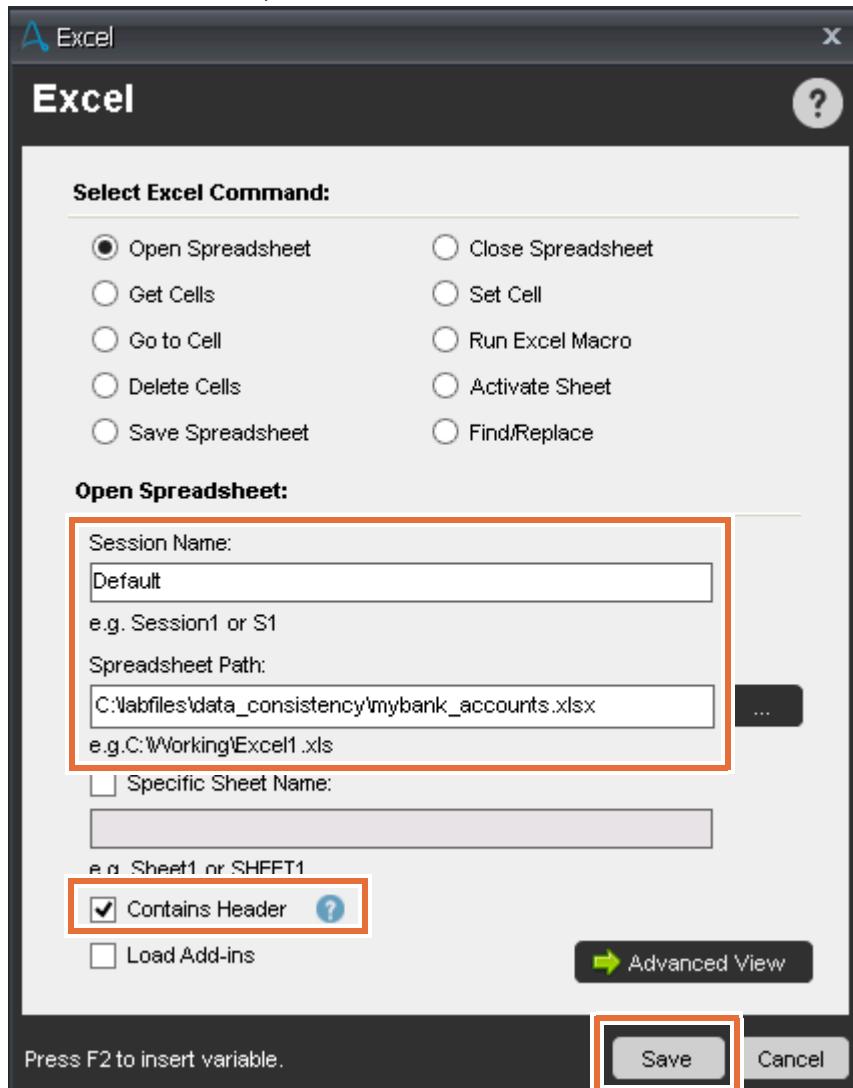
`C:\labfiles\data_consistency\mybank_accounts.xlsx`



Note

You can use the ellipses (...) button to browse to the directory and select the `mybank_accounts.xlsx` file.

- ___ d. Select **Contains Header**, and click **Save**.



- ___ 4. Add an Excel command to find the vAccountNumber variable value in the spreadsheet.

- ___ a. In the **Excel** section of the **Commands** list, add a **Find/Replace** command.
 ___ b. In the **Find/Replace** section of the Excel window, in the **Find** field, enter:
 \$vAccountNumber\$

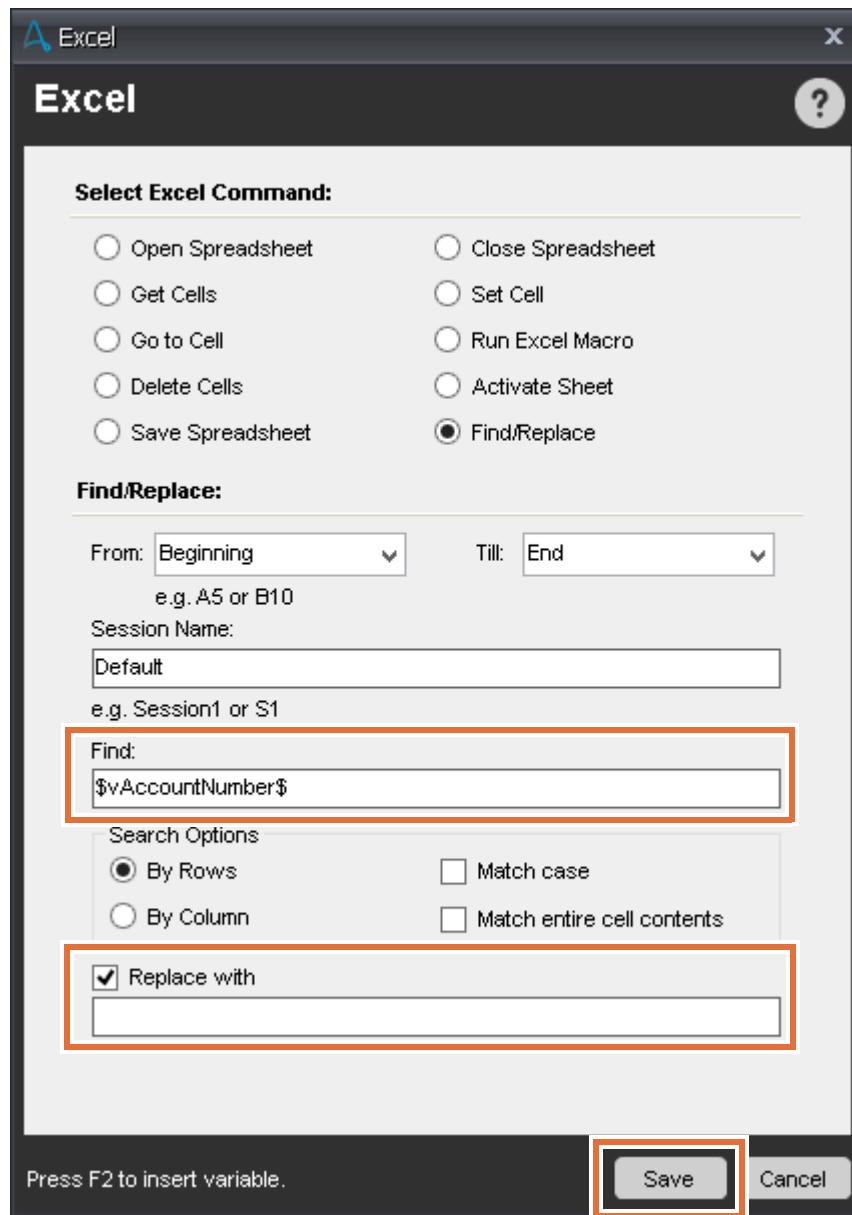


Reminder

You can press F2 (or Fn+F2) to insert the variable in the field.

- ___ c. Select **Replace with**, but leave the field empty.

- ___ d. Click Save.



4.2. Finding the company value and assigning it to the vCompany3 variable

In this section, you tell the bot to go to the cell in the row that contains the company data, retrieve the data from that cell, and assign it to the vCompany3 variable.

- ___ 1. Add the following comment to the Workbench:

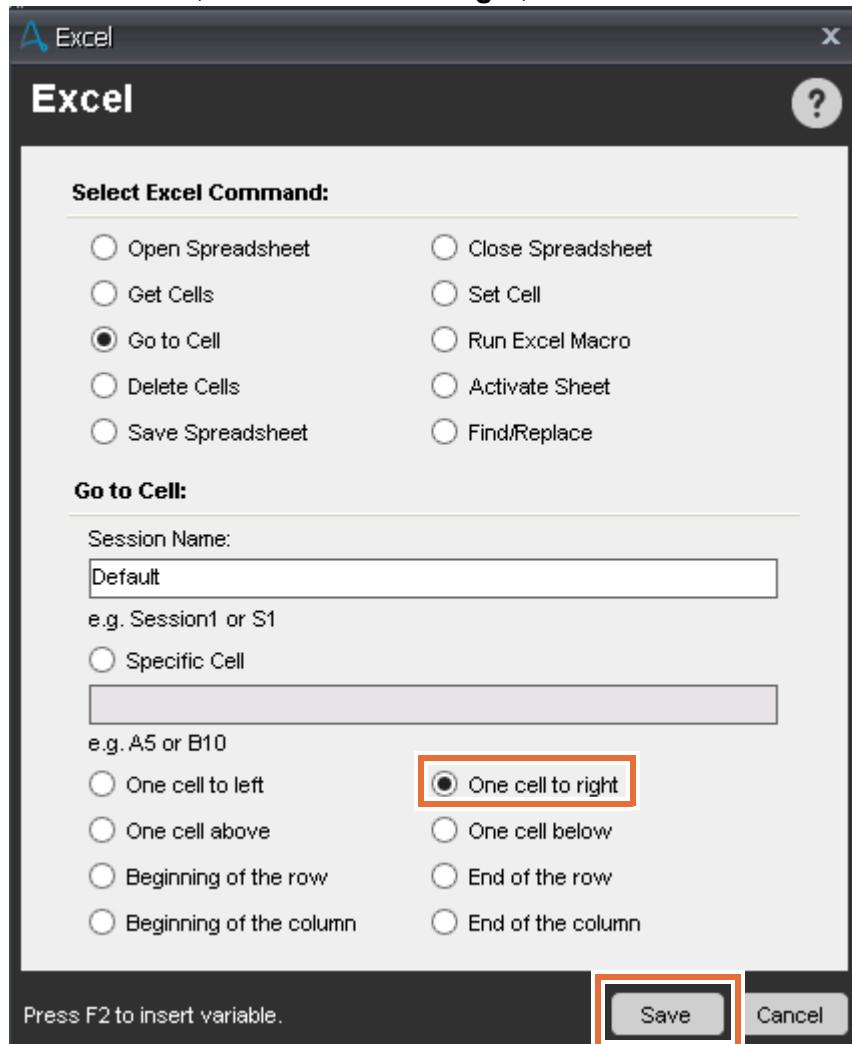
Find the company data for the account, and assign it to vCompany3

- ___ 2. Add an Excel command to move to the **Company** column of the row.

The **Company** column is located one column to the right of the **Account** column.

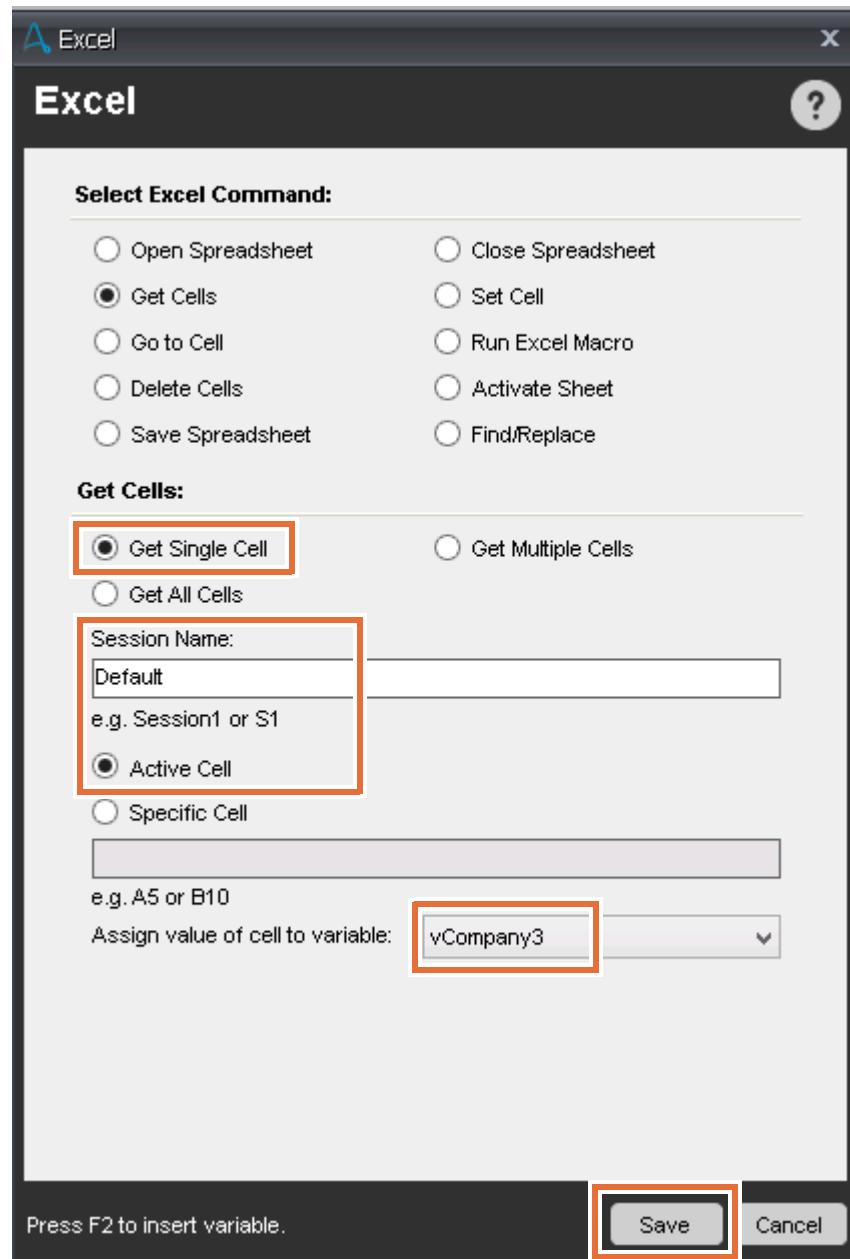
- ___ a. In the **Excel** section of the **Commands** list, add a Go to Cell command.

- ___ b. In the Excel window, select **One cell to right**, and click **Save**.



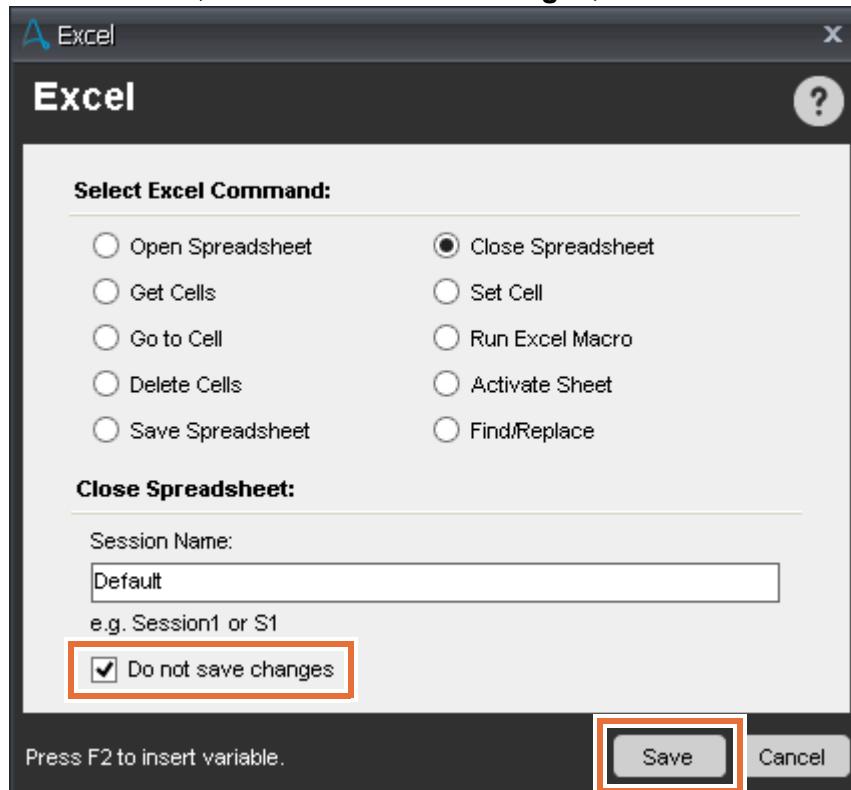
- ___ 3. Add a command to retrieve the value of the active cell with the company data, and assign it to vCompany3.
- ___ a. In the **Excel** section of the **Commands** list, add a **Get Cells** command.
- ___ b. In the Excel window, leave the following settings at their default value:
- **Get Single Cell**
 - **Session Name:** Default
 - **Active Cell**
- ___ c. For **Assign value of cell to variable**, select **vCompany3**.

- ___ d. Click **Save**.



- ___ 4. Add a command to close the Excel spreadsheet to the Workbench.
___ a. In the **Excel** section of the **Commands** list, add a **Close Spreadsheet** command.

- ___ b. In the Excel window, select **Do not save changes**, and click **Save**.



- ___ 5. Save your work.

4.3. Adding vCompany3 to Message Box

Next, you add vCompany3 to the message that is displayed by the `Message Box` command.

- ___ 1. Double-click the last line of the code to edit the Message Box.

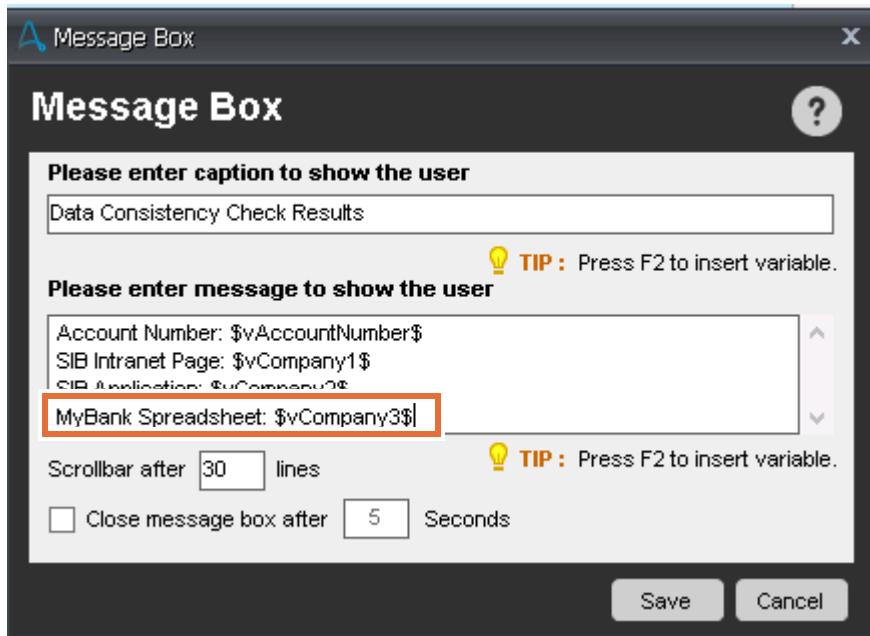


Note

If **Data Consistency Check Results** Message Box is not the last line, then move the line to the last line of the code.

- __ 2. In the **Please enter caption to show the user** field, append the following line:

MyBank Spreadsheet: \$vCompany3\$

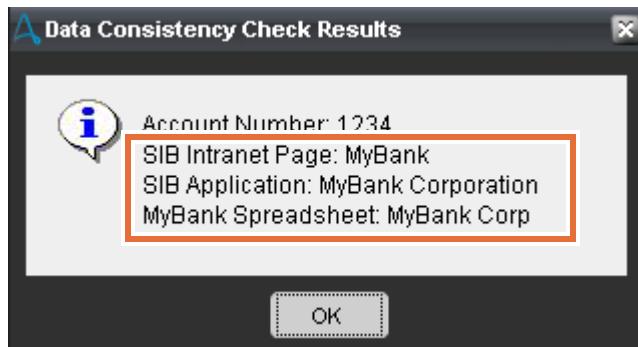


- __ 3. Click **Save**.
 __ 4. Save your work.

4.4. Testing the bot

Next, you run the bot to make sure that the bot correctly retrieves the company data from all three company data sources.

- __ 1. On the toolbar, click **Run**.
 __ 2. In the “Prompt in ‘Taskbar’” window, enter 1234 and click **OK**.
 __ 3. Confirm that the message lists the three company data sources.



- __ 4. Click **OK** to close the window.
 __ 5. *Optional.* If you want to retest the bot with other valid account numbers, try these numbers:
 - 2345
 - 3456
 - 4567

Section 5. *Optional.* Viewing the solution file

If you were unable to complete and run the bot successfully, you can open the solution file for this exercise to see how the bot should be coded.

- __ 1. Go to C:\labfiles\data_consistency\solution, right-click the Data Consistency solution.atmx file and click **Edit**.

The file opens in the Workbench.

- __ 2. To run the bot, click **Run** on the Workbench toolbar.



Information

If you want to load the solution file from the Enterprise Client, move the Data Consistency solution.atmx file to the following directory: C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks

The Enterprise Client **My Tasks** view automatically accesses files in that folder.

From the **My Tasks** list, you can either run the bot by double-clicking it or you can view it by right-clicking the task and clicking **Edit**.

End of exercise

Exercise review and wrap-up

In the first part of the exercise, you defined the variables and created a message box to prompt users to enter an account number. You then added commands to the bot to search the SIB Intranet customer list page for the account and retrieve the company value. Next, you added commands to the bot to open and log in to an Access-based Windows application, search for the account number, and retrieve the company value from the customer record. You then added commands to open and search an Excel spreadsheet for the account number, and to retrieve the associated company value from the spreadsheet row. Finally, you created a message box for the bot to display the three company data sources that were accessed by the bot search.

Exercise 8. Creating a login MetaBot

Estimated time

00:30

Overview

This exercise introduces you to MetaBots.

Objectives

After completing this exercise, you should be able to:

- Create a MetaBot to handle application login
- Reuse the login MetaBot in an existing bot

Introduction

In this exercise, you replace the login commands in the Data Consistency bot with a reusable MetaBot.

The exercise includes these sections:

- [Section 1, "Building the MetaBot"](#)
- [Section 2, "Replacing login commands with the login MetaBot"](#)

Requirements

You should complete [Exercise 7, "Creating an interactive bot to check values in disparate systems"](#) before you start this lab.

Section 1. Building the MetaBot

In this section, you create a MetaBot that logs in to the Customer Manager Access database.

1.1. Opening the “start” file for the bot

- ___ 1. In Windows Explorer, open the `C:\labfiles\metabot_login` directory.
- ___ 2. Right-click the `Data Consistency with Login.atmx` file, and click **Copy**.
- ___ 3. Paste the `Data Consistency with Login.atmx` file in the
`C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks` directory.
- ___ 4. Go to the Enterprise Client main window.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

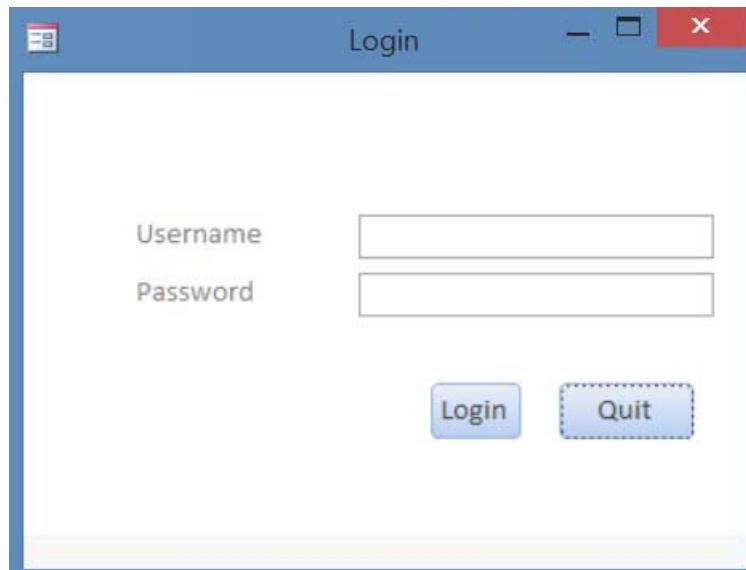
- ___ 5. Verify that `Data Consistency with Login.atmx` is listed in **My Tasks**.
- ___ 6. Right-click the `Data Consistency with Login.atmx` file, and click **Edit**.

The file opens in the Workbench.

1.2. Starting the Customer Manager Access database

- ___ 1. In Windows Explorer, open the `C:\labfiles\data_consistency` directory.
- ___ 2. Double-click the `Customer Manager.accdb` file.

The Login window opens. You can minimize the window, but you need to keep it open during this exercise.

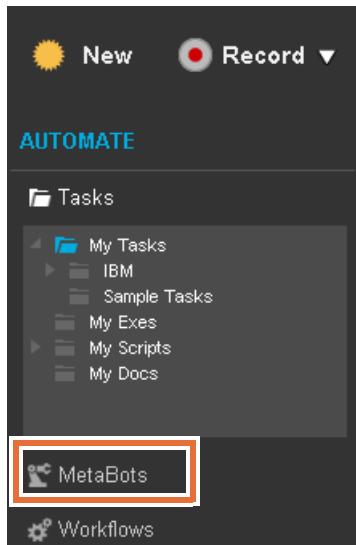


1.3. Writing the MetaBot code

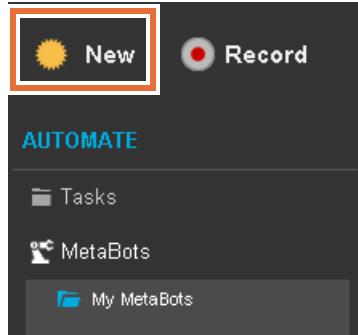
- 1. Go to the Enterprise Client main window to create the MetaBot.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

- 2. In the Automate pane to the left, click **MetaBots**.

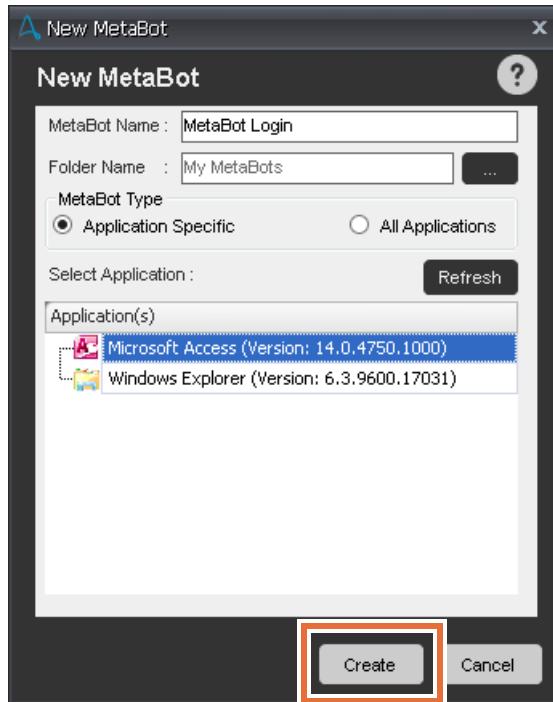


- 3. On the toolbar, click **New**.

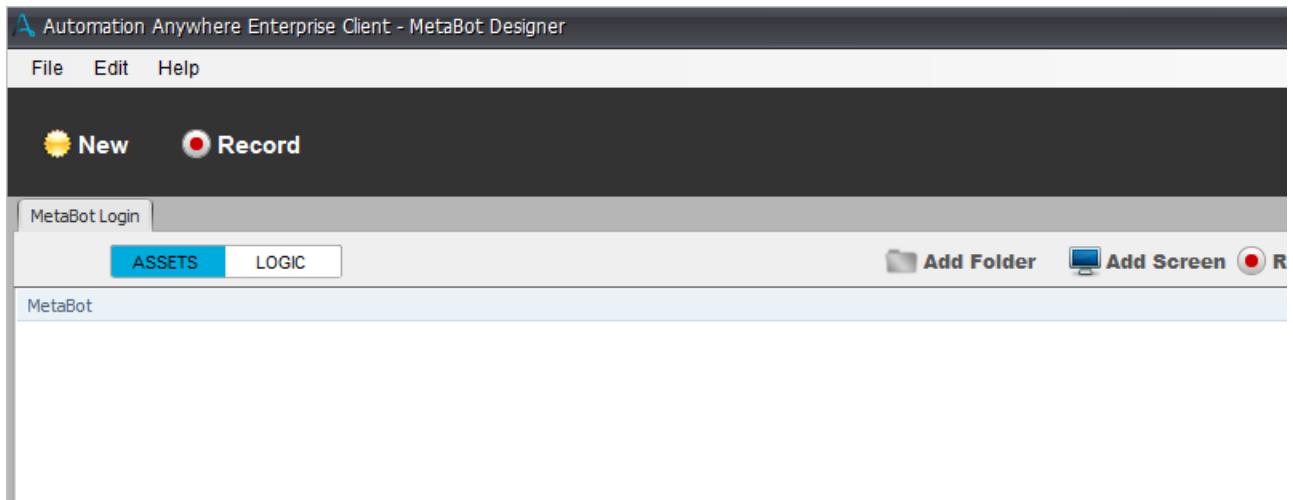


- a. In the **MetaBot Name** field, type: MetaBot Login
 — b. In the Application list, select **Microsoft Access**.

— c. Click **Create**.



The MetaBot Designer opens.



Information

When you choose the **Application Specific** option, you can create automation that uses screens from that application only. When you choose the **All Applications** option, you can create automation that uses a combination of screens from different applications.

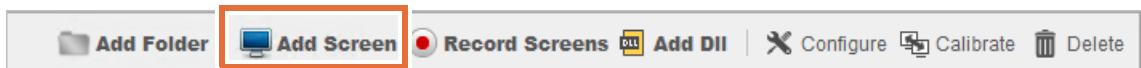


Information

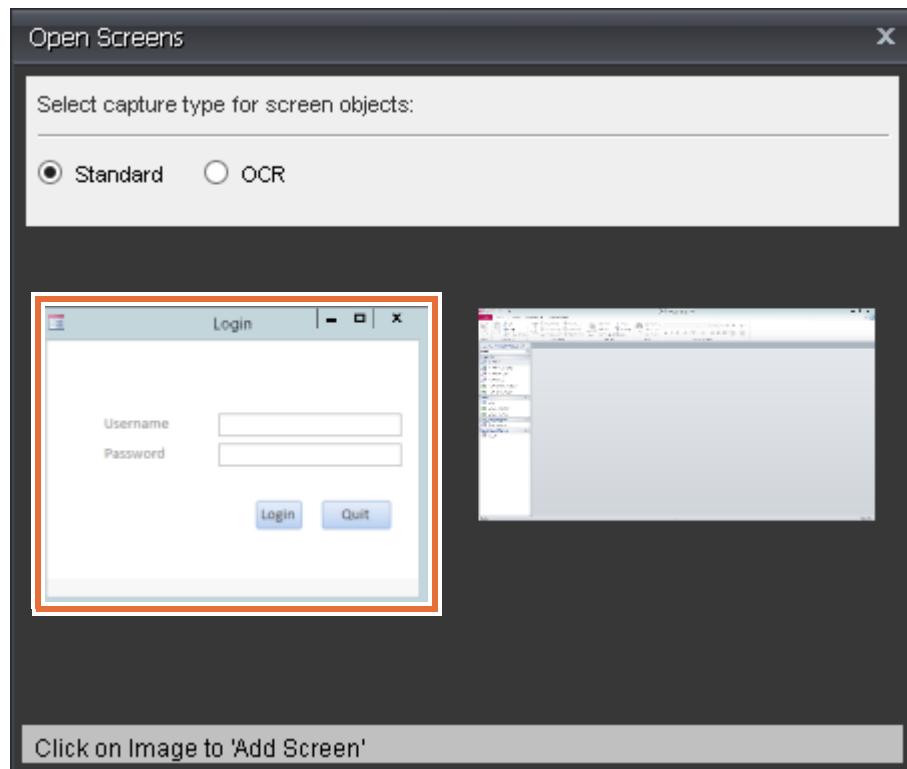
If you closed the Microsoft Access Login window, you can reopen it now (see [Section 1.2, "Starting the Customer Manager Access database"](#)), and click **Refresh** on the New MetaBot window. Your application appears for selection in the list.

4. Add the **Login screen** as an Asset to the Meta Bot.

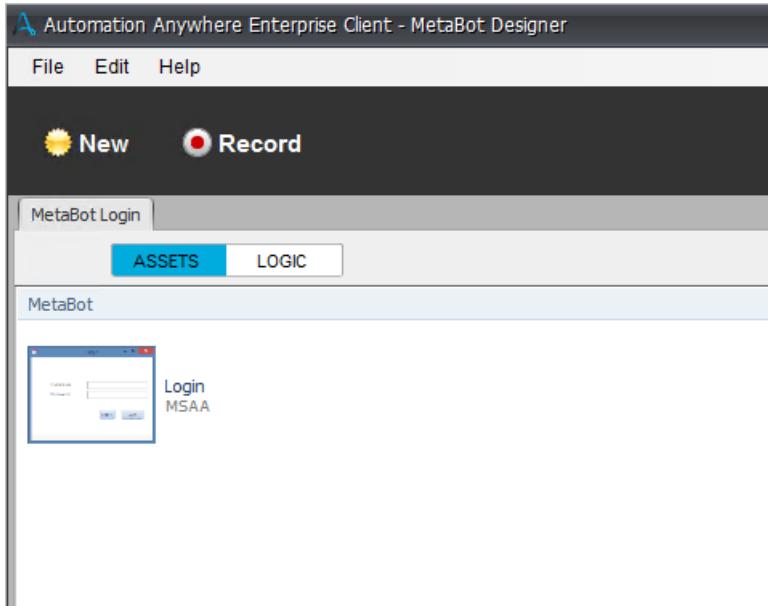
- a. On the toolbar, click **Add Screen**.



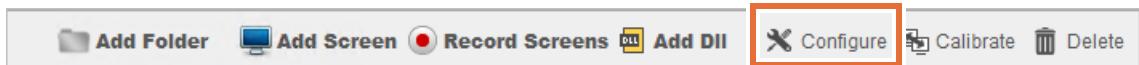
- b. Click the image on the left. (Note that the image on the left might appear blurry or distorted.)



The Login dialog box is added to the MetaBot designer.



- ___ c. In the MetaBot Designer, click the image of the **Login** window, and click **Configure**.



Information

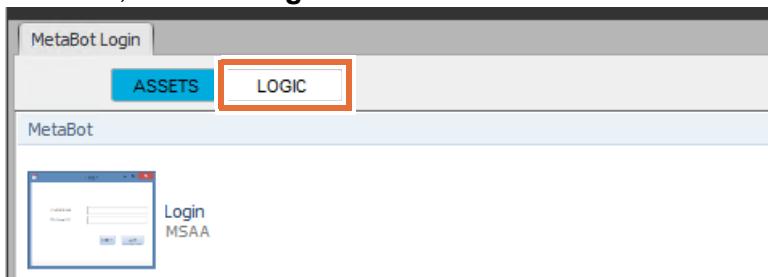
Use **Configure** to edit object properties for the screen.

Here, you can provide aliases such as **Screen Name**. You can also select an object to define its properties such as **Name**, **Path**, **Value**, **ID**, **Class**, and **Index**. For this simple MetaBot, you do not need to perform extra configurations.

- ___ d. After reviewing the screen and the **Object List**, click the **X** in the upper-right corner to close the window.

1.4. Defining the login logic for the MetaBot

- ___ 1. In the MetaBot editor, click the **Logic** tab.



- ___ 2. On the toolbar, click **Add Logic**.

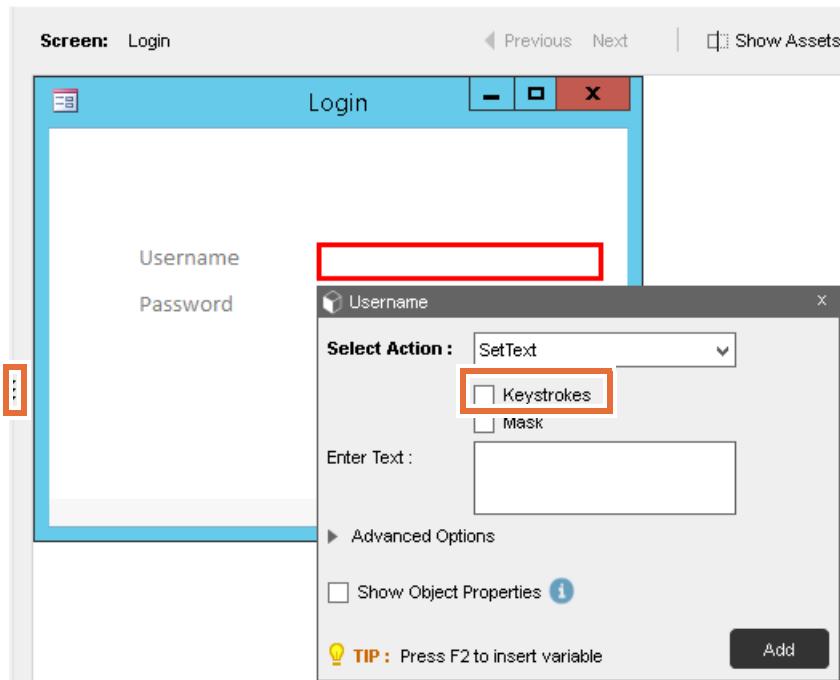




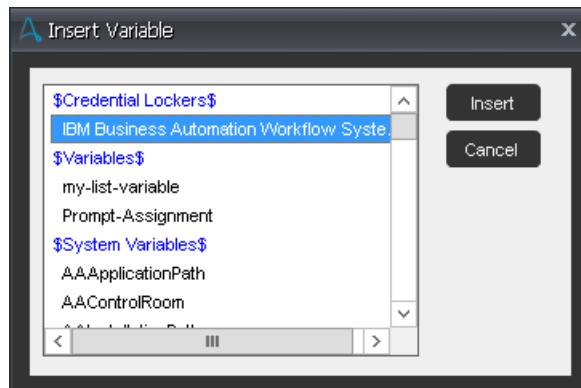
Information

The Add Logic feature opens the Workbench, which is where you define the navigational flow, add commands, define logic, and save and play the flow to verify behavior. If there is not enough space for the screen logic section, you can extend it by dragging the three dots.

- ___ 3. Click the field next to **Username**, and in the dialog box that opens, define the automation task for the **Username** field.
 - ___ a. Select **Keystrokes**.



- ___ b. In the **Enter Text** field, press the F2 (or Fn+F2) key.
- ___ c. In the **\$Credential Lockers\$** section of the Insert Variable dialog box, select **IBM Business Automation Workflow Systems** section, and click **Insert**.

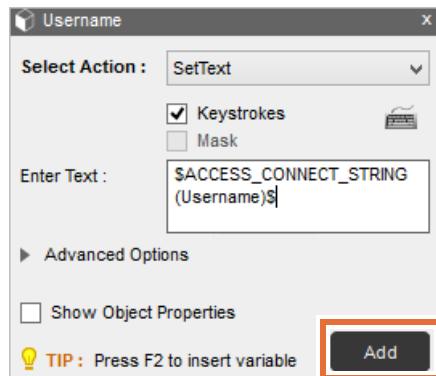


- ___ d. In the **Select Credential** dialog box, select **ACCESS_CONNECT_STRING** for the **Credential Name** and **Username** for the **Attribute Name**.

___ e. Click **OK**.



___ f. Back in the **Username** dialog box, click **Add**.



___ 4. Click the field next to **Password**, and define the automation for the Password field.

___ a. Select **Keystrokes**.

___ b. In the **Enter Text** field, press F2 (or Fn+F2).

___ c. In the **\$Credential Lockers\$** section of the Insert Variable dialog box, select **IBM Business Automation Workflow Systems** section, and click **Insert**.

___ d. In the **Select Credential** dialog box, select **ACCESS_CONNECT_STRING** for the **Credential Name** and **Password** for the **Attribute Name**.

___ e. Click **OK**.

___ f. Back in the **Password** dialog box, click **Add**.

___ 5. Add logic for the Login button.

___ a. Click the **Login** button.

___ b. In the Login screen, click **Add**.

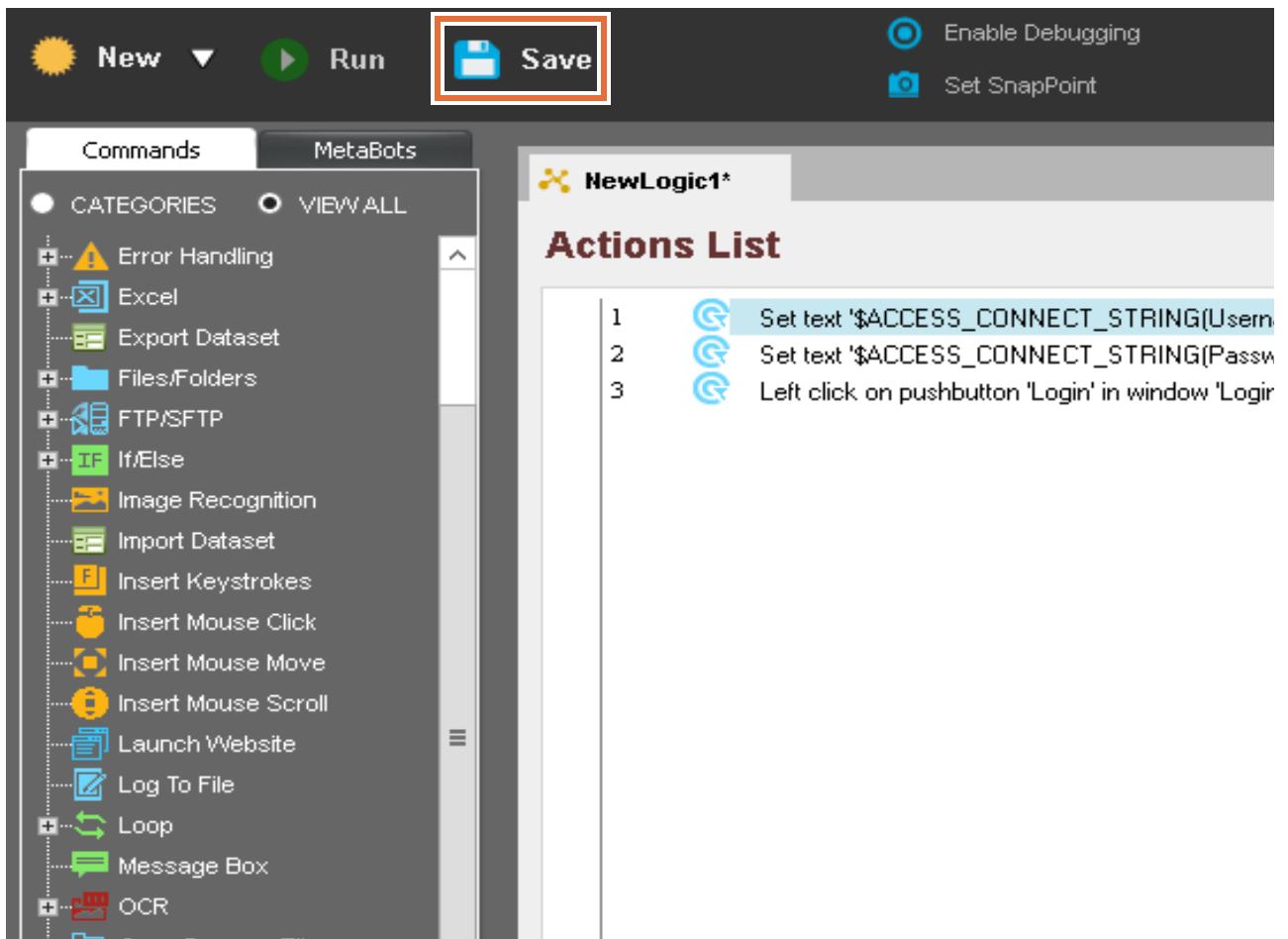
Your **Actions List** code should resemble the following:

Actions List

1		Set text '\$ACCESS_CONNECT_STRING(Username)\$' using keystrokes in textbox 'Username' in window 'Login'
2		Set text '\$ACCESS_CONNECT_STRING>Password)\$' using keystrokes in textbox 'Password' in window 'Login'
3		Left click on pushbutton 'Login' in window 'Login'

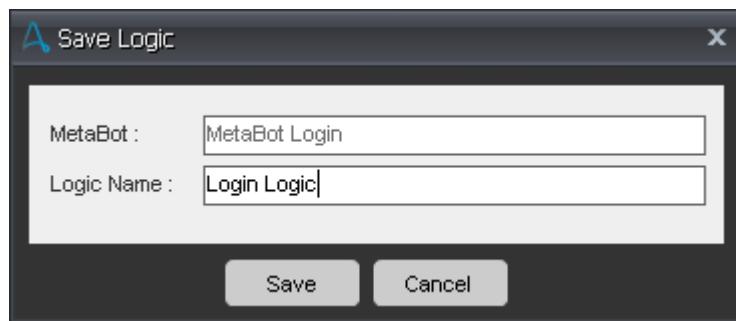
___ 6. Save your logic.

___ a. On the toolbar, click **Save**.



___ b. In the Name field of the Save Logic dialog box, type: Login Logic.

___ c. Click **Save**.



1.5. Testing the logic behavior

___ 1. On the toolbar, click **Run** to test the logic. The **Username** and **Password** fields are filled with the information taken from the **ACCESS_CONNECT_STRING** credential.

**Note**

The Login window that you opened in [Step 2](#) on page 8-2 must still be open.

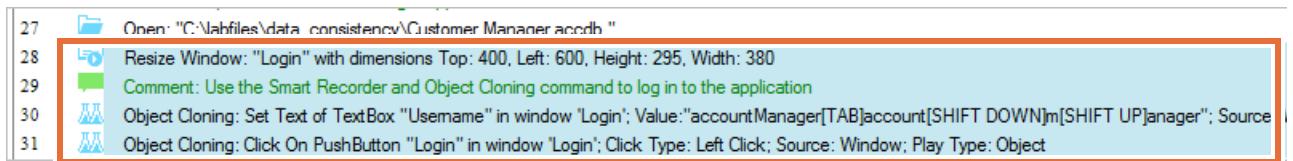
-
- ___ 2. After the test run, close the application windows.
 - ___ a. Close the Access database window by clicking the X in the upper-right corner.
 - ___ b. Close the Logic Editor and the MetaBot Designer windows.

Section 2. Replacing login commands with the login MetaBot

In this part of the exercise, you implement the login MetaBot that you created.

2.1. Disabling the login commands in the bot code

- ___ 1. Return to the Enterprise Client window and click **Tasks** on the Navigation bar to the left.
- ___ 2. Right-click the **Data Consistency with Login** bot and click **Edit**.
- ___ 3. Disable the lines of code in the bot that perform the login.
 - ___ a. Look for the command to open the Customer Manager.accdb file and highlight the lines after that are used to log in to the Access database.



```

27 Open: "C:\ahfiles\data_consistency\Customer Manager.accdb"
28 Resize Window: "Login" with dimensions Top: 400, Left: 600, Height: 295, Width: 380
29 Comment: Use the Smart Recorder and Object Cloning command to log in to the application
30 Object Cloning: Set Text of TextBox "Username" in window 'Login'; Value:"accountManager[TAB]account[SHIFT DOWN]m[SHIFT UP]anager"; Source: Window; Play Type: Object
31 Object Cloning: Click On PushButton "Login" in window 'Login'; Click Type: Left Click; Source: Window; Play Type: Object
  
```



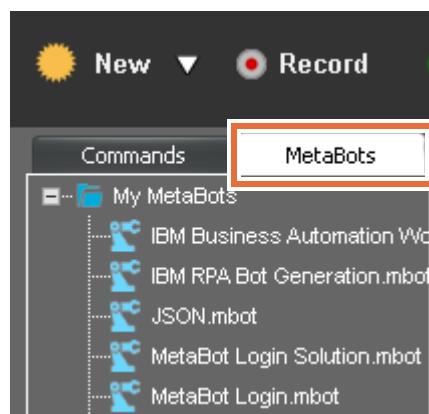
Hint

To highlight multiple lines, select the first line with your mouse, and press Shift+Down arrow to select more lines.

- ___ b. Right-click the highlighted lines, and click **Disable**.

2.2. Replacing the logic commands with the MetaBot

- ___ 1. Add your MetaBot to the Data Consistency bot.
 - ___ a. In the Command list, select the **MetaBots** tab.



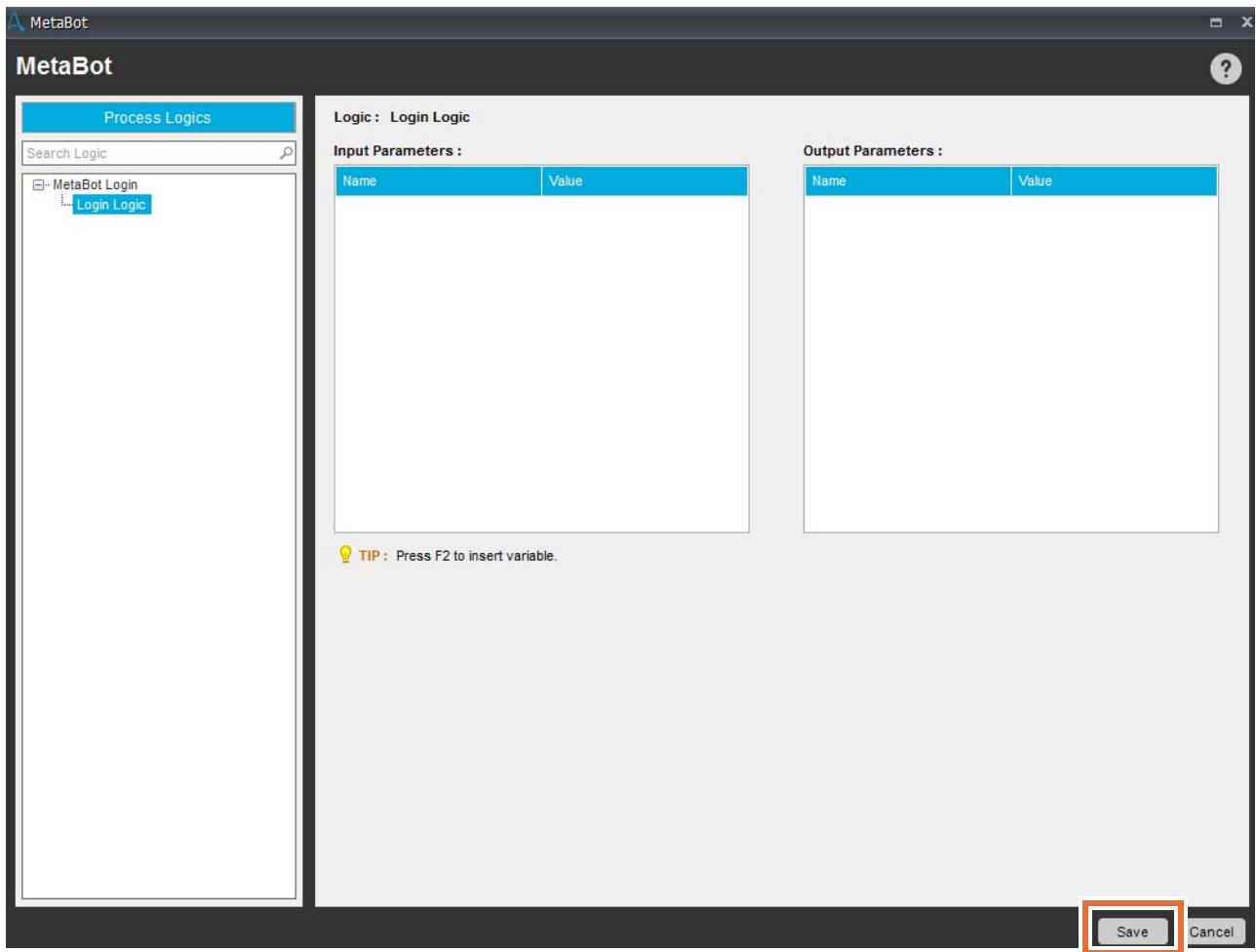
- ___ b. Drag the **MetaBot Login** bot from the MetaBots section on the left to the highlighted line of code that opens the Access database.

```

27  ⚡ Open: "C:\labfiles\data_consistency\Customer Manager.accdb"
28  ⚡ Resize Window: "Login" with dimensions Top: 400, Left: 600, Height: 295, Width: 380
29  📝 Comment: Use the Smart Recorder and Object Cloning command to log in to the application
30  🚀 Object Cloning: Set Text of TextBox "Username" in window 'Login'; Value: "accountManager"
31  🚀 Object Cloning: Click On PushButton "Login" in window 'Login'; Click Type: Left Click; Sourc
32  🚀 Run MetaBot "MetaBot Login.Login Logic" (Logic)

```

- ___ c. Click **Save** to close the MetaBot dialog box.



You should see that the MetaBot Login was added to your bot code.

```

27  ⚡ Open: "C:\labfiles\data_consistency\Customer Manager.accdb"
28  ⚡ Run Logic "Login Logic" from MetaBot "My MetaBots\MetaBot Login.mbot"
29  ⚡ Resize Window: "Login" with dimensions Top: 400, Left: 600, Height: 295, Width: 380
30  📝 Comment: Use the Smart Recorder and Object Cloning command to log in to the application
31  🚀 Object Cloning: Set Text of TextBox "Username" in window 'Login'; Value: "accountManager[TAB]account[SHIFT DOWN]"
32  🚀 Object Cloning: Click On PushButton "Login" in window 'Login'; Click Type: Left Click; Source: Window; Play Type: Obj

```



Information

You will not see the information passed to the Login MetaBot because it is using predefined credentials. If the MetaBot utilized field definitions instead of credentials, you would be asked to fill those in as part of this configuration.

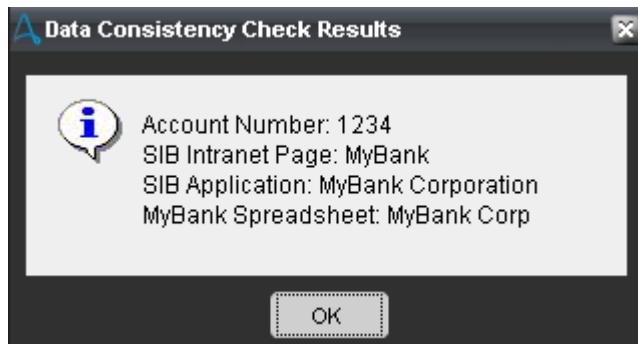
- ___ 2. Save your work.

2.3. Running the bot

When you run the original Data Consistency bot, it should run normally, but use the MetaBot for the login process.

- ___ 1. Run the bot, and search for account 1234.
 - ___ a. In the Workbench toolbar, click **Run**.
 - ___ b. In the “Prompt in ‘Taskbar’” window, enter 1234 as the account number, and click **OK**.

After the bot finishes, you should see the Data Consistency Check Results window. It displays the account number and the three different values for the company data that is associated with the account:



- ___ c. Click **OK** to close the window.
- ___ 2. *Optional.* Run the bot again with different account numbers. You can use valid or invalid account numbers to test the bot.

End of exercise

Exercise review and wrap-up

In this exercise, you created a MetaBot and defined the login logic for the Access database login screen. To implement it, you disabled the current commands that perform the login and replaced those commands with a call to the Metabot. To verify MetaBot functionality, the Data Consistency bot for which the MetaBot was created was run by using the newly implemented login logic of the MetaBot.

Exercise 9. Working with web services and error handling

Working with web services and error handling

Estimated time

02:30

Overview

This exercise demonstrates the use of advanced features, such as a REST web service calls and error handling.

Objectives

After completing this exercise, you should be able to:

- Implement basic error handling in a bot
- Work with a REST service to incorporate web service data in a bot
- Implement more complicated conditional (If/Else) logic in a bot

Introduction

This lab builds on the Trade Booking bot from [Exercise 6, "Creating a bot to evaluate data from a PDF and send an email"](#). It covers advanced features such as a REST web service call to obtain the price of the stock that is in the trade receipt, and it introduces error handling.

The REST service command uses Quandl, which is a free REST web service that provides after-trade-day prices. The stock price that is retrieved is used to calculate a monetary amount for the trade, which is included in the notification email that is sent by the bot. Although the stock price is one day late, the Quandl call serves the purpose of providing a REST-based interaction for this lab. As a free service, Quandl is not always available. The service occasionally returns an error that states that the anonymous user limit was reached. You configure the bot to handle the error in situations when the error is returned.

To test the REST service invocation, you can run the bot (double-click) called Rest Test from this folder:

C:\labfiles\trade_booking_advanced\solution

When prompted, enter ibm and click OK. You should receive a response similar to the following:

In addition, you implement basic error handling by using the Error Handling commands that are provided by the Workbench. The error handling that you implement includes a general way to handle bot errors when they occur, and handling two specific REST service errors.

The exercise instructions are broken down into the following sections:

- [Section 1, "Creating a copy of the Trade Booking bot and adding custom variables"](#)
- [Section 2, "Adding error handling to the Advanced Trade Booking bot"](#)
- [Section 3, "Working with REST web service calls"](#)
- [Section 4, "Evaluating whether the REST response has an error"](#)
- [Section 5, "Working with the Quandl stock price data"](#)
- [Section 6, "Evaluating the length of decimal values in the stock price"](#)
- [Section 7, "Evaluating decimal values and concatenating zeros when needed"](#)
- [Section 8, "Running the bot"](#)
- [Section 9, "Optional. Viewing the solution file"](#)

Requirements

Make sure that you complete [Exercise 6, "Creating a bot to evaluate data from a PDF and send an email"](#) before you work on this lab.

Section 1. Creating a copy of the Trade Booking bot and adding custom variables

In this section, you copy the Trade Booking bot that you created in [Exercise 6, "Creating a bot to evaluate data from a PDF and send an email"](#). Because this exercise extends the original Trade Booking bot, you work with this copy throughout this exercise. You also define local user variables that are used in this exercise, which you use along with the REST commands that you add to the bot.

1.1. Opening the “start” file for the bot

- ___ 1. In Windows Explorer, open the `C:\labfiles\trade_booking_advanced` directory.
- ___ 2. Right-click the `Trade Booking Advanced.atmx` file, and click **Copy**.
- ___ 3. Paste the `Trade Booking Advanced.atmx` file in the `C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks` directory.
- ___ 4. Go to the Enterprise Client main window.

If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.

- ___ 5. Verify that `Trade Booking Advanced.atmx` is listed in **My Tasks**.
- ___ 6. Right-click the `Trade Booking Advanced.atmx` file and click **Edit**.

The file opens in the Workbench.

1.2. Creating user variables

You need to add four variables to house data that is related to the REST call to the Quandl stock service. You add variables to store one calculated value (Trade Amount) and to evaluate the need for a decimal and trailing zeros.

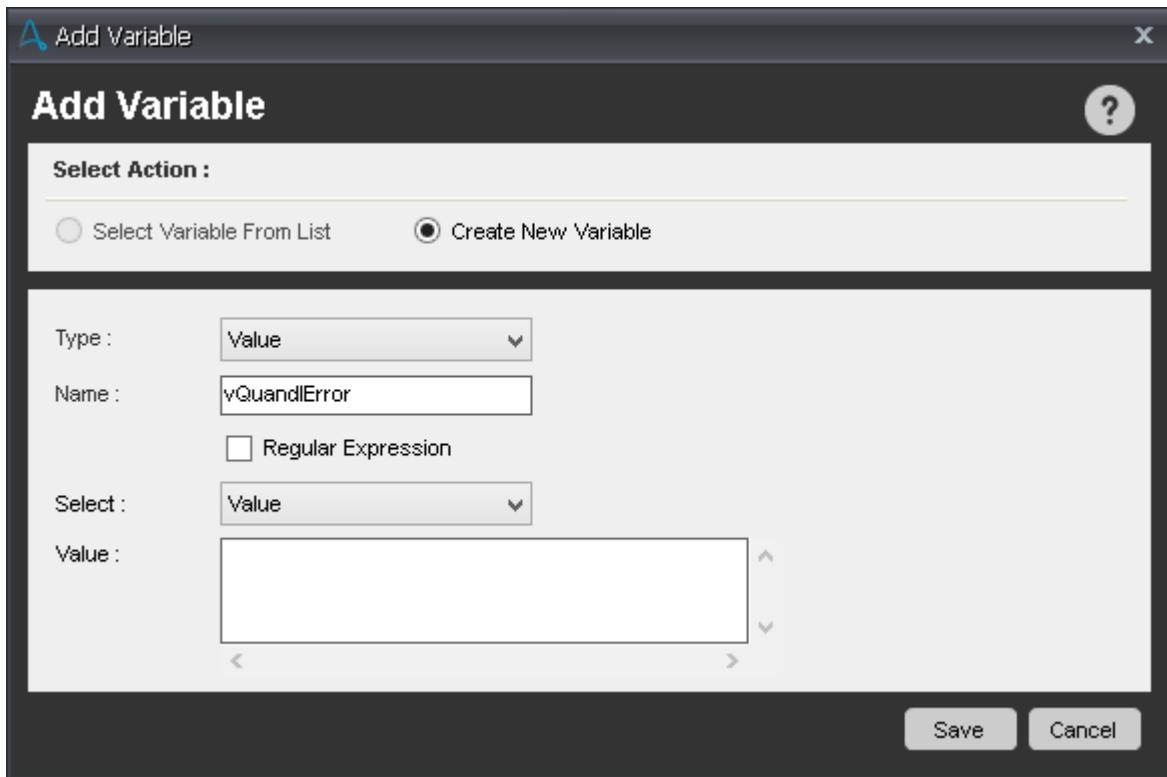


Information

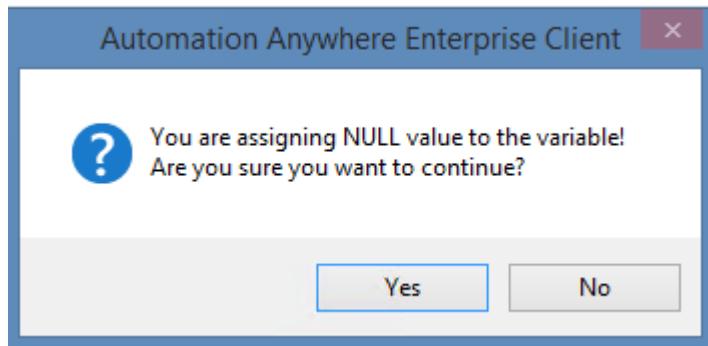
The IBM Robotic Process Automation with Automation Anywhere Enterprise Client automatically truncates the leading and trailing zeros in decimals. This behavior can cause issues when calculating prices, which typically include decimal values up to the 100ths position. Because of this behavior, you must define more variables and bot commands to correctly assign decimal values to the stock price.

- ___ 1. In the Workbench, open the Variable Manager and click **Add**.

- 2. In the **Name** field, enter `vQuandlError` and click **Save**.



- 3. When asked to confirm the null value for the variable, click **Yes**.



- 4. Repeat these steps to define the following variables.

Variable name	Description
<code>vQuoteDate</code>	Variable to hold the date of the stock quote
<code>vQuotePrice</code>	Variable to hold the price of the stock quote
<code>vStockReturn</code>	Variable to hold the return response from the Quandl REST service
<code>vTradeAmount</code>	Variable to hold the calculated value of the total stock price, based on the retrieved quote price
<code>vQuoteDecimal</code>	Variable to hold the decimal value of the stock quote price
<code>vQuoteDecimalLength</code>	Variable to determine the length of the stock quote price decimal value

Variable name	Description
vTradeAmountDecimal	Variable to hold the decimal value of the total amount of the trade price
vTradeAmountDecimalLength	Variable to determine the length of the total stock trade price decimal value

Section 2. Adding error handling to the Advanced Trade Booking bot

In this section, you add **Error Handling** commands to the Advanced Trade Booking bot, which can be used to configure automated responses to errors that occur when the bot runs. To apply error handling globally to the entire bot, you must enclose the entire list of bot commands within the **Begin Error Handling** and **End Error Handling** commands.

- 1. Add global error handling to the bot.
 - a. In the Commands list, expand the **Error Handling** section.
 - b. From the Error Handling section of the Commands list, drag the **Begin Error Handling** subcommand to the first line in the Actions List.
-



Information

When you add the **Begin Error Handling** subcommand to the Actions List, it also adds the **End Error Handling** command.

Both error handling commands are placed after the command that is highlighted when you drag the commands to the Actions List. You move the **Begin Error Handling** and **End Error Handling** commands to their correct locations after you configure the error handling actions.

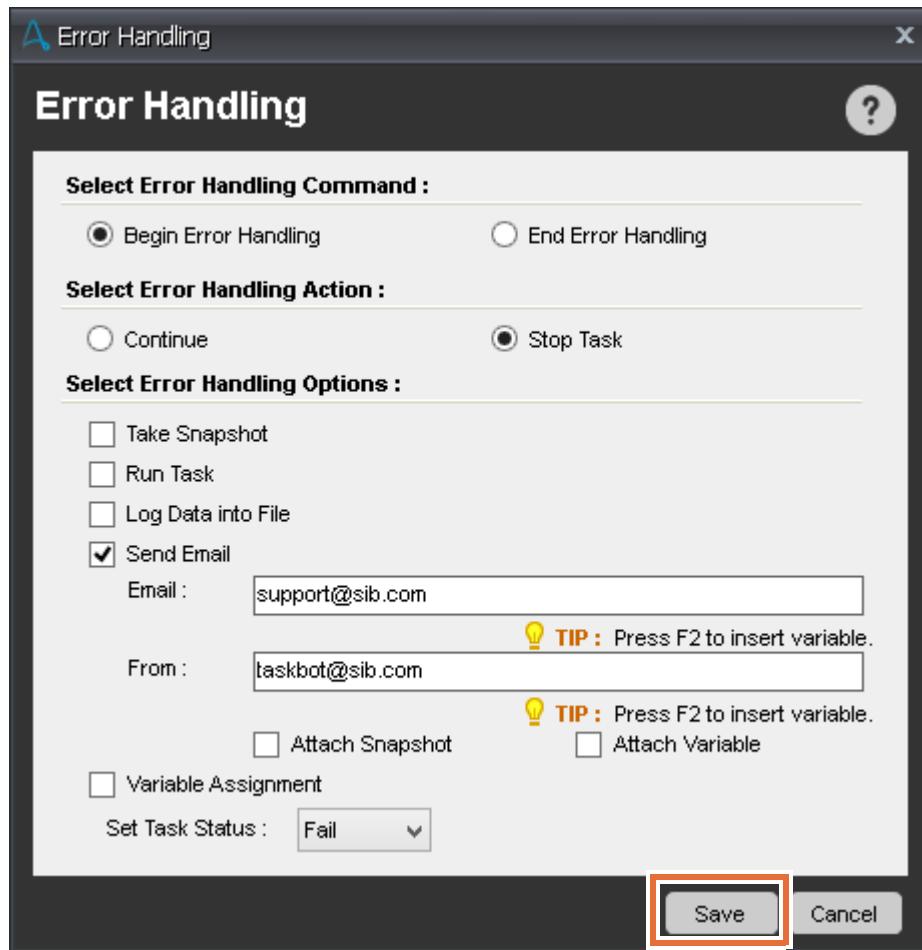
- 2. Configure the **Begin Error Handling** command to stop the task and send a notification email.
 - a. In the Select Error Handling Action section, select **Stop Task**.
 - b. In the Select Error Handling Options section, select **Send Email**.
 - c. In the **Email** field, enter: support@sib.com
The **Email** field is for defining the recipient of the error notification.
 - d. In the **From** field, enter: taskbot@sib.com
-



Information

The outgoing email settings must be defined in the Enterprise Client before you can use automated email commands in bots.

- ___ e. Click **Save**.

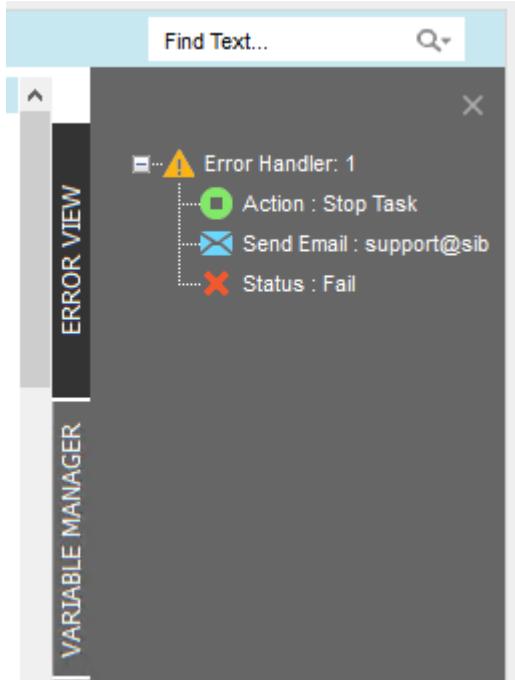


- ___ 3. Enclose the Advanced Trade Booking bot code within the Error Handling commands.
- ___ a. Drag the first **Comment** command and place it after the **Begin Error Handling** command.
- ___ b. Drag the **End Error Handling** command to the last line of the bot.
- The Advanced Trade Booking bot is now covered by the Error Handling command. If the bot encounters an error while it runs, it stops running and sends a notification email to the support@sib.com email address.



Information

When you add Error Handling commands to the Actions List, the **ERROR VIEW** vertical tab populates with information about the error handling that you implemented. You can expand the **Error Handler** nodes to view details about the error handling implementation in your bot.



- ___ 4. Delete the added Comment command that was placed with the Begin Error Handling command.
- ___ 5. Save your work.

```

1  ⚠️ Begin Error Handling; Action: Stop Task; Options: Send Email, Task Status: Fail
2  ➔ Comment: =====
3  ➔ Comment: Pull Trade Receipts from email, save to local folder, and extract values from PDF.
4  ➔ Comment: =====
5  ➔ Comment: Loop through emails extracting attachments from the TradeReceiptsDL mailbox.
6  ↗ Start Loop "Each message on server: localhost, User Name: tradeList@sib.com, ServerType:
7  ↘ End Loop
8  ➔ Comment: Extract custom-designed Form Fields from PDF.

41 ➔ Comment: =====
42 ➔ Comment: Cleanup: Delete all emails and files in folder.
43 ➔ Comment: =====
44 ➔ 🗑 Delete All Messages on server: 'localhost'
45 ➔ 📁 Delete Files "C:\labfiles\trade_booking\Trade Receipt MuBank.pdf"
46 ⚠️ End Error Handling

```

Section 3. Working with REST web service calls

In this part of the exercise, you configure the bot to work with a REST web service call.



Reminder

The original Trade Booking If/Else statement handles two conditions:

1. If condition: The trade receipt is missing any of the required information.

The bot determines whether the information is missing by calculating the length of the data values that are extracted from the trade receipt PDF fields. If the length of any of the data strings is 0, this condition is triggered.

In this case, the bot sends an email to the Trade Reporter.

2. Else condition: The trade receipt has all of the required information.

If the length of the data strings that are extracted from the trade receipt PDF fields are greater than 0, this condition is triggered.

In this case, the bot sends an email to the Trade Reporter Supervisor.

In both cases, the bot sends emails that include the extracted information from the trade receipt, whether the data is valid or invalid.

3.1. Understanding the REST Web Service command

In this section, you add a REST Web Service command and several associated commands to the original Else statement in the bot.

- The new REST command issues a GET request to the Quandl service.
 - It passes the value of `vStockSymbol` in the request as the stock symbol to query.
 - It also assigns the request response to the `vStockReturn` variable.
- You then add a Variable Operation command and several String Operation commands to work with the JSON objects that are returned by the GET request.

The following examples are in pseudo-code, and compare the current logic to what you implement in this lab.

Current If/Else Statement:

```

If any field length = 0
  Comment: Missing data
  Send email to Trade Reporter

Else
  Comment: All data present
  Send email to Trade Reporter Supervisor
End If
  
```

New If/Else Statement with REST call:

The original logic is represented by the first three lines and final three lines of this pseudo-code.

```

If any field length = 0
    Comment: Missing data
    Send email to Trade Reporter
Else
    Perform REST Web Service call
    Assign vStockReturn value to vQuandlError
    Delete spaces in vQuandlError value
    Extract substring error code from vQuandlError and assign output to vQuandlError
    Extract extra characters from vStockReturn

    Comment: Evaluate response from Quandl service for an error.
    If the Quandl error = QECx02 or QELx01
        Comment: REST Error
        Send an email to support
    Else
        Comment: Extract stock price and date
        Extract substring from vStockReturn and assign output to vQuotePrice
        Extract substring from vStockReturn and assign output to vQuoteDate

        Comment: Calculate Trade Amount
        vTradeAmount = vQuotePrice * vShares

        Comment: Extract values to evaluate decimal length
        Assign values after decimal to vQuoteDecimal
        Assign Length of vQuoteDecimal to vQuoteDecimalLength
        Assign values after decimal to vTradeAmountDecimal
        Assign Length of vTradeAmountDecimal to vTradeAmountDecimalLength

        Comment: Evaluate need for decimal and concatenate as necessary.
        If vQuoteDecimalLength = 1
            Concatenate 0 to vQuotePrice
        Else If vQuoteDecimalLength = 0 or > 2
            Concatenate .00 to vQuotePrice
        End If
        If vTradeAmountDecimalLength = 1
            Concatenate 0 to vTradeAmount
        Else If vTradeAmountDecimalLength = 0 or > 2
            Concatenate .00 to vTradeAmount
        End If
    End If

    Comment: All data present
    Send email to Trade Reporter Supervisor
End If

```

**Note**

Recall that the Quandl service is a free service and is not always available. To verify functioning of the service, you can run a Task Bot that performs a REST call to the service.

To test the REST service invocation, you can run the bot (double-click) called **Rest Test** from this folder:

C:\labfiles\trade_booking_advanced\solution

When prompted, enter `ibm` and click **OK**. You should receive a response similar to the following:

```
{ "dataset_data": { "limit": 1, "transform": null, "column_index": 4, "column_names": [ "Date", "Close" ], "start_date": "1962-01-02", "end_date": "2018-03-27", "frequency": "daily", "data": [ [ "2018-03-27", 151.91 ] ], "collapse": null, "order": null } }
```

Although the date and price will be different, if you get a response similar to the one above, the service is working correctly.

If you get a message similar to the following:

```
{ "quandl_error": { "code": "QELx01", "message": "You have exceeded the anonymous user limit of 50 calls per day. To make more calls today, please register for a free Quandl account and then include your API key with your requests." } }
```

the service is not working as expected. However, the bot is developed to handle this error so even if you run into service issues, you can run the bot and handle that error specifically.

3.2. Adding the REST Web Service command

In this section, you update the main **IF** condition (which determines whether information is missing from the trade receipt PDF) to refer to the new Quandl error scenario.

__ 1. Update the comment for the main **IF** condition.

__ a. Double-click the following **Comment** command in the **Trade Booking** bot:

Evaluate data and send email based on one of **two** situations.

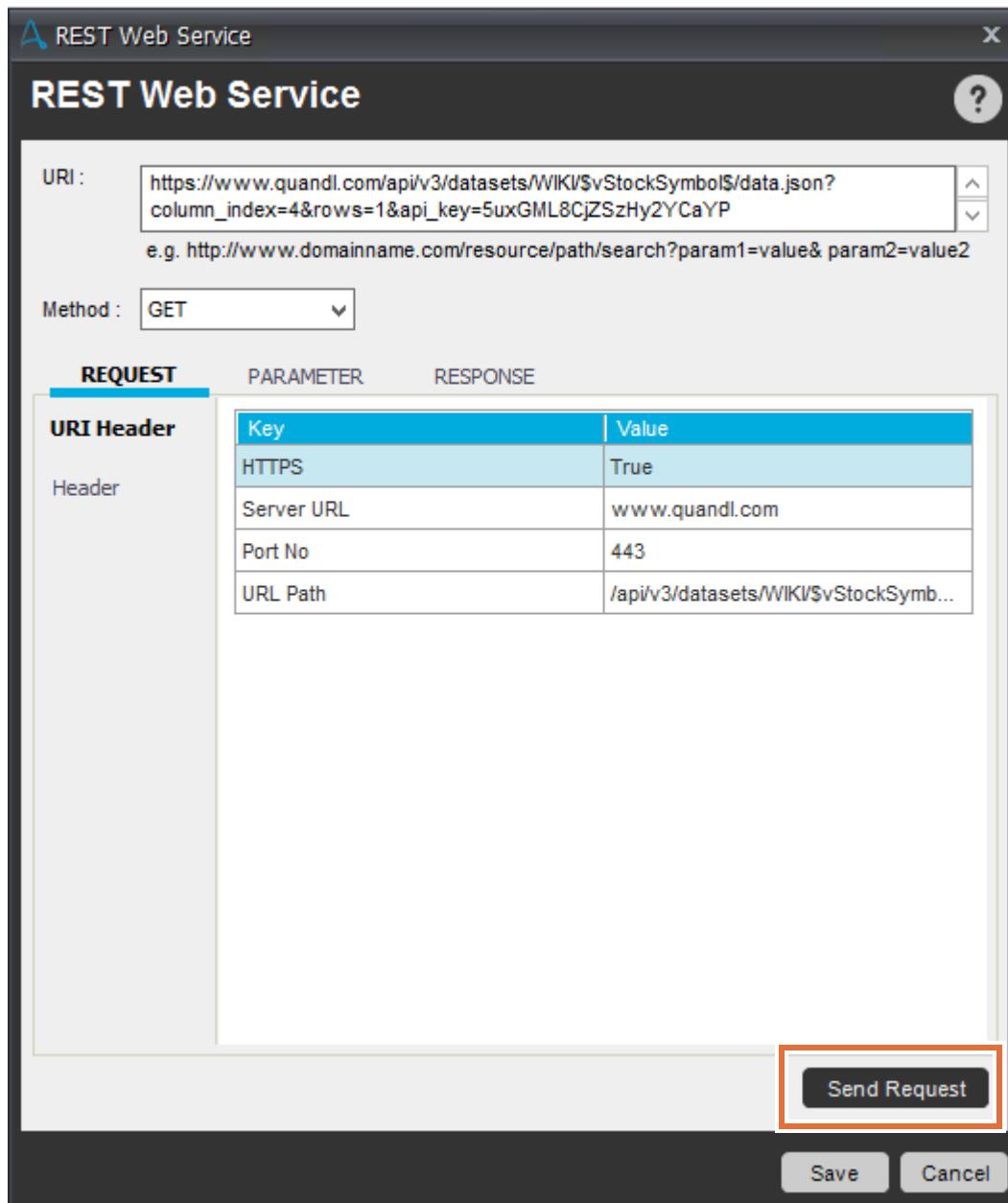
```

31  Comment: =====
32  Comment: Evaluate data and send email based on one of two situations.
33  Comment: =====
34  Comment: Evaluate length of each variable.
35  IF $vLengthParty1$ Equal To (=) "0" OR $vLengthParty2$ Equal To (=) "0" OR
36      Comment: Scenario: Missing Data
37      Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
38  ELSE
39      Send Email: Subject "Trade: $vTradeID$ Details"
40  End If

```

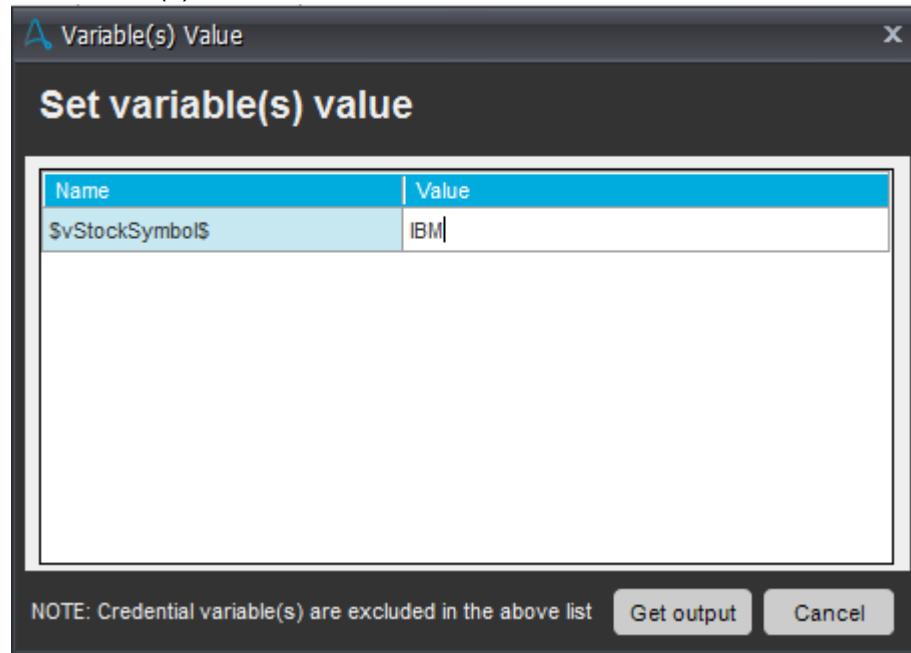
__ b. In the comment, change **two** to **three**.

- __ 2. Configure the REST Web Service command to query Quandl, and use IBM as the argument for the \$vStockSymbol parameter.
- __ a. From the Commands list, drag the **REST Web Service** command to the Actions List, and place it after the **Else** command.
 - __ b. In the **URI** field, enter the following path:
`https://www.quandl.com/api/v3/datasets/WIKI/$vStockSymbol$/data.json?column_index=4&rows=1&api_key=5uxGML8CjZSzHy2YCaYP`
 - __ c. Click **Send Request**.



Because the request passes a parameter (\$vStockSymbol\$), you must enter a value for the parameter.

- ___ d. In the Variable(s) Value window, in the **Value** field, enter IBM and click **Get output**.



In the REST Web Service, the **RESPONSE** tab is shown with details about the response. The **REQUEST** and **PARAMETER** tabs are also populated.

The screenshot shows the REST Web Service interface with the following details:

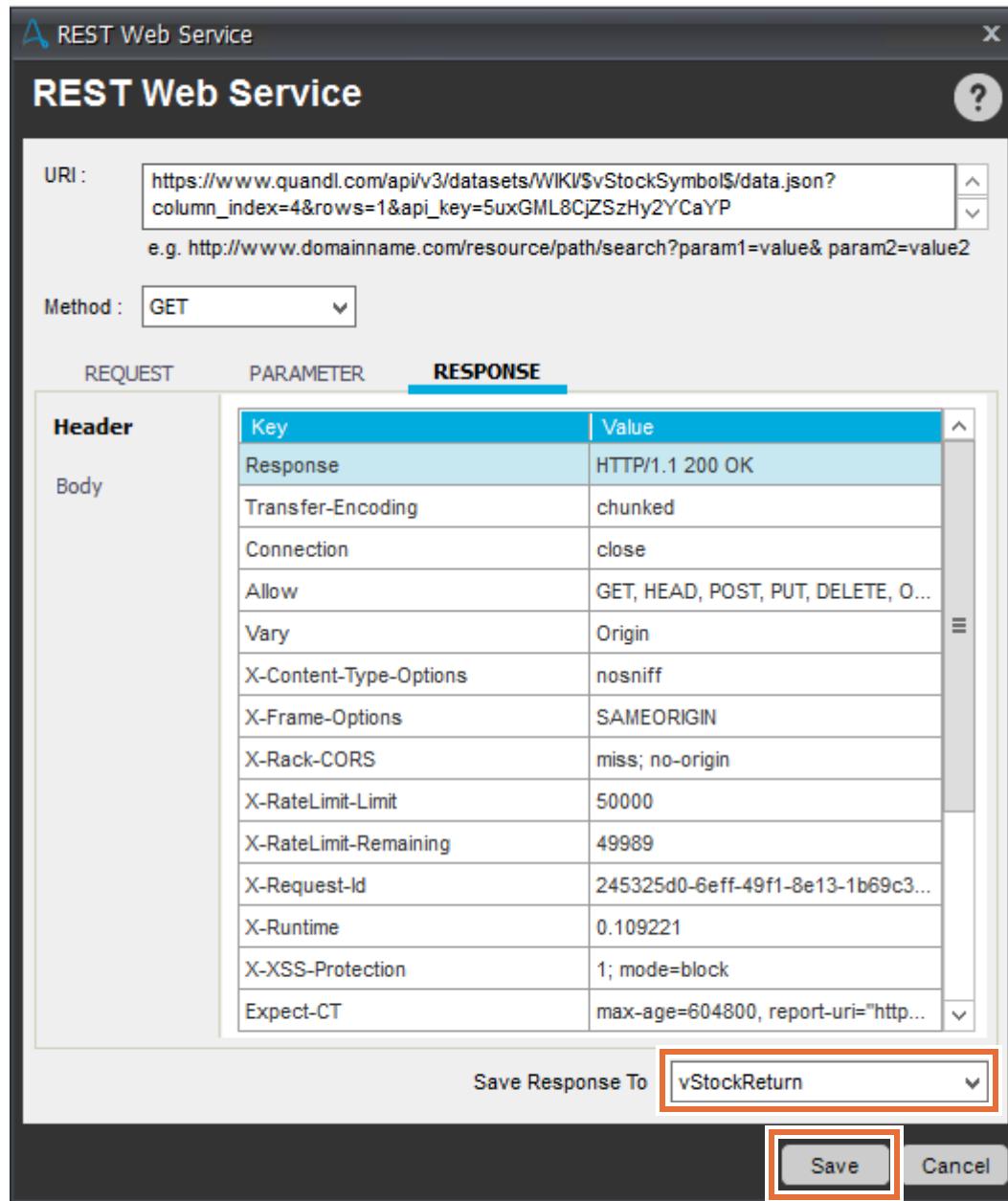
- URI:** `https://www.quandl.com/api/v3/datasets/WIKI/$vStockSymbol$/data.json?column_index=4&rows=1&api_key=5uxGML8CjZSzHy2YCaYP`
- Method:** GET
- RESPONSE Tab:** Selected. A table displays the following headers:

Header	Key	Value
Body	Response	HTTP/1.1 200 OK
	Transfer-Encoding	chunked
	Connection	close
	Allow	GET, HEAD, POST, PUT, DELETE, O...
	Vary	Origin
	X-Content-Type-Options	nosniff
	X-Frame-Options	SAMEORIGIN
	X-Rack-CORS	miss; no-origin
	X-RateLimit-Limit	50000
	X-RateLimit-Remaining	49989
	X-Request-Id	245325d0-6eff-49f1-8e13-1b69c3...
	X-Runtime	0.109221
X-XSS-Protection	1; mode=block	
Expect-CT	max-age=604800, report-uri="http...	

- Save Response To:** vStockReturn
- Buttons:** Save, Cancel

- ___ e. Verify that the **Response** field says: HTTP/1.1 200 OK
- ___ f. In the Rest Web Service window, in the Save Response To section, click **Select Variable**, and select **vStockReturn**.

- g. Click **Save**.



Information

The Quandl service returns information as JavaScript Object Notation (JSON) objects.

3.3. Constructing the error message from the REST response

- 1. From the Commands list, add the following comment after the REST Web Service command:

Construct error message from REST response.

- ___ 2. Assign the value of vStockReturn value to vQuandlError.

In this step, you copy the value of the vStockReturn response message to the vQuandlError variable. By doing so, you can use the vQuandlError variable to determine whether the response returned an error.

- ___ a. In the Commands list, double-click the **Variable Operation** command to add it to the Actions List.
- ___ b. In the Specify Variable section, click **Select variable**, and select **vQuandlError**.
- ___ c. In the **Specify value** field, enter `$vStockReturn$`.
- ___ d. Click **Save**.



Information

Quandl formats the two errors that you configure the bot to handle in different ways. When you extract the error code as a substring of the error message string, the error code must be in the same location of the message string. Thus, you must configure the bot to remove spaces from the error message string to ensure that the error code is in a consistent location, regardless of the error message that you receive.



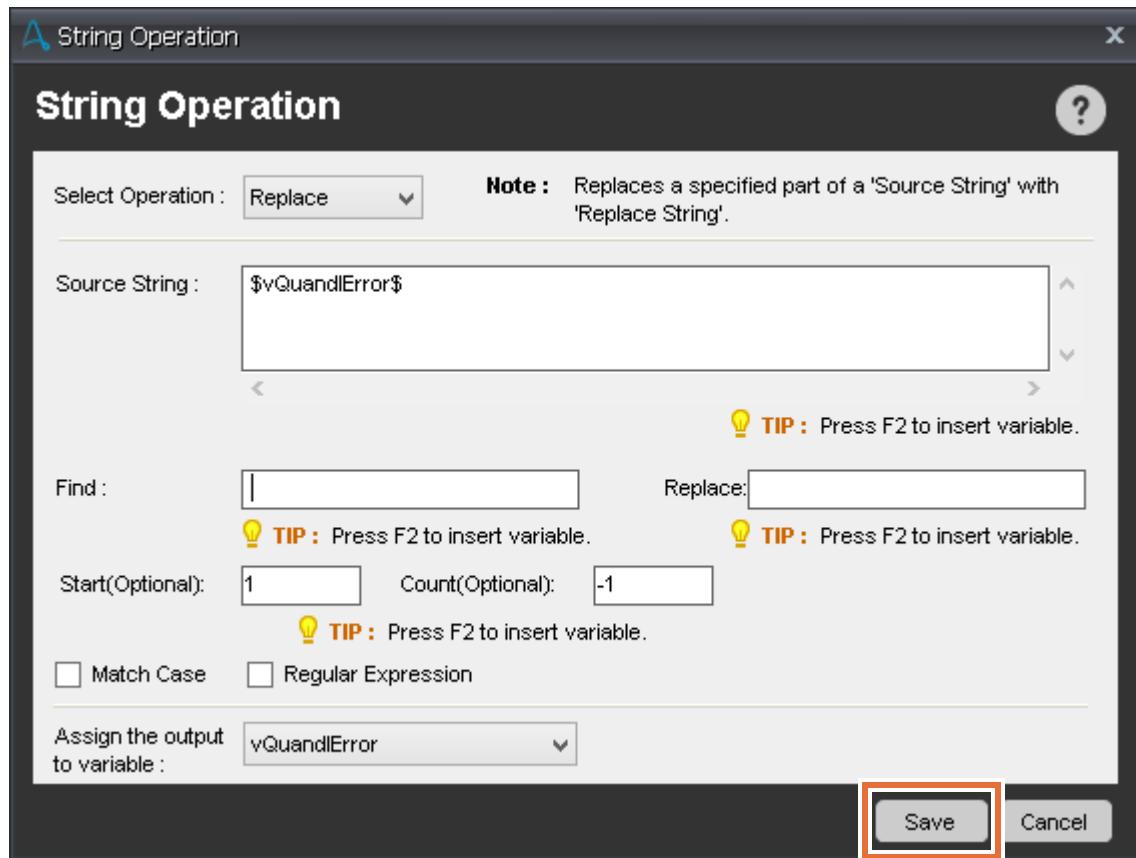
Note

You can see examples of the expected return messages from the Quandl service by viewing the following document:

C:\labfiles\trade_booking_advanced\solution\REST Responses.txt

- ___ 3. Add a String Operation command to delete spaces from the error message string.
- ___ a. Double-click the **Replace** command under **String Operation**.
 - ___ b. Click the **Source String** field, and press F2 (or Fn+F2) to enter a variable.
 - ___ c. Select the **vQuandlError** variable, and click **Insert**.
 - ___ d. Click in the **Find** field, and press the **Spacebar** to enter a space.
 - ___ e. Click the **Assign the output to variable**, and select **vQuandlError**.

__ f. Click **Save**.



3.4. Extracting the Quandl error code and error message



Information

The following JSON object is an example of a Quandl error:

```
{ "quandl_error":{ "code": "QECx02", "message": "You have submitted an incorrect Quandl code. Please check your Quandl codes and try again."}}
```

In this step, you work with the JSON key-value pairs to identify and extract the error code and the error message.

**Note**

In this section, you use the Before-After command to identify a substring that is between the two values that you define in the **Before** and **After** fields. The values in the **Before** and **After** fields function like bookends that encapsulate the substring that you want to extract. You can assign the identified substring to a variable.

-
- ___ 1. In the **Commands** list, double-click **String Operation** and double-click the Before-After command.
 - ___ 2. Click the **Source String** field, and press F2.
 - ___ 3. Select the **vQuandIError** variable and click **Insert**.
 - ___ 4. Copy and paste the following text into the **Before (Optional)** text box:

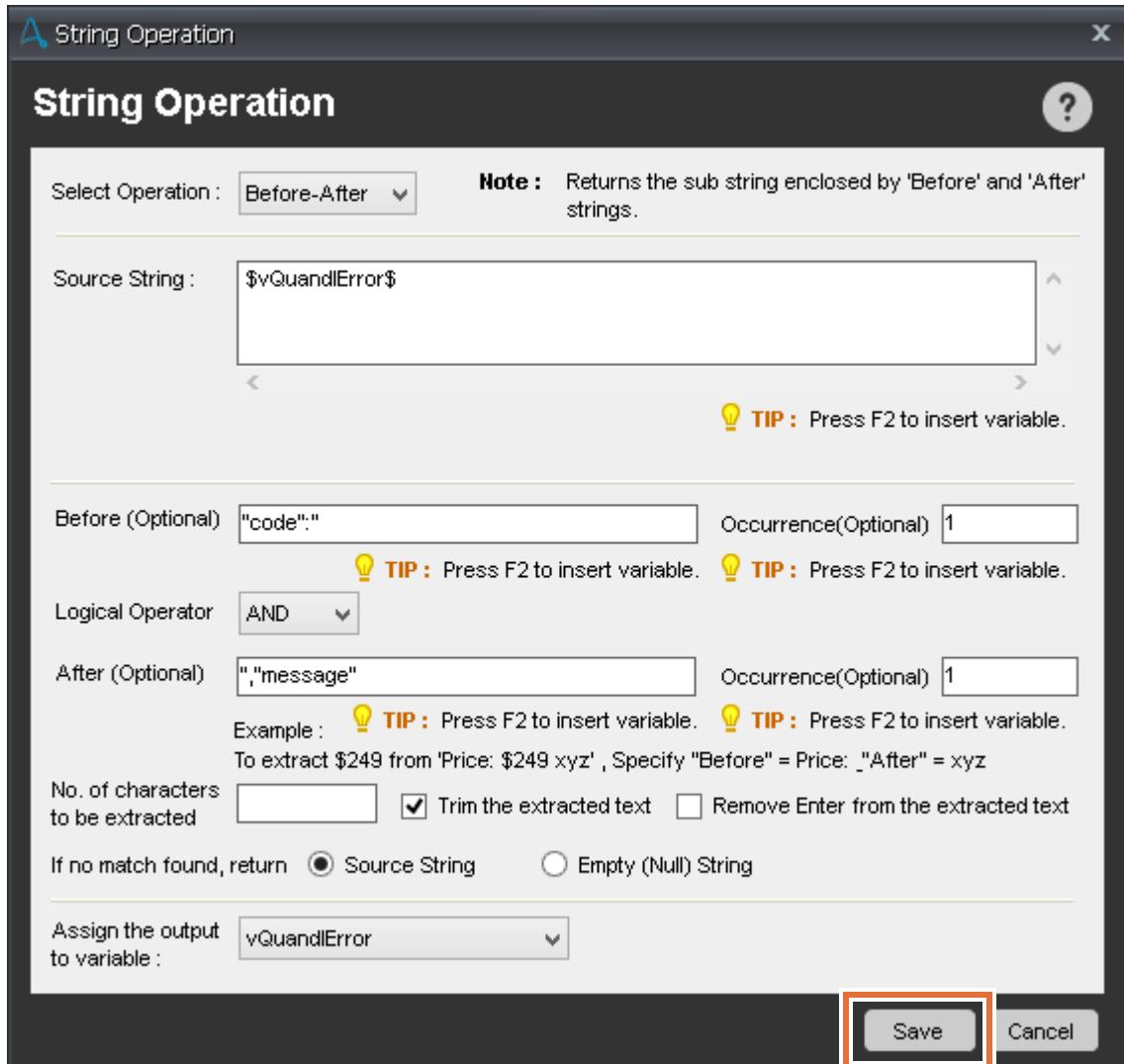
`"code" : "`

This setting defines the text that is before the error code substring. The bot looks for the value that you enter here, and identifies the text immediately after the defined value as the substring to extract.

- ___ 5. Change the **Logical Operator** from the default OR to **AND**.
 - ___ 6. Copy and paste the following text into the **After (Optional)** text box:

`", "message"`
- This setting defines the text that is after the error code substring. The bot stops extracting the substring when it identifies the text value that you enter in this field.
- ___ 7. From the **Assign the output to variable** list, select **vQuandIError**.

- __ 8. Click **Save**.



3.5. Removing unneeded data from the stock return message

In this step, you configure the `Before-After` String Operation command to assign the elements in the `"data"`: array to the appropriate local variables.



Information

The following JSON object is an example of a successful Quandl response:

```
{"dataset_data":{"limit":1,"transform":null,"column_index":4,"column_names":["Date","Close"],"start_date":"1962-01-02","end_date":"2018-02-28","frequency":"daily","data":[[{"Date": "2018-02-28", "Close": 155.83}]],"collapse":null,"order":null}}
```

The value of the `"data"`: key is an array that holds the date of the stock price, and the amount of the stock price in US dollars.

- ___ 1. In the **Commands** list, double-click **String Operation** and double-click the **Before-After** command.
 - ___ 2. Click the **Source String** field, and press F2.
 - ___ 3. Select the **vStockReturn** variable and click **Insert**.
 - ___ 4. Copy and paste the following text into the **Before (Optional)** text box:
`"data": [[`
 - ___ 5. From the **Logical Operator** list, select **AND**.
 - ___ 6. Copy and paste the following text into the **After (Optional)** text box:
`]], "collapse":null, "order":null}]}`
-

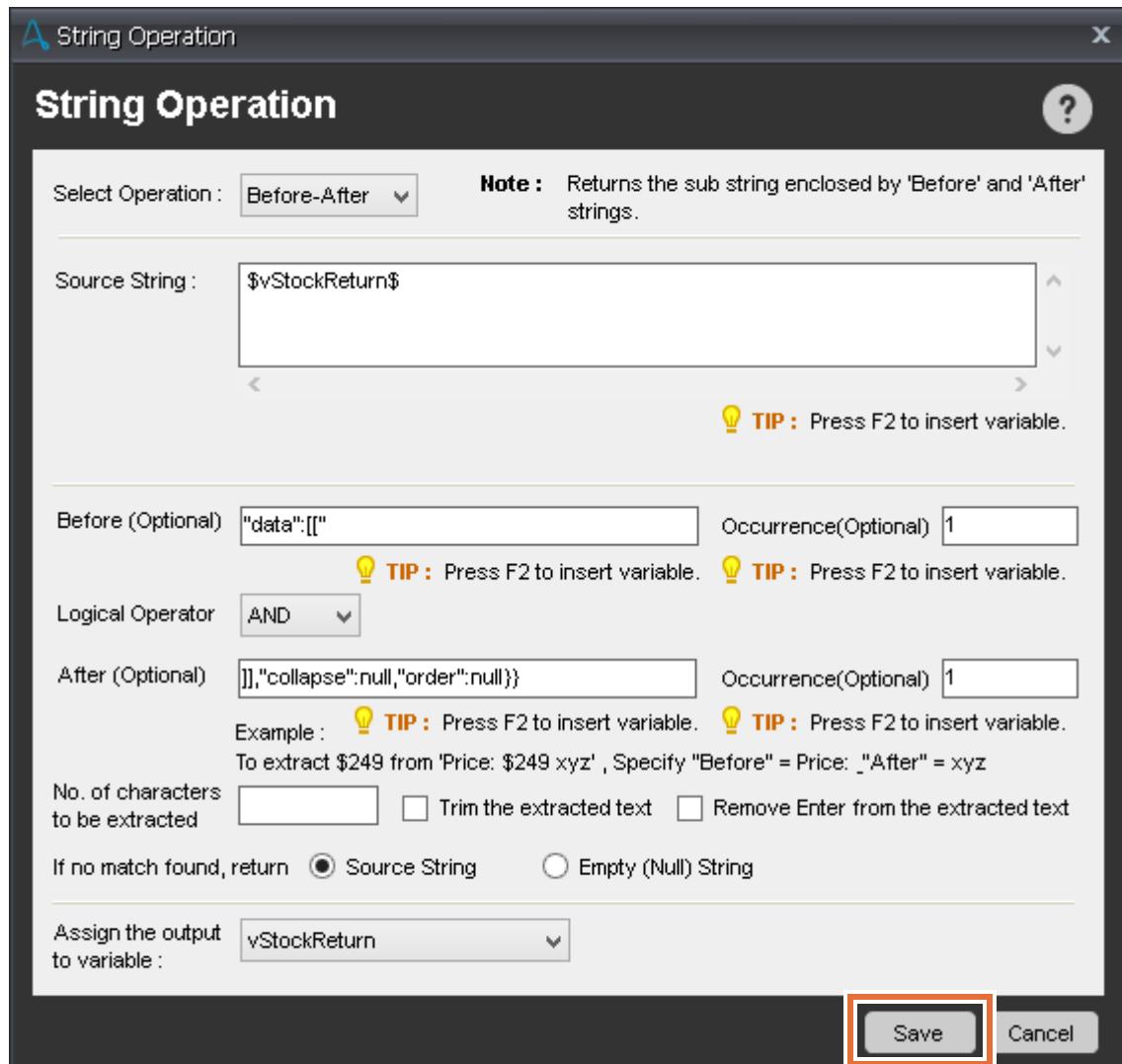
**Note**

The values in the **Before** and **After** fields demarcate the stock date and stock price as the substring to extract.

- ___ 7. From the **Assign the output to variable** list, select **vStockReturn**.

This setting reassigns the stock date and price to the **vStockReturn** variable.

8. Click **Save**.



Section 4. Evaluating whether the REST response has an error

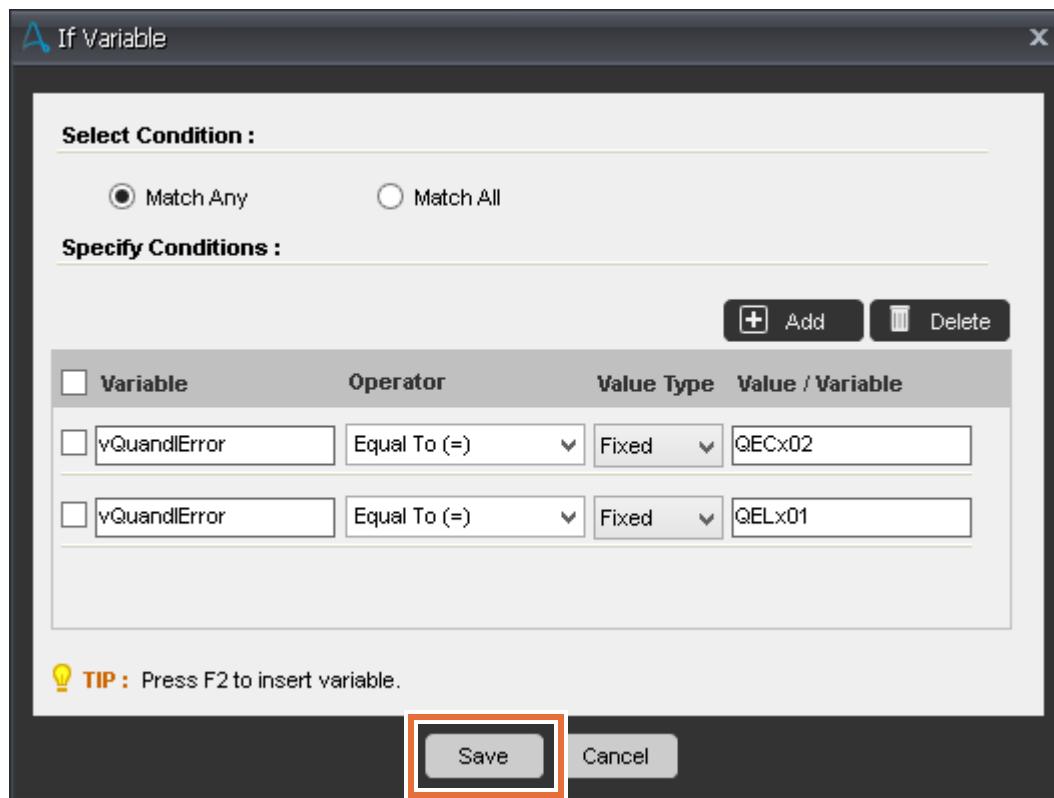
In this section, you add error handling to deal with errors in the Quandl response. You use an If/Else command to see whether the response contains either of the two error codes. You also configure the bot to send a notification email if either one of the error codes is present.

- ___ 1. Add the following comment:

Evaluate response from Quandl service to see whether error codes are present

- ___ 2. Double-click the **If/Else** command and double-click the **Variable** command.
- ___ 3. Click the **Edit** button next to the **IF Condition** field.
- ___ 4. Click **Add more conditions**.
- ___ 5. Click the **Variable** field, and press F2.
- ___ 6. Select the **vQuandlError** variable and click **Insert**.
- ___ 7. For **Operator**, select **Equal To (=)**.
- ___ 8. Leave **Value Type** as **Fixed**.
- ___ 9. In the **Value / Variable** field, enter: `QECx02`
- ___ 10. Click **Add**, and define a second condition with the following parameters:
- Variable: **vQuandlError**
 - Operator: **Equal To (=)**
 - Value Type: **Fixed**
 - Value / Variable: `QELx01`

— 11. Click **Save**.



Your code should match this screen capture.

```

31  [Comment] Comment: =====
32  [Comment] Comment: Evaluate data and send email based on one of three situations.
33  [Comment] Comment: =====
34  [Comment] Comment: Evaluate length of each variable.
35  [IF] If $vLengthParty1$ Equal To (=) "0" OR $vLengthParty2$ Equal To (=) "0" OR $vLengthReporterName$ Eq
36  [Comment] Comment: Scenario: Missing Data and click Save (replacing the default text).
37  [Send Email] Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
38  [ELSE]
39  [REST Web Service] REST Web Service: Method: "GET"; Request: "https://www.quandl.com/api/v3/datasets/WIKI/$vSto
40  [Comment] Comment: Construct error message from REST response.
41  [Variable Operation] Variable Operation: $vStockReturn$ To $vQuandlError$
42  [String Operation] String Operation: Replace " " with "" in "$vQuandlError$" and assign output to $vQuandlError$
43  [String Operation] String Operation: Before-After "$vQuandlError$"; Before: ""code"":""; After: """, "message"" and assign ou
44  [String Operation] String Operation: Before-After "$vStockReturn$"; Before: ""data":[""], After: "], "collapse":null, "order":n
45  [Comment] Comment: Evaluate response from Quandl service to see whether error codes are present
46  [IF] If $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
47  [Comment] Comment: Please enter the conditional commands here.
48  [End If]
49  [Comment] Comment: Scenario: All Data Present
50  [Send Email] Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
51  [End If]

```

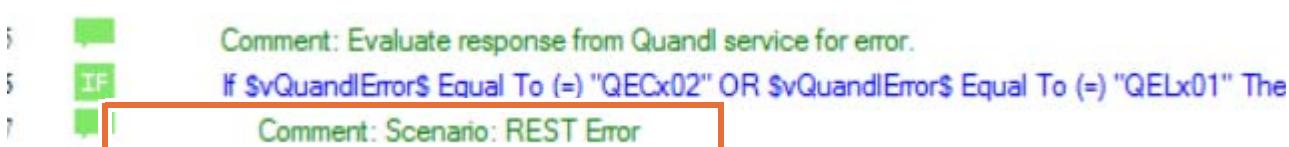


Information

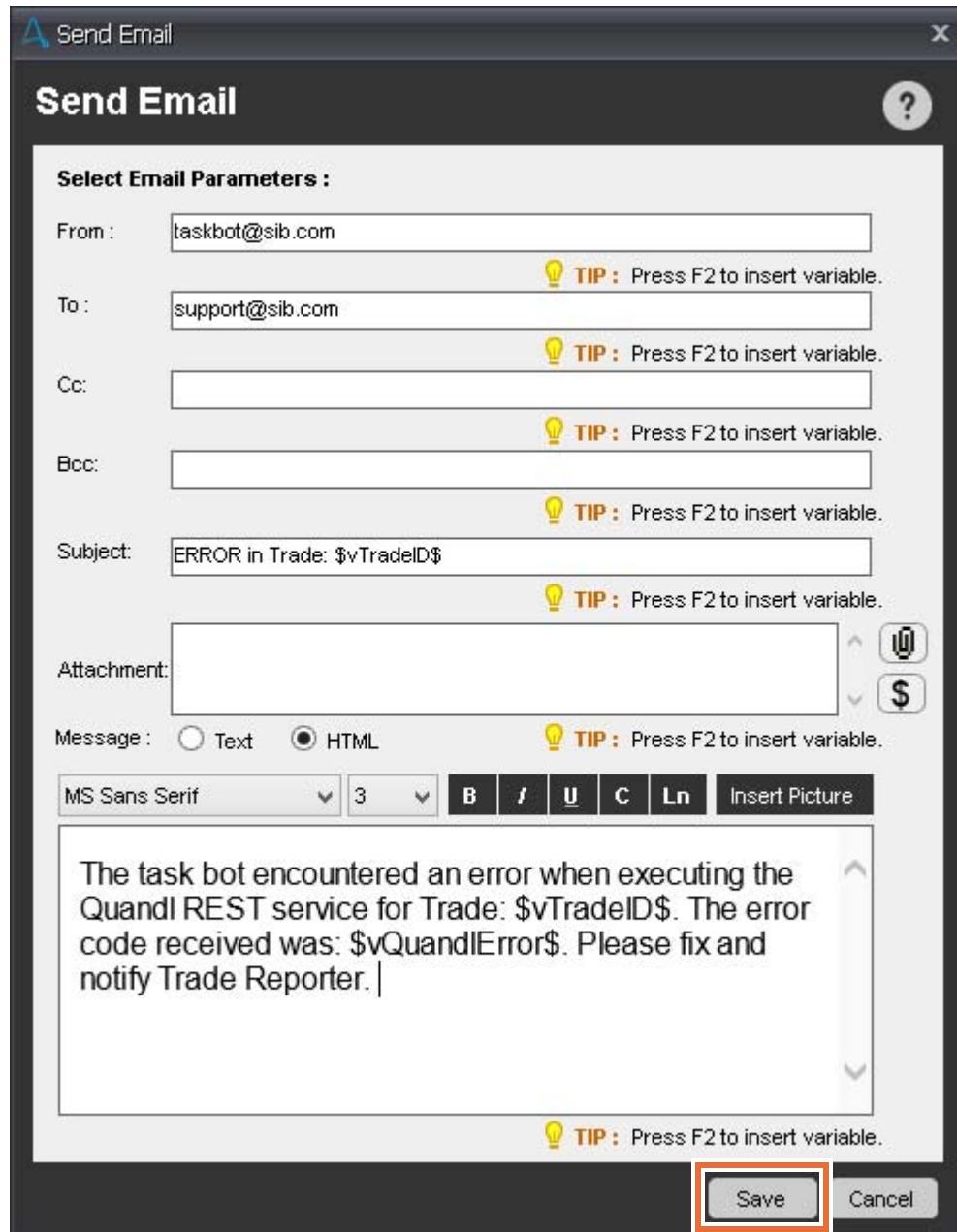
The QECx02 error code means that a bad stock symbol was passed in the request. This scenario is tested later in this lab.

The QELx01 error code occurs when the limit is exceeded for the free web service. Although this situation is unlikely to happen, the bot handles the error and responds if needed.

4.1. Configuring the email response to a REST error

- 1. Edit the comment.
 - a. Double-click the **Comment** command after the **If/Else** command that you just added and enter: Scenario: REST Error
 
 - b. Click **Save**.
- 2. Configure the **Send Email** command to send an email to the SIB Trade Reporter for the REST error scenario.
 - a. Double-click the **Send Email** command.
 - b. Enter the following values in the corresponding fields, and click **Save**.
 - **From:** taskbot@sib.com
 - **To:** support@sib.com
 - **Subject:** ERROR in Trade: \$vTradeID\$
 - **Message:** Select **HTML**, and enter the following message for the body of the email.
The task bot encountered an error when executing the Quandl REST service for Trade: \$vTradeID\$. The error code received was: \$vQuandlError\$. Please fix and notify Trade Reporter.

- ___ c. Click **Save**.



- ___ 3. Configure an **Else** condition.
- ___ a. In the **Commands** list, go to the **If/Else** section, double-click the **Else** command.

- ___ b. Drag the following two commands for the **Scenario: All Data Present** after the **Else** command.

Comment: Scenario: All Data Present

Send Email: Subject "Trade: \$vTradeID\$ Details"

```

46 IF      If $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
47   Comment: Scenario: REST Error
48   Send Email: Subject "ERROR in Trade: $vTradeID$"
49 ELSE    Else
50   End If
51   Comment: Scenario: All Data Present
52   Send Email: Subject "Trade: $vTradeID$ Details"
53 End If

46 IF      If $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
47   Comment: Scenario: REST Error
48   Send Email: Subject "ERROR in Trade: $vTradeID$"
49 ELSE    Else
50   Comment: Scenario: All Data Present
51   Send Email: Subject "Trade: $vTradeID$ Details"
52   End If
53 End If

```

Section 5. Working with the Quandl stock price data

In this section of the exercise, you extract the stock price and date from the data that was sent by the Quandl service, and assign the stock price and date to local variables.

- 1. Add the following comment after the new `Else` command. The following commands need to run before the `Send Email` command.

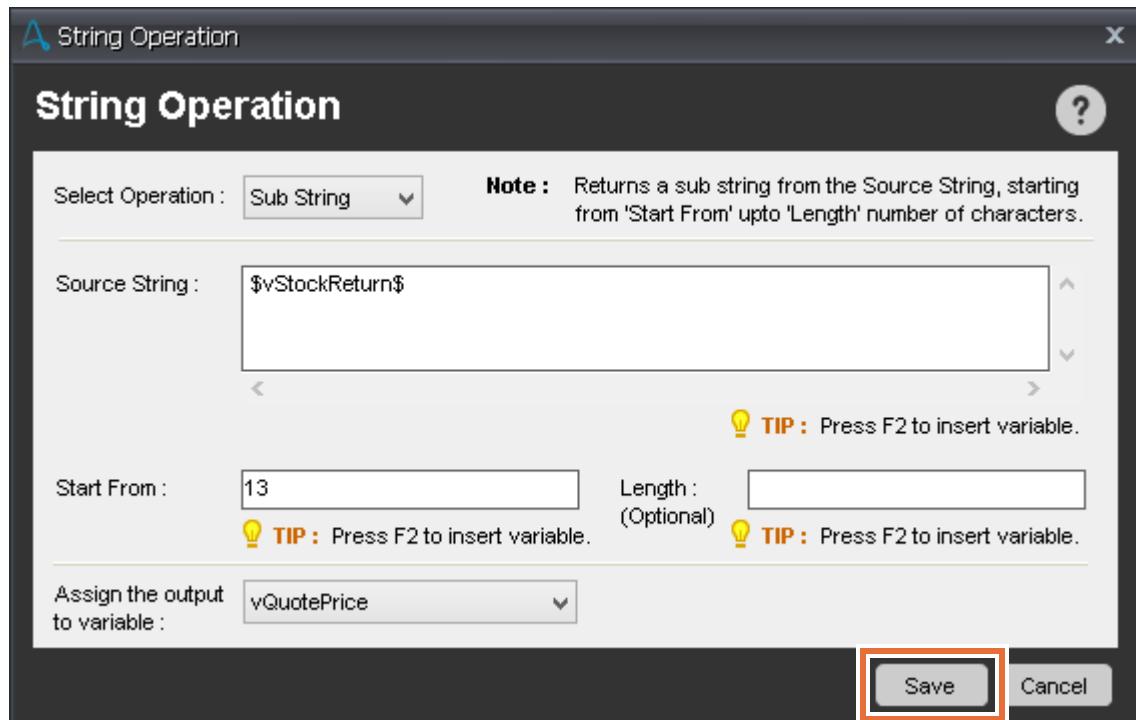
Extract Stock Price and Date

```

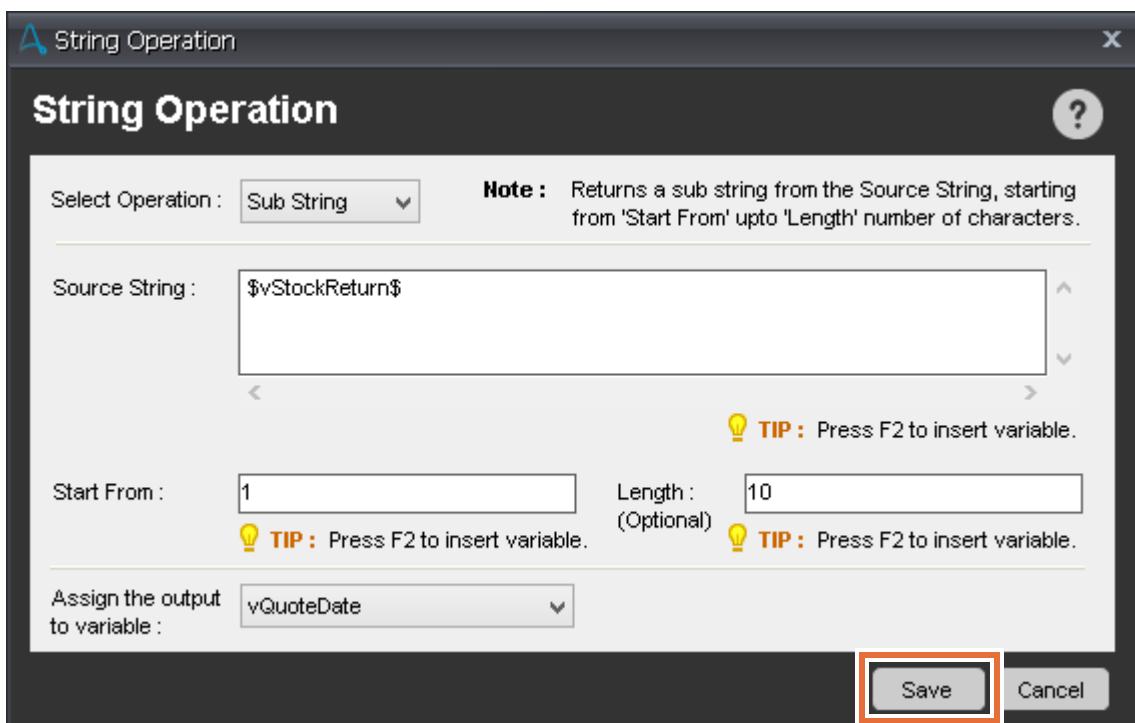
46  IF      If $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
47      Comment: Scenario: REST Error
48      Send Email: Subject "ERROR in Trade: $vTradeID$"
49  Else    Else
50      Comment: Extract Stock Price and Date.

```

- 2. Configure a `String Operation` command to extract the stock price data and assign it to `vQuotePrice`.
 - a. In the String Operation section of the **Commands** list, double-click the `Sub String` command.
 - b. Click the **Source String** field and press F2.
 - c. Select the **vStockReturn** variable and click **Insert**.
 - d. In the **Start From** field, enter: 13
 - e. From the **Assign the output to variable** list, select **vQuotePrice**.
 - f. Click **Save**.



- ___ 3. Configure a String Operation command to extract the stock quote date data and assign it to vQuoteDate.
- In the String Operation section of the Commands list, double-click the Sub String command.
 - Click the **Source String** text box and press F2.
 - Select the **vStockReturn** variable and click **Insert**.
 - In the **Start From** field, enter: 1
 - In the **Length** field, enter: 10
 - From the **Assign the output to variable** list, select **vQuoteDate**.
 - Click **Save**.



- ___ 4. Add the following comment:
- Calculate Trade Amount.
- ___ 5. Configure a Variable Operation command to calculate the price of the total trade based on the retrieved stock price, and assign the total amount to vTradeAmount.
- In the **Commands** list, double-click the **Variable Operation** command.
 - From the **Specify Variable** list, select **vTradeAmount**.
 - In the **Specify value for \$vTradeAmount\$** field, enter the following value for the calculation:

$\$vQuotePrice\$ * \$vShares\$$

This calculation multiplies the price of the stock, which was retrieved from Quandl, by the number of shares that was listed from the trade receipt PDF.

- ___ d. Click **Save**.
- ___ 6. Update the original email to the trade supervisor so that it includes information from the Quandl service.
 - ___ a. Go to the `Send Email` command under the *Scenario: All Data Present* portion of the code and double-click it to edit the command.
 - ___ b. Add the following lines to the email after the **Number of Shares** line:

Trade Stock Price: \$vQuotePrice\$
 Trade Stock Date: \$vQuoteDate\$
 Total Trade Amount: \$vTradeAmount\$



Information

You can copy the complete message and paste it into the field by using the following text.

- ___ c. Verify the email against the following text:

Dear Trade Reporter Supervisor,
 The taskbot finished evaluating data for trade: \$vTradeID\$
 It determined all required fields are present.

The trade details are as follows:

Trade Stock Symbol: \$vStockSymbol\$
 Number of Shares: \$vShares\$
 Trade Stock Price: \$vQuotePrice\$
 Trade Stock Date: \$vQuoteDate\$
 Total Trade Amount: \$vTradeAmount\$
 Trade Reporter Name: \$vReporterName\$
 Reporting Country: \$vReportingCountry\$
 Party1: \$vParty1\$
 Party2: \$vParty2\$

 Trade Description: \$vTradeDesc\$

- ___ d. Click **Save**.

Your code should match this screen capture.

```

35  IF
36  |
37  | Comment: Scenario: Missing Data
38  | Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
39  ELSE
40  |
41  | REST Web Service: Method: "GET"; Request: "https://www.quandl.com/api/v3/datasets/WIKI/$
42  | Comment: Construct error message from REST response.
43  | Variable Operation: $vStockReturn$ To $vQuandlError$
44  | String Operation: Replace "" with "" in "$vQuandlError$" and assign output to $vQuandlError$
45  | String Operation: Before-After "$vQuandlError$"; Before: ""code"":""; After: ""","message"" and assig
46  | String Operation: Before-After "$vStockReturn$"; Before: ""data":[]; After: "],"collapse":null,"ord
47  | Comment: Evaluate response from Quandl service to see whether error codes are present
48  IF $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
49  |
50  | Comment: Scenario: REST Error
51  | Send Email: Subject "ERROR in Trade: $vTradeID$"
52  ELSE
53  |
54  | Comment: Extract Stock Price and Date
55  | String Operation: Extract substring from "$vStockReturn$" and assign output to $vQuotePrice$"
56  | String Operation: Extract substring from "$vStockReturn$" and assign output to $vQuoteDate$"
57  | Comment: Calculate Trade Amount.
58  | Variable Operation: $vQuotePrice$ * $vShares$ To $vTradeAmount$"
59  | Comment: Scenario: All Data Present
60  | Send Email: Subject "Trade: $vTradeID$ Details"
61  End If
62  End If

```

Section 6. Evaluating the length of decimal values in the stock price

When you use the **Variable Operation** command in the Workbench, decimal values are rounded.

For example, if the value 04.50 is assigned to a variable, the value that is held in memory is 4.5. This rounding convention works for calculations, but it might not work well for displaying currency values.

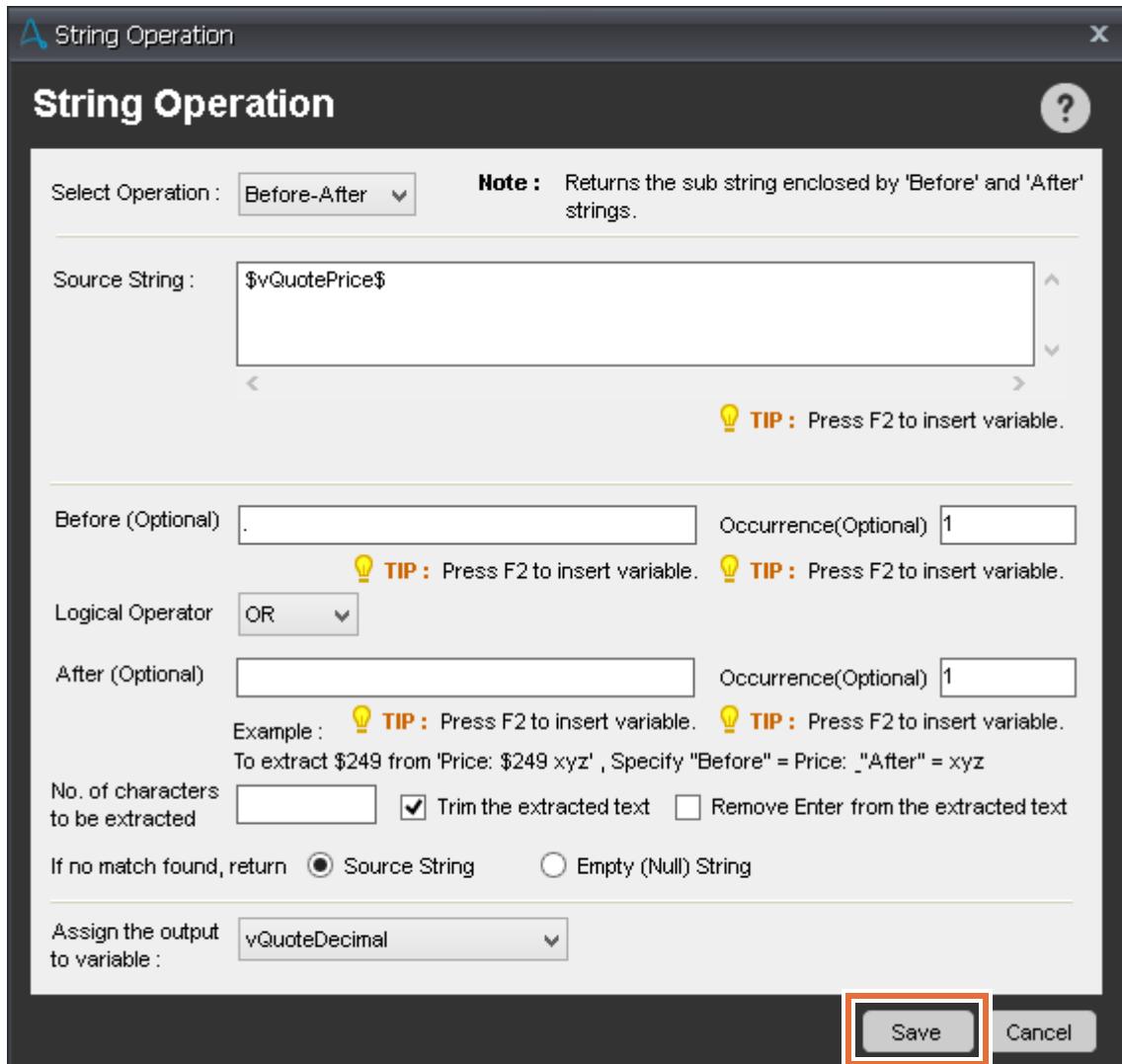
To represent the trade amount and quote price values with decimals, the bot needs to evaluate the value and determine whether it must add a decimal or zeros.

You add the following lines of code after the calculation of the Trade Amount and before the `Send Email` command for the All Data Present scenario.

- ___ 1. Add the following comment after the **Variable Operation** command that calculates the **Trade Amount**.

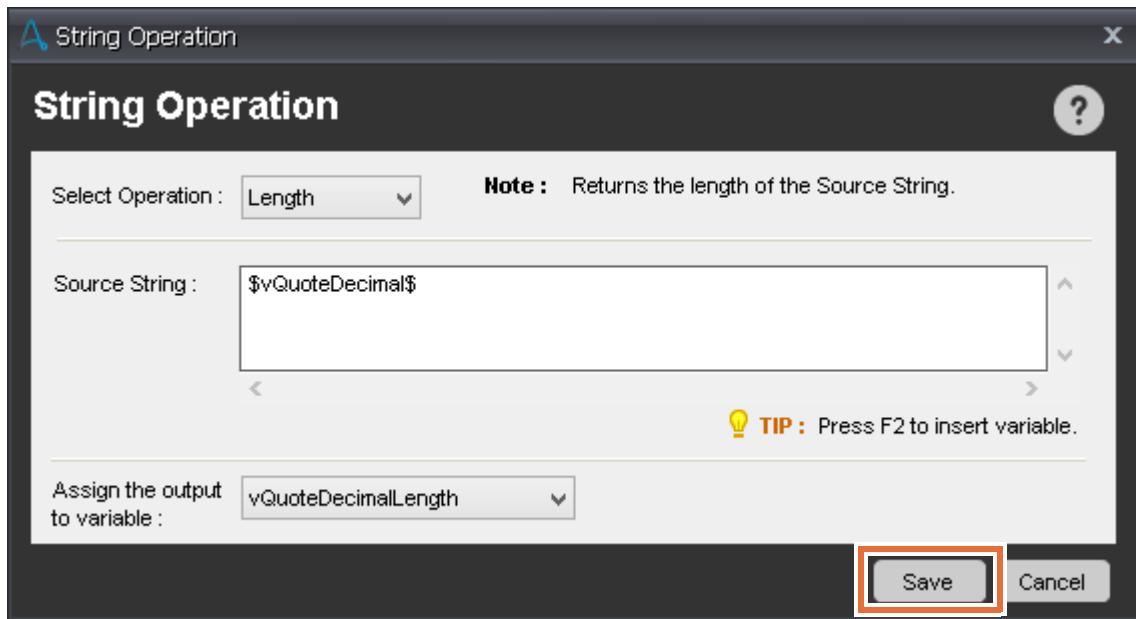
Extract values to evaluate decimal length.
- ___ 2. Add a **Before-After String Operation** command to extract the decimal value from the `vQuotePrice` value.
 - ___ a. In the String Operation section of the **Commands** list, double-click the **Before-After** command.
 - ___ b. Click the **Source String** field, and press F2 to insert a variable.
 - ___ c. Select `vQuotePrice` and click **Insert**.
 - ___ d. Type a period (.) in the **Before (Optional)** field.
 - ___ e. From the **Assign the output to variable** list, select `vQuoteDecimal`.

___ f. Click **Save**.



- ___ 3. Add a **Length** String Operation command to evaluate the length of the value of the **vQuoteDecimal** variable.
- In the String Operation section of the **Commands** list, double-click the **Length** command.
 - Click the **Source String** field, and press F2.
 - Select **vQuoteDecimal** and click **Insert**.
 - From the **Assign the output to variable** list, select **vQuoteDecimalLength**.

- ___ e. Click **Save**.



- ___ 4. Add a Before-After String Operation command to extract the decimal values for the vTradeAmount variable.
- In the **Commands** list, go to the String Operation section and double-click the **Before-After** command.
 - Click the **Source String** field, and press F2.
 - Select **vTradeAmount** and click **Insert**.
 - Type a period (.) in the **Before (Optional)** field.
 - From the **Assign the output to variable** list, select **vTradeAmountDecimal**.
 - Click **Save**.
- ___ 5. Add a Length String Operation command to calculate the length of the vTradeAmount decimal value.
- In the String Operation section of the **Commands** list, double-click the **Length** subcommand.
 - Click the **Source String** field, and press F2.
 - Select **vTradeAmountDecimal** and click **Insert**.
 - From the **Assign the output to variable** list, select **vTradeAmountDecimalLength**.
 - Click **Save**.

Your code for this section should match this screen capture.

```

35  IF      If $vLengthParty1$ Equal To (=) "0" OR $vLengthParty2$ Equal To (=) "0" OR $vLengthReporterName$ Equal To (=)
36  ,       Comment: Scenario: Missing Data
37  X       Send Email: Subject "Trade: $vTradeID$ Details" with Attachment(s).
38  ELSE    Else
39  ,       REST Web Service: Method: "GET"; Request: "https://www.quandl.com/api/v3/datasets/WIKI/$vStockSymbol"
40  ,       Comment: Construct error message from REST response.
41  $       Variable Operation: $vStockReturn$ To $vQuandlError$
42  ,       String Operation: Replace "" with "" in "$vQuandlError$" and assign output to $vQuandlError$
43  ,       String Operation: Before-After "$vQuandlError$"; Before: "";"code"":"""; After: "";"message"" and assign output to $v
44  ,       String Operation: Before-After "$vStockReturn$"; Before: "";"data":[]; After: "],"collapse":null,"order":null}" and a
45  ,       Comment: Evaluate response from Quandl service to see whether error codes are present
46  IF      If $vQuandlError$ Equal To (=) "QECx02" OR $vQuandlError$ Equal To (=) "QELx01" Then
47  ,       Comment: Scenario: REST Error
48  X       Send Email: Subject "ERROR in Trade: $vTradeID$"
49  ELSE    Else
50  ,       Comment: Extract Stock Price and Date
51  ,       String Operation: Extract substring from "$vStockReturn$" and assign output to $vQuotePrice$
52  ,       String Operation: Extract substring from "$vStockReturn$" and assign output to $vQuoteDate$
53  ,       Comment: Calculate Trade Amount.
54  $       Variable Operation: $vQuotePrice$ * $vShares$ To $vTradeAmount$
55  ,       Comment: Extract values to evaluate decimal length.
56  ,       String Operation: Before-After "$vQuotePrice$"; Before: "." and assign output to $vQuoteDecimal$
57  ,       String Operation: Get length of "$vQuoteDecimal$" and assign output to $vQuoteDecimalLength$
58  ,       String Operation: Before-After "$vTradeAmount$"; Before: "." and assign output to $vTradeAmountDecimal$
59  ,       String Operation: Get length of "$vTradeAmountDecimal$" and assign output to $vTradeAmountDecimalLength
60  ,       Comment: Scenario: All Data Present
61  X       Send Email: Subject "Trade: $vTradeID$ Details"
62  X       End If
63  X       End If

```

Section 7. Evaluating decimal values and concatenating zeros when needed

In this part of the exercise, you configure the bot so that it correctly handles decimal values.

- ___ 1. Add the following comment.

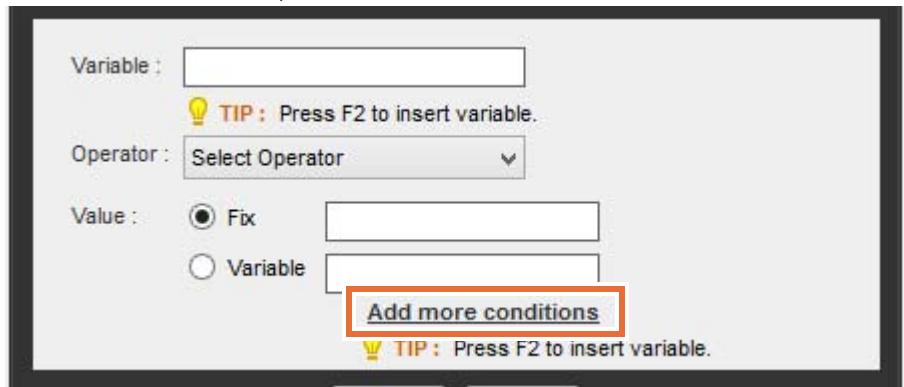
Evaluate need for decimal and concatenate as necessary.

- ___ 2. Configure an If/Else command to evaluate whether the length of the decimal value of the stock price in the vQuotePrice variable is equal to one.

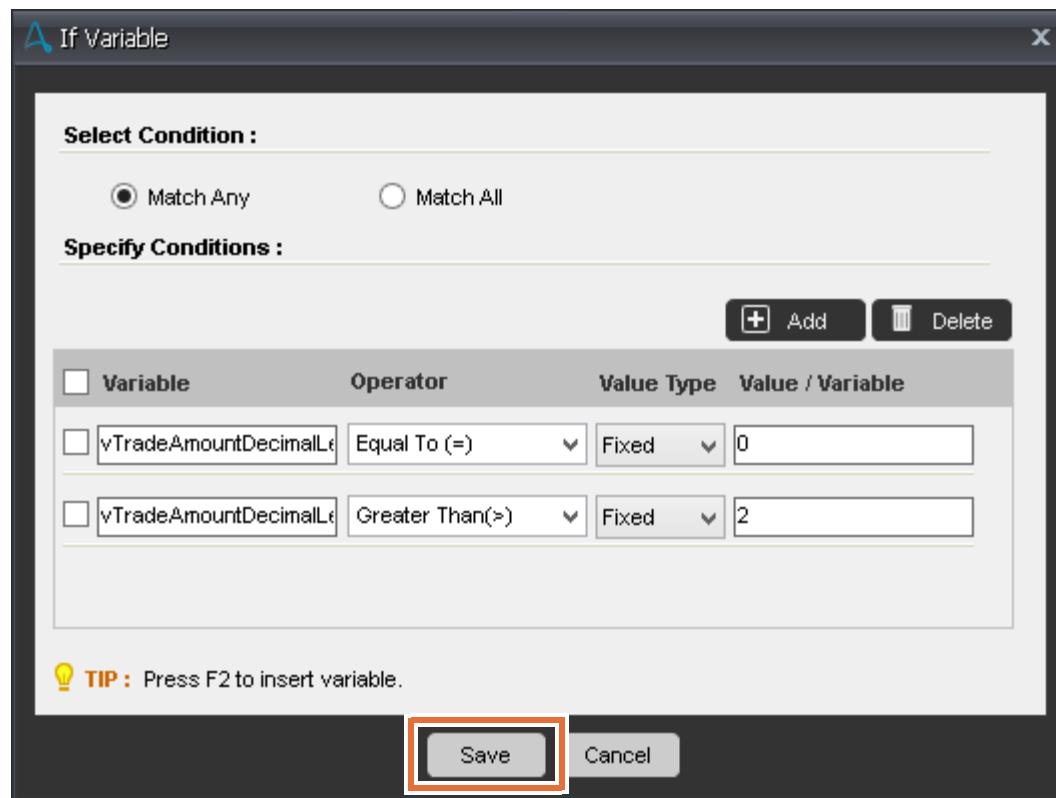
If the decimal value goes only to the tenths place (that is, it has only one digit in the decimal value), you configure the bot to add a zero in the hundredths place.

- ___ a. In the **Commands** list, go to the If/Else section, and double-click the **Variable** command.
- ___ b. Click the **Edit** button to edit the **IF Condition**.
- ___ c. Click the **Variable** field and press F2.
- ___ d. Select the **vQuoteDecimalLength** variable and click **Insert**.
- ___ e. From the **Operator** list, select **Equal To (=)**.
- ___ f. For the **Value Type** option, keep it at **Fix**.
- ___ g. In the **Value / Variable** field, enter: 1
- ___ h. Click **Save**.
- ___ i. Delete the added **Comment** command that states *Please enter the conditional commands here*.
- ___ j. Click the **If** statement above the comment you just deleted.
- ___ 3. Configure Variable Operation command so that when the decimal length of vQuotePrice is one digit (that is, when it ends in the tenths place), it concatenates a zero to the hundredths place.
 - ___ a. In the **Commands** list, double-click the **Variable Operation** command.
 - ___ b. From the **Specify Variable** list, select **vQuotePrice**.
 - ___ c. In the **Specify value for \$vQuotePrice\$** field, enter the following value:
\$vQuotePrice\$0
This value concatenates a zero to the end of the decimal value.
 - ___ d. Click **Save**.
- ___ 4. Configure an Else If command to evaluate whether length of the decimal value in the vQuoteDecimal variable is 0.
 - ___ a. Go to the If/Else section of the **Commands** list, and drag the **Else If** command over the Variable Operation command that you configured in step 3.
 - ___ b. In the **Select Condition** section, select **Variable**.
 - ___ c. In the **IF Condition** section, click **Edit**.

- ___ d. In the If Variable window, click **Add more conditions**.



- ___ e. Click the **Variable** text box and press F2.
 ___ f. Select the **vQuoteDecimalLength** variable and click **Insert**.
 ___ g. For the **Operator**, select **Equal To (=)**.
 ___ h. For **Value Type**, leave it at **Fixed**.
 ___ i. In the **Value / Variable** field, enter: 0
 ___ j. Click **Add**, and define the second condition by using the following settings:
 - Variable: **vQuoteDecimalLength**
 - Operator: **Greater Than (>)**
 - Value Type: **Fixed**.
 - Value / Variable: 2
 ___ k. Click **Save**.





Information

In cases where the bot does not retrieve decimal values, the calculation incorrectly evaluates to the total number of digits because the number values do not have decimals. The second evaluation accounts for that situation.

- ___ 5. Configure a Variable Operation command that concatenates a period (.) and two zeros to the price when the price does not have a decimal value.
 - ___ a. In the **Commands** list, double-click the Variable Operation command.
 - ___ b. From the **Specify Variable** list, select **vQuotePrice**.
 - ___ c. In the **Specify value for \$vQuotePrice\$** field, enter the following value:
\$vQuotePrice\$.00
 - ___ d. Click **Save**.



Information

The Workbench automatically indents the **If** and their associated **End If** commands for ease of reading.

In the following steps, you configure the bot to evaluate the decimal values for the **vTradeAmount** variable so that the decimal values are represented consistently by the bot.

- ___ 6. Configure an **If/Else** command to evaluate length of the decimal value in **vTradeAmountDecimal**.
This **If/Else** command follows the **End If** line for the previous **If/Else** statement.

```

61 IF      If $vTradeAmountDecimalLength$ Equal To (=) "1" Then
62 $       Variable Operation: $vQuotePrice$0 To $vReporterName$
63 EI     Else If $vTradeAmountDecimalLength$ Equal To (=) "0" OR $vTradeAmountDecimalLength$ Greater Than(>)
64 $       Variable Operation: $vQuotePrice$.00 To $vQuotePrice$
65 X      End If
  
```

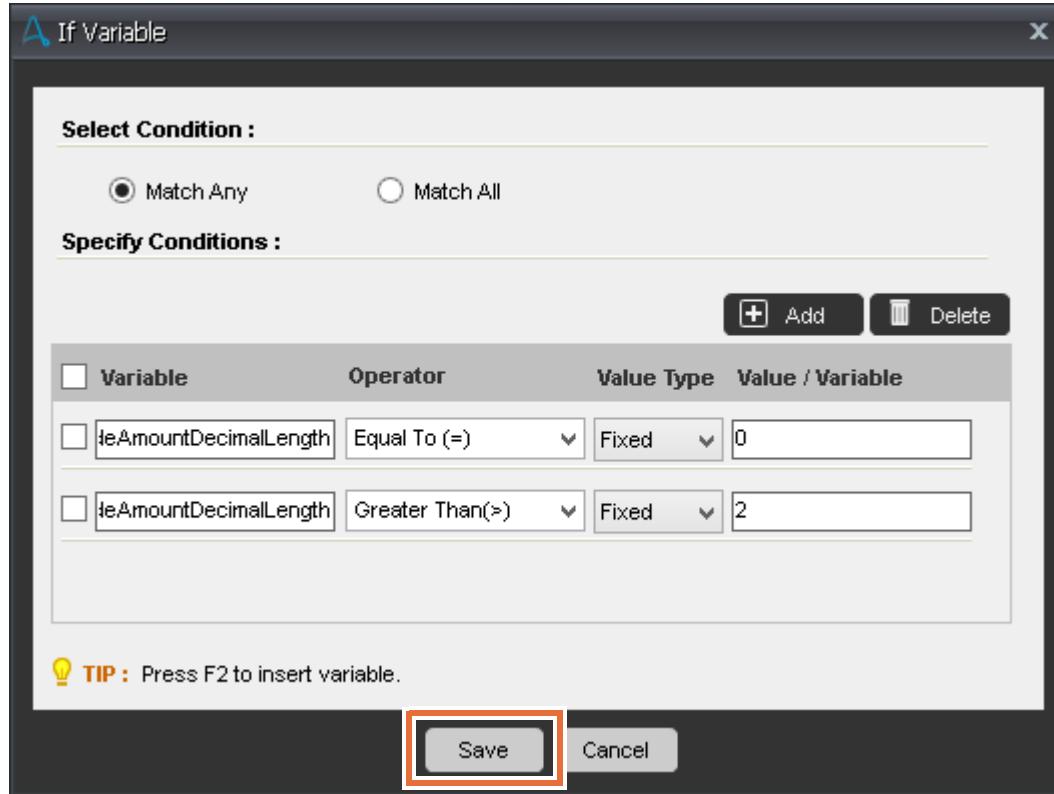
- ___ a. In the **Commands** list, go to the **If/Else** section, and double-click the **Variable** command.
- ___ b. Click the **Edit** button to edit the **IF Condition**.
- ___ c. Click the **Variable** field and press F2.
- ___ d. Select the **vTradeAmountDecimalLength** variable and click **Insert**.
- ___ e. Click the **Operator** menu and select **Equal To (=)**.
- ___ f. For the **Value Type** option, keep it at **Fix**.
- ___ g. In the **Value / Variable** field, enter: 1
- ___ h. Click **Save**.

- ___ i. Delete the added **Comment** command that states *Please enter the conditional commands here.*
- ___ j. Click the **If** statement above the comment you deleted.

Next, you add commands within this If/Else statement.

- ___ 7. Configure a Variable Operation command so that when the decimal length of vTradeAmount is one digit, it concatenates a zero to the hundredths place.
 - ___ a. In the **Commands** list, double-click the **Variable Operation** command.
 - ___ b. From the **Specify Variable** list, select **vTradeAmount**.
 - ___ c. In the **Specify value for \$vTradeAmount\$** field, enter the following value:
\$vTradeAmount\$0
 - ___ d. Click **Save**.
- ___ 8. Configure an Else If command to evaluate whether length of the decimal value in the vTradeAmount variable is 0.
 - ___ a. Go to the If/Else section of the **Commands** list, and drag the **Else If** command Variable Operation command that you configured in a previous step.
 - ___ b. In the **Select Condition** section, select **Variable**.
 - ___ c. In the IF Condition section, click **Edit**.
 - ___ d. In the If Variable window, click **Add more conditions**.
 - ___ e. Click the **Variable** text box and press F2.
 - ___ f. Select the **vTradeAmountDecimalLength** variable and click **Insert**.
 - ___ g. For the **Operator**, select **Equal To (=)**.
 - ___ h. For **Value Type**, leave it at **Fixed**.
 - ___ i. In the **Value / Variable** field, enter: 0
 - ___ j. Click **Add**, and define the second condition by using the following settings:
 - Variable: **vTradeAmountDecimalLength**
 - Operator: **Greater Than (>)**
 - Value Type: **Fixed**.
 - Value / Variable: 2

- ___ k. Click **Save**.



- ___ 9. Configure a Variable Operation command that concatenates a period (.) and two zeros to the price when the price does not have a decimal value.
- In the **Commands** list, double-click the Variable Operation command.
 - From the **Specify Variable** list, select **vTradeAmount**.
 - In the **Specify value for \$vTradeAmount\$** field, enter the following value:
\$vTradeAmount\$.00
 - Click **Save**.
- ___ 10. Save your work.

7.1. Viewing the completed bot

To view the completed bot code, see the `Trade Booking Advanced Solution.txt` file in the `C:\labfiles\trade_booking_advanced\solution` directory.

Section 8. Running the bot

For this exercise, you disable the trigger commands. Instead, you run the bot in 4 limited-scope scenarios to verify the new code.



Information

Although you do not run regression tests during this exercise, regression testing is a good practice for production-level implementations.

Three scenarios were added to this bot. In this section of the exercise, you test the default scenario and the three new scenarios.

As a review, the following list includes all of the scenarios that are handled by this bot:

1. All information present (already implemented)
 - Send email to Trade Reporter Supervisor
2. Missing information (already implemented)
 - Send email to Trade Reporter
3. Alternative stock symbol (new)
 - Send email to Trade Reporter Supervisor
4. REST Error (new)
 - Send Support Help Desk email
5. Error (new)
 - Send Support Help Desk email

In scenarios 1 and 3, additional information is sent to the trade reporter supervisor, including the stock price, the date of the stock price, and the total amount of the trade.

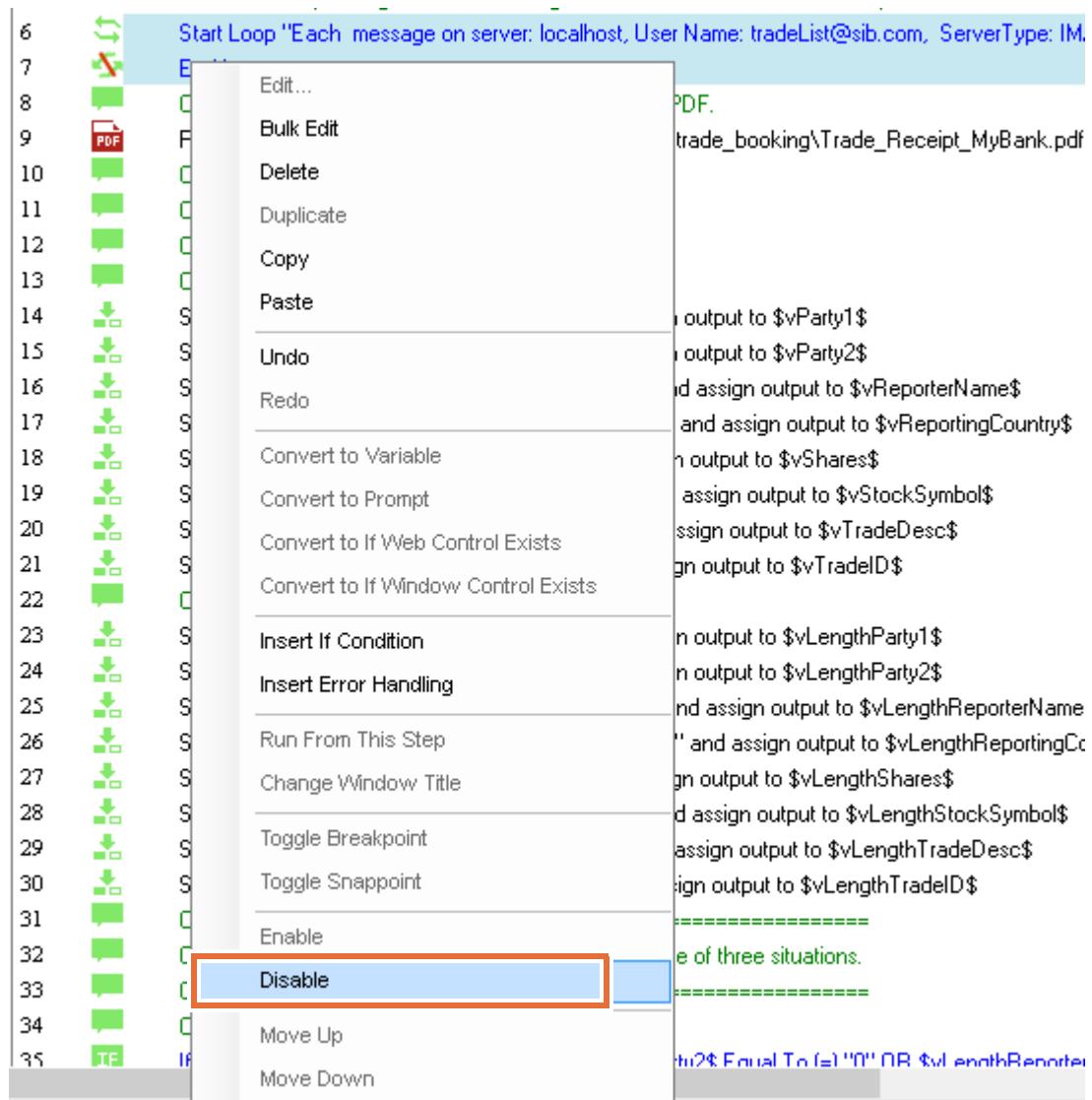
In scenario 4, the bot handles the specific error that is returned by the Quandl service.

Scenario 5 uses the `Error Handling` command that handles all other errors.

8.1. Scenario 1: Verifying that all information is present

This scenario is the default scenario. You disable the commands to download the PDF attachments from the tradeList@sib.com email account, but you run the bot to confirm that it still works.

- 1. Disable the commands that extract the trade receipt PDF attachment from the tradeList@sib.com email account and delete the downloaded files.
 - a. Select the **Start Loop** and **End Loop** commands (lines 6 and 7), right-click them, and click **Disable**.



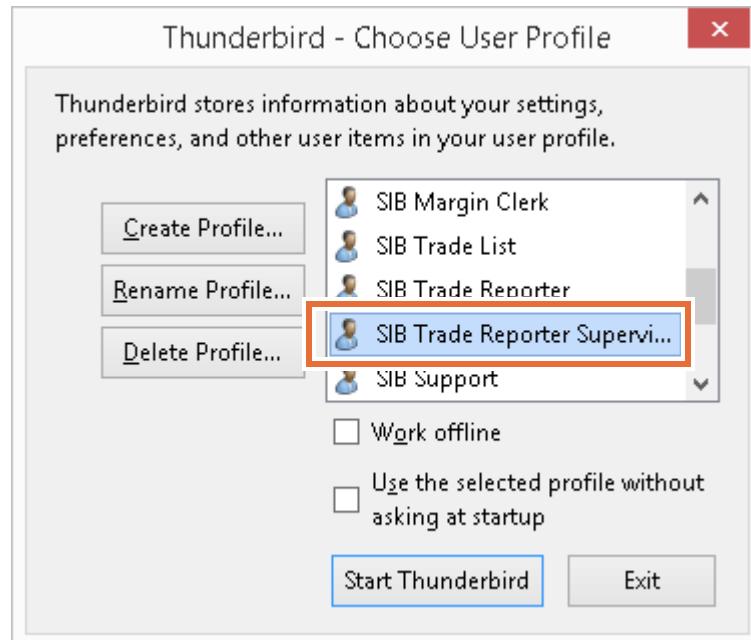
- b. Disable the **Delete All Messages on server** and **Delete Files** commands, by right-clicking these lines, and clicking **Disable**.

75	Comment: =====
76	Comment: Cleanup: Delete all emails and files in folder.
77	Comment: =====
78	Delete All Messages on server: 'localhost'
79	Delete Files 'C:\labfiles\trade_booking\Trade_Receipt_MyBank.pdf'

- ___ c. On the toolbar, click **Save** to save your changes.
- ___ 2. Go to the C:\labfiles\trade_booking folder and delete any remaining PDF files that are in the folder from your testing in [Exercise 6, "Creating a bot to evaluate data from a PDF and send an email"](#).
- ___ 3. Set up the bot for the *Scenario: All information present* test.
 - ___ a. Go to the C:\labfiles\trade_booking\trade_receipts_scenarios\all_information folder.
 - ___ b. Copy Trade_Receipt_MyBank.pdf and paste it into the C:\labfiles\trade_booking folder.
- ___ 4. On the toolbar, click **Run** to start the bot.
- ___ 5. Confirm that the SIB Trade Reporter Supervisor received the email message.
 - ___ a. On the Windows toolbar, click **Mozilla Thunderbird** to start it.



- ___ b. Select the **SIB Trade Reporter Supervisor** user name and click **Start Thunderbird**.



- ___ c. Enter tradeSupervisor for the password.

- ___ d. Open the new email message with the subject: **Trade: 1234 Details** and verify that it includes all of the correct information, including the Quandl stock price and date data.

From taskBot@sib.com <taskBot@sib.com>☆

Subject **Trade: 1234 Details**

To Me☆

Dear Trade Reporter Supervisor,

Task Bot has finished evaluating data for Trade: **1234** and has determined all required fields are present. Please see below for details:

Trade Stock Symbol: **IBM**
 Number of Shares: **10**
 Trade Stock Price: **151.91**
 Trade Stock Date: **2018-03-27**
 Total Trade Amount: **1519.10**
 Trade Reporter Name: **Hans Wegman**
 Reporting Country: **U.S.**
 Party1: **MyBank**
 Party2: **SIB**

Trade Description: **Please buy 10 shares**

8.2. Scenario 2: Testing an alternative stock symbol

In this section, you test the second scenario by using a different stock symbol when you send the Quandl request.

- ___ 1. Set up the bot for the Scenario: Alternative stock symbol test.
 - ___ a. Go to the following directory:
C:\labfiles\trade_booking_advanced\trade_receipts_scenarios\AAPL_stock_symbol
 - ___ b. Copy `Trade_Receipt_MyBank.pdf` and paste it into the `C:\labfiles\trade_booking` folder to replace the existing PDF.
- ___ 2. On the toolbar, click **Run** to start the bot.
- ___ 3. Confirm that the SIB Trade Reporter Supervisor received the email message.
 - ___ a. Go back to the SIB Trade Reporter Supervisor email account in Mozilla Thunderbird.

- b. Open the new email message with the subject: **Trade: 1234 Details** and verify that it includes all of the correct information, including the Quandl stock price and date data for the alternative stock symbol.

Inbox

Trade: 1234 Details - Inbox

Get Messages Write Chat Address Book Tag Quick Filter Search <Ctrl+K>

From taskBot@sib.com <taskBot@sib.com>☆

Subject **Trade: 1234 Details**

To Me☆

Dear Trade Reporter Supervisor,

Task Bot has finished evaluating data for Trade: **1234** and has determined all required fields are present. Please review the following details:

Trade Stock Symbol: **AAPL**
 Number of Shares: **10**
 Trade Stock Price: **168.34**
 Trade Stock Date: **2018-03-27**
 Total Trade Amount: **1683.40**
 Trade Reporter Name: **Hans Wegman**
 Reporting Country: **U.S.**
 Party1: **MyBank**
 Party2: **SIB**

Trade Description: **Please buy 10 shares**



Note

To test other stock symbols, you can use a Rest Test bot that was prepared for this course. It is in the following folder: C:\labfiles\trade_booking_advanced\solution.

8.3. Scenario 3: Handling a REST error

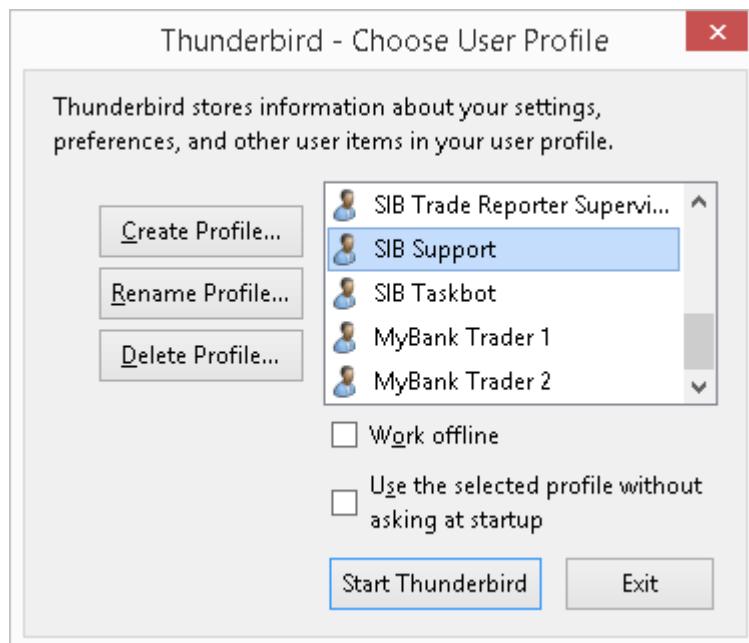
In this test, you cause the Quandl service to return an error so that you can see how the bot handles REST errors.

After running the bot, you log in to Thunderbird as the SIB Support team to view the error notification email.

- __ 1. Set up the bot for the [Scenario: Handling a REST error](#) test.
 - __ a. Go to the following directory:
C:\labfiles\trade_booking_advanced\trade_receipts_scenarios\bad_stock_sym
bol
 - __ b. Copy Trade_Receipt_MyBank.pdf and paste it into the C:\labfiles\trade_booking folder to replace the existing PDF.
- __ 2. On the toolbar, click **Run** to start the bot.
- __ 3. Close Thunderbird.
- __ 4. Restart Thunderbird, and log in as the SIB Support team.
 - __ a. On the desktop, double-click the **Mozilla Thunderbird** shortcut.



- __ b. Select the **SIB Support** account, and click **Start Thunderbird**.

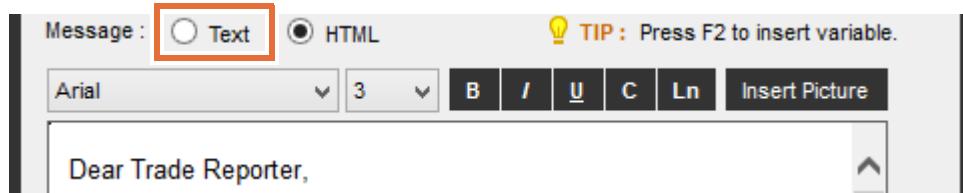


- __ c. Enter support for the password.
- __ 5. In the SIB Support email Inbox, open the email with subject: **ERROR in Trade: 1234**.
- __ 6. Review the email and verify that the message lists the trade ID number and the error code.



Troubleshooting

If you have issues with email that does not display correctly, you can select the **Text** option in the Send Email command to avoid formatting issues.

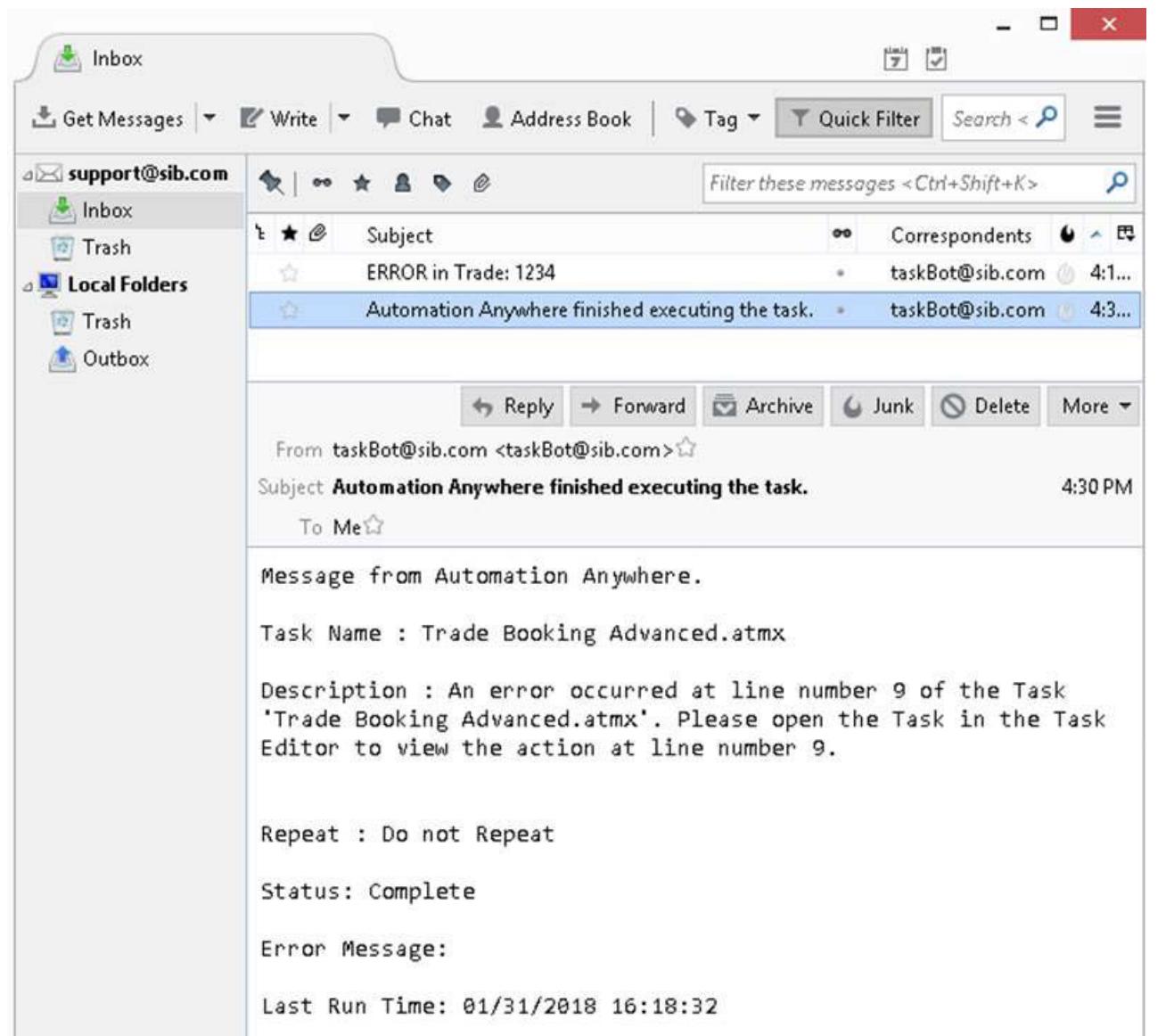


8.4. Scenario 4: Handling a general bot error

In this scenario, you test the general error handling capabilities of the Error Handling command that you implemented in this bot.

Here, you cause an error by deleting the trade receipt PDF. Because the bot cannot find the specified PDF, it treats this situation as an error and stops running.

- ___ 1. Set up the bot for the Scenario: General error test.
 - ___ a. Go to the following directory: C:\labfiles\trade_booking
 - ___ b. Delete the `Trade_Receipt_MyBank.pdf` file.
- ___ 2. On the toolbar, click **Run** to start the bot.
- ___ 3. Confirm that the SIB Support email account received the email message.
 - ___ a. In Mozilla Thunderbird, go back to the SIB Support email account.
 - ___ b. Open the new email message with the subject: **Automation Anywhere finished executing the task**
 - ___ c. Review the email message. It contains basic information about the error, and where the error occurred in the Actions List.



Section 9. *Optional.* Viewing the solution file

- __ 1. Go to C:\labfiles\trade_booking_advanced\solution directory, right-click the Trade Booking Advanced solution.atmx file, and click **Edit**.
The file opens in the WorkbenchActions List.
- __ 2. If you want to load the solution file from the Enterprise Client, move the Trade Booking Advanced solution.atmx file to the following directory:
C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks
The Enterprise Client **My Tasks** view automatically accesses files in that folder.
From the **My Tasks** list, you can either run the bot by double-clicking it or you can view it by right-clicking the task and clicking **Edit**.

End of exercise

Exercise review and wrap-up

In this exercise, you extended the Trade Booking bot from Exercise 6. First, you added global error handling to the bot. You then added commands to send a REST web service call, and to handle the data that was returned. Next, you implemented more conditional logic to evaluate and handle decimal values for the REST data. Finally, you tested the new features in the Trade Booking bot by running four different testing scenarios.

Exercise 10. Hardening the Account Opening bot

Estimated time

02:30

Overview

This exercise demonstrates how to “harden” a bot against potential failure points and implement error handling.

Objectives

After completing this exercise, you should be able to:

- Implement bot commands that harden the bot against anticipated failure points and known issues that can affect successful task automation
- Implement Error Handling commands that address specific error situations
- Create a reusable error handling bot

Introduction

In this exercise, you are provided a set of requirements for the Account Opening bot so that it responds to errors and known issues that are related to the source spreadsheet. As part of the strategy to “harden” the bot against potential failure points, you centralize error handling by creating a separate bot that starts when the main bot encounters an error.

Scenario:

While the automation team develops bots, the business team identified more requirements for the Account Opening bot. A MyBank business analyst discovered the following aspects about the account spreadsheets that are used by the bot:

- The format of the Excel spreadsheet can vary depending on the source.
- The format changes included the following situations:
 - The Excel spreadsheets are saved as Microsoft Office 2003 files (.xls) instead of the expected Microsoft Office 2010 files (.xlsx).
 - The spreadsheet format changes, and **Customer ID** is the second column instead of the first.
- At times, the data in the spreadsheet is corrupted.

- If the spreadsheet data is corrupted, the bot needs to detect the error and inform a new team that deals with the data integrity of the integration efforts.
- Finally, the MyBank Analyst also mentioned the database that is used for the cross-reference is being updated, and password resets might occur before the bot code can be updated.
 - If this situation occurs, the bot needs to detect the error and inform the user security department directly so that the bot can be updated immediately.

The IT department also mentioned that they want to implement a reusable bot to handle errors. This bot can then be shared across all bots to standardize error handling. The IT department is interested in handling SQL errors, and suggested that the bot might also be used in other departments that might encounter similar SQL errors.

For this exercise, you “harden” the original Account Opening bot to account for the situations that the business team identified. While you implemented basic error handling in the last exercise, for this exercise, you implement more complex error handling commands that address specific error situations. These commands include actions such as logging, and configuring the email notification that is sent when the error occurs.

In this exercise, you complete the following tasks:

- Build a reusable bot to handle errors
- Implement bot commands that handle the following situations by addressing the error, and then directing the task to continue running:
 - The spreadsheet file has a different extension (.xls)
 - The spreadsheet file columns are formatted differently
- Implement error handling commands that detect the following SQL errors and notify the SIB team:
 - SQL error 22001: Notify the SIB Data Integrity team (dataIntegrity@sib.com)
 - SQL error 08001: Notify the SIB User Security team (userSecurity@sib.com)
- Detect all other errors and send an error description to the SIB Support team (support@sib.com)

This exercise includes these sections:

- [Section 1, "Preparing to code bots"](#)
- [Section 2, "Preparing the bot hardening code"](#)
- [Section 3, "Preparing the error handling code"](#)
- [Section 4, "Building the error handling bot"](#)
- [Section 5, "Responding to errors"](#)
- [Section 6, "Configuring the trigger for the Handle Errors bot"](#)
- ["Appendix: Testing bots"](#)
- ["Appendix: Logging in to Thunderbird"](#)

Use case to Spreadsheet Matrix:

To simulate the following use cases, this lab uses three different spreadsheets, which are in the C:\labfiles\hardening_lab folder.

Use Case	Spreadsheet	Expected Bot Handling
Happy path (default scenario)	GOOD - new_customers	Normal processing
Column format is different	BAD FORMAT - new_customers	Move column to appropriate place, continue processing
Extension is .xls	GOOD - new_customers	Rename the file, continue processing
Bad data	BAD DATA - new_customers	Detect the error, inform the Data Integrity team (SQL error)
DB2 password change	GOOD - new_customers	Detect the error, inform the User Security team (SQL error)
All other errors	GOOD - new_customers	Detect the error, inform the Support team

Exercise tasks

The tasks that you complete in this exercise coding is organized in the following sections:

- Prepare to code bots
 - Copy and create bot files
 - Define variables
- Prepare hardening code
 - Rename the .xls spreadsheet to have an .xlsx extension
 - Get the location of the **Customer ID** column
 - If needed: Move the **Customer ID** column
 - Run the bot (optional)
 - Paste the new code into the `Account_Opening with Hardening.atmx` bot
 - Run the bot (optional)
- Prepare error handling code
 - Add global error handling to the `Account_Opening with Hardening.atmx` bot
 - Build a separate error handling bot (`Handle Errors.atmx`) that includes the following actions:
 - Open a log file and copy a log entry
 - If the bot encounters an SQL error: Extract SQL error code and bot name
 - Respond to errors

- Configure a trigger so the `Handle Errors.atmx` bot runs when needed

This exercise also includes the following two appendixes:

1. ["Appendix: Testing bots"](#)

- ["Excel Scenarios"](#)

- ["Excel file is present, and has the correct extension \(.xlsx\) and column format"](#)
 - ["Excel file has the wrong extension \(.xls\)"](#)
 - ["Excel file has incorrect column format \(Customer ID is in the wrong column\)"](#)

- ["Handling error scenarios"](#)

- ["SQL 08001 error"](#)
 - ["SQL 22001 error"](#)
 - ["All other errors"](#)

2. ["Appendix: Logging in to Thunderbird"](#)

Section 1. Preparing to code bots

In this section, you prepare two bots for coding, and you also create the variables that are used in the separate error-handling bot.



Note

You build the error handling code in a separate bot for ease of use. In subsequent sections of this exercise, you build the bot code for the Excel scenarios in a separate bot file. These commands are added later to the Account Opening bot.

1.1. Opening the “start” file for the bot

- ___ 1. In Windows Explorer, open the `C:\labfiles\hardening_lab` directory.
- ___ 2. Right-click the `Account Opening with Hardening.atmx` file, and click **Copy**.
- ___ 3. Paste the `Account Opening with Hardening.atmx` file in the `C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks` directory.
- ___ 4. Go to the Enterprise Client main window.
If the Enterprise Client is not running, double-click the **AA Enterprise Client** desktop shortcut and sign in as `devUser1` as described in [Section 1.1, "Signing in to the Enterprise Client,"](#) on page 2-2.
- ___ 5. Verify that `Account Opening with Hardening.atmx` is listed in **My Tasks**.
- ___ 6. Right-click the `Account Opening with Hardening.atmx` file and click **Edit**.

1.2. Creating a bot file

- ___ 1. On the toolbar of the Enterprise Client, click **New** to create a bot.
- ___ 2. In the Automate window, click **Workbench**.
- ___ 3. Open the Variable Manager, and click **Add**.
- ___ 4. Define a variable with these attributes:
 - **Type:** List
 - **Name:** vCustomerIdCell
 - **List Value:** XX
- ___ 5. Click **Add To List**.
- ___ 6. Click **Save**.



Information

This variable houses the location of the column labeled Customer ID. It is created as a List type because there might be more than one location when using a Find/Replace Excel command.

-
- ___ 7. Save the bot.
 - ___ a. Click **Save** on the toolbar.
 - ___ b. In the Save Task window, enter a name for the bot: `Excel File Prep`
 - ___ c. Click **Save**.

Section 2. Preparing the bot hardening code

In this section, you add and configure commands so that the bot handles the issues that are related to the Excel spreadsheet:

- If the spreadsheet file has the incorrect extension, then the bot renames the file and continues processing.
- If the column format is incorrect, the bot moves the **Customer ID** column to the correct location and continues processing.

These actions are completed in the following sections:

- Incorrect extension:
 - Rename a spreadsheet that has an `.xls` extension so that the extension is `.xlsx`.
- Incorrect column format:
 - Determine the location of the **Customer ID** column.
 - If needed, move the **Customer ID** column.

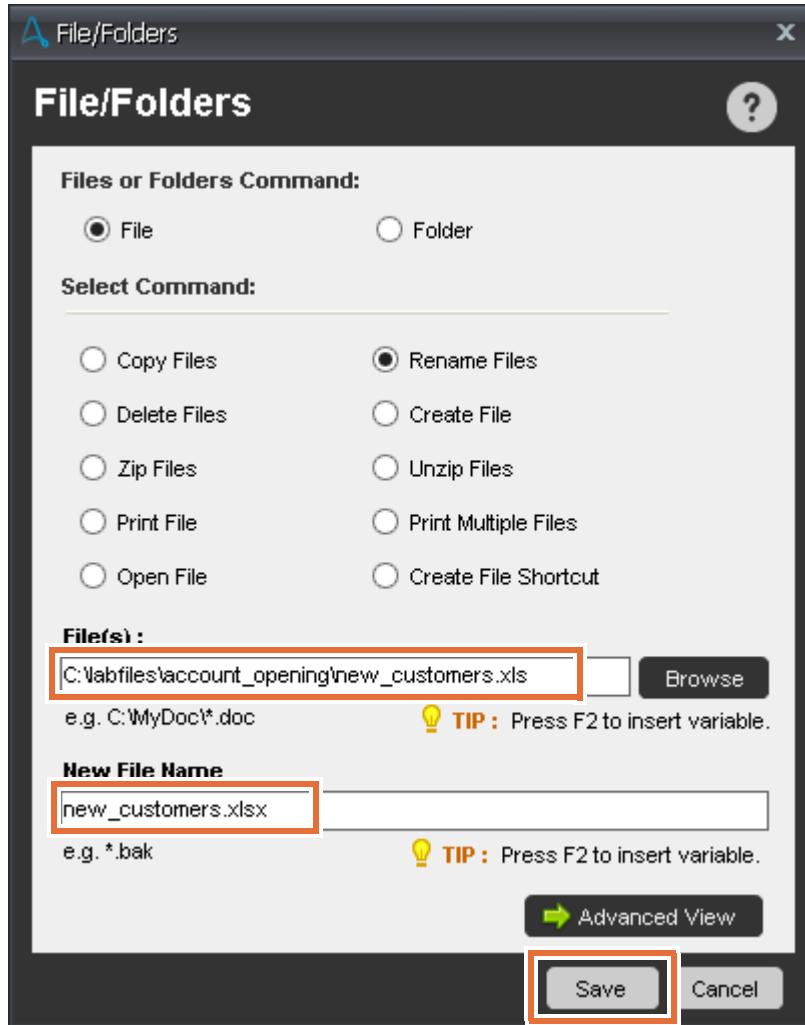
After the error handling commands are complete, you copy the code to the `Account_Opening_with_Hardening.atmx` file.

2.1. Spreadsheet scenario 1: Renaming the spreadsheet extension

To handle the scenario where the spreadsheet has an `.xls` extension, the bot renames the spreadsheet so that it has an `.xlsx` extension. The bot then processes the file as expected.

- ___ 1. In the Actions List for `Excel File Prep.atmx`, add the following comment:
Evaluate and prepare file for processing
- ___ 2. Add If/Else commands to evaluate whether the spreadsheet has an `.xls` extension.
 - ___ a. In the **Commands** list, expand the **If/Else** section and double-click the **File Exists** command.
 - ___ b. In the **Select File** field, enter:
`C:\labfiles\account_opening\new_customers.xls`
 - ___ c. Click **Save**.
- ___ 3. Change the default **Comment** added after the **If File Exists** command to:
Rename a spreadsheet with an `.xls` extension to `.xlsx`
- ___ 4. Add a Files/Folders command to rename the spreadsheet file after the `Rename spreadsheet` comment.
 - ___ a. Expand the **Files/Folders** section and double-click **Rename Files**.
 - ___ b. In the **File(s)** field, enter:
`C:\labfiles\account_opening\new_customers.xls`
 - ___ c. In the **New File Name** field, enter: `new_customers.xlsx`

- ___ d. Click **Save**.



- ___ 5. Save your work.

2.2. Getting the location of the Customer ID column

To handle the scenario where **Customer ID** is the second column of the `new_customers.xlsx` spreadsheet, the bot finds **Customer ID** column and copies it and pastes it to the correct location in the spreadsheet. You use the Smart Recorder to implement this action.

However, before it performs the copy and paste action, the bot evaluates whether the spreadsheet file exists and whether it contains the correct file extension.

- ___ 1. Add the following comment after the **Rename Files** command:

Check to see whether the spreadsheet exists and has a `.xlsx` extension

- ___ 2. Add an **Else If** command to the **If File Exists** section to verify that the file exists, and that it has an `.xlsx` extension.

- ___ a. Under the **If/Else** section of the **Commands** list, double-click the **Else If** command.

- ___ b. In the **Select Condition** section, make sure that **File exists** is selected.

- ___ c. In the **Select File** field, enter:
C:\labfiles\account_opening\new_customers.xlsx
 - ___ d. Click **Save**.
- ___ 3. Add a comment below the Else If command to read:
- Open the spreadsheet and get the location of the Customer ID column
- ___ 4. Add an Excel command to open the spreadsheet.
- ___ a. In the **Commands** list, expand the **Excel** section and double-click **Open Spreadsheet**.
 - ___ b. In the **Spreadsheet Path** field, enter:
C:\labfiles\account_opening\new_customers.xlsx
 - ___ c. Ensure that **Contains Header** is not selected.
 - ___ d. Click **Save**.



Information

To determine the location of the **Customer ID** column, the spreadsheet needs to be opened with the header included. For processing after preparing the Excel file, the Open Spreadsheet command is performed again with the **Contains Header** option selected.

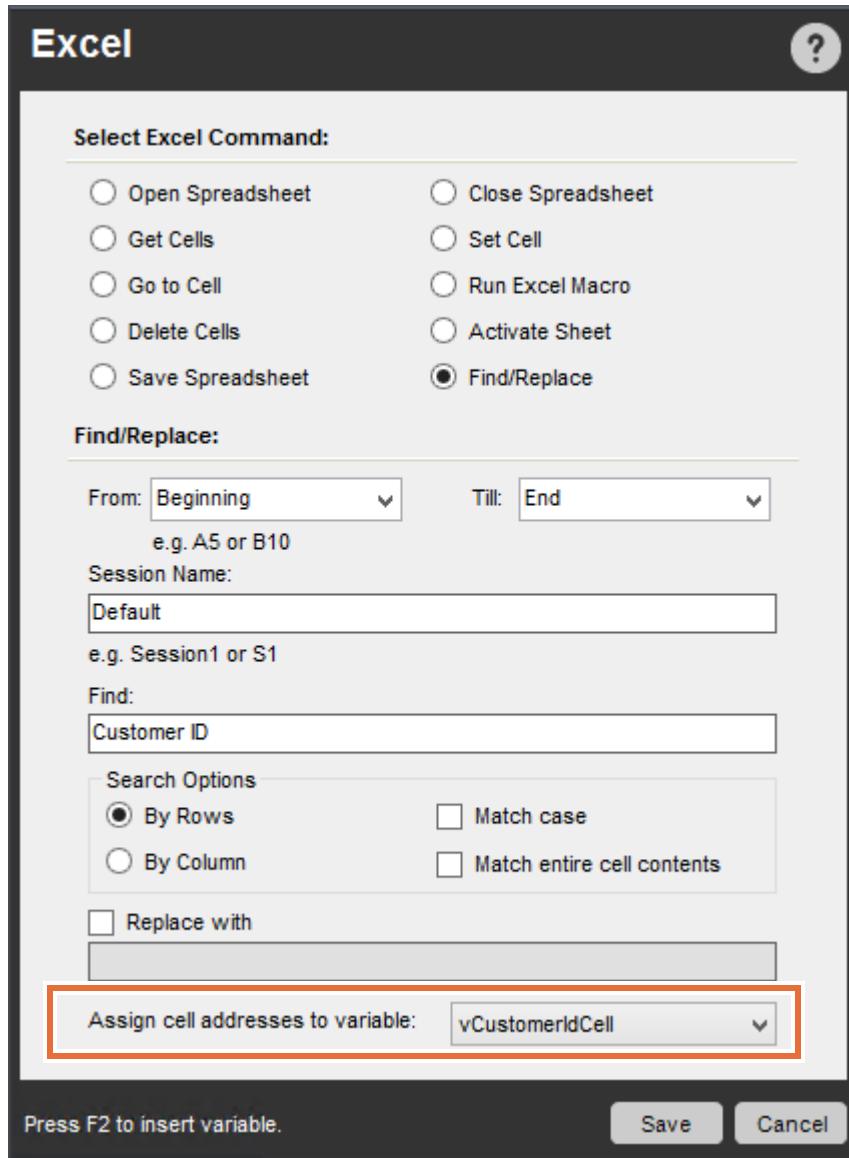
-
- ___ 5. Add a command to find the **Customer ID** column.
 - ___ a. In the **Excel** section of the **Commands** list, double-click the **Find/Replace** command.
 - ___ b. In the **Find** field, enter: Customer ID



Important

When you type Customer ID in the field, make sure that you do not include any trailing or leading spaces. Including a trailing or leading space can lead to errors in how the bot runs.

- ___ c. From the **Assign cell address to variable** list, select **vCustomerIdCell**.



- ___ d. Click **Save**.
___ 6. Save your work.

2.3. Moving the Customer ID column

In this section, you configure the bot so that if the **Customer ID** column is in the wrong location, the bot finds the column, copies it, and inserts it in the correct location. These actions are accomplished by adding nested **If/Else** commands to evaluate the location of the **Customer ID** column. You then use the Smart Recorder to capture and perform the column move.

- ___ 1. In the **If/Else** section of the **Commands** list, double-click the **Variable** command.
- ___ 2. Go to the **IF Condition** field, and click **Edit** to add a condition.
- ___ 3. In the If Variable window, click the **Variable** field, and press F2 (or Fn+F2) to insert a variable.

- ___ 4. In the Insert Variable window, select **vCustomerIdCell** and click **Insert**.
- ___ 5. From the **Operator** list, select **Not Equal To (<>)**.
- ___ 6. For the Value section, keep the **Fix** option selected, and in the **Fix** field, enter: A1
- ___ 7. In the If Variable window, click **Save**.
- ___ 8. Edit the default comment for the `If $vCustomerIdCell$` command to read:

Move the Customer ID column

2.4. Preparing the spreadsheet file for processing

- ___ 1. Open Windows Explorer, and in the `C:\labfiles\account_opening` directory, delete the `new_customers.xlsx` file.
 - ___ 2. Go to `C:\labfiles\hardening_lab` directory.
 - ___ 3. Copy `BAD FORMAT - new_customers.xlsx` and paste it into:
`C:\labfiles\account_opening`
 - ___ 4. Rename the file to: `new_customers.xlsx`
 - ___ 5. Open the spreadsheet in Microsoft Excel, and maximize it.
-



Information

This version of the `new_customers.xlsx` file has the **Customer ID** column as the second column. This file is used to record the actions that move the **Customer ID** column to the correct location as the first column in the spreadsheet.

- ___ 6. Add a command to maximize the Excel window.
 - ___ a. Return to the Workbench and make sure that the `Move the Customer ID column` comment is selected.
 - ___ b. Expand the Window Actions section of the **Commands** list, and double-click the **Maximize Window** command.
 - ___ c. From the **Select Window** list, select **Microsoft Excel – new_customers.xlsx**.
 - ___ d. Click **Save**.
- ___ 7. Save your work.

2.5. Using the Smart Recorder to select and move the Column ID column



Information

Object cloning commands are inserted after the selected line in the Actions List. Make sure that **Maximize Window** is selected. You can also position the instructions before you start the recording so that you can avoid using mouse or keyboard movements to move the document during the recording process.

- ___ 1. On the toolbar, click **Record**.
- ___ 2. From the **Select Window** list, select **Microsoft Excel – new_customers.xlsx**.
Your recording should capture you selecting the B column with your mouse, pressing **Ctrl+X** to cut the column, then inserting that column to the left of the A column.
- ___ 3. Click **Start**.
- ___ 4. Select the **Customer ID** (B1) column by moving your mouse above the column, and when your mouse pointer changes into the black down arrow, click the column to select it.

	A	B	C	D	E
1	First Name	Customer ID	Last Name	Account Number	Account Type
2	Jason	345789	Watson	4563214569	Savings
3	Mallory	987345	Brighton	9651245896	Deposit
4	Walter	564123	Mason	7426395648	Loan
5	Jodie	956412	James	5621345698	Credit

- ___ 5. Press **Ctrl+X** to cut the column. The column is highlighted until it is pasted.
- ___ 6. Click the **First Name** (A1) column to select it.
- ___ 7. Press **Ctrl+Shift+Plus (+)** to insert the column. The **Customer ID** column is inserted to the left of the **First Name** column, making it column A.

- ___ 8. In the Recorder window, click **Stop**. Four Object Cloning commands are inserted into the Workbench:

Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type: Left Click; Source: Window; Play Type: Object

Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Value: "[CTRL UP]x[CTRL DOWN]"; Source: Window; Play Type: Object

Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type: Left Click; Source: Window; Play Type: Object

Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Value: "[CTRL DOWN][SHIFT DOWN]=[SHIFT UP][CTRL UP]"; Source: Window; Play Type: Object

- | | |
|----|--|
| 13 |  Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type |
| 14 |  Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Value |
| 15 |  Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type |
| 16 |  Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Value |

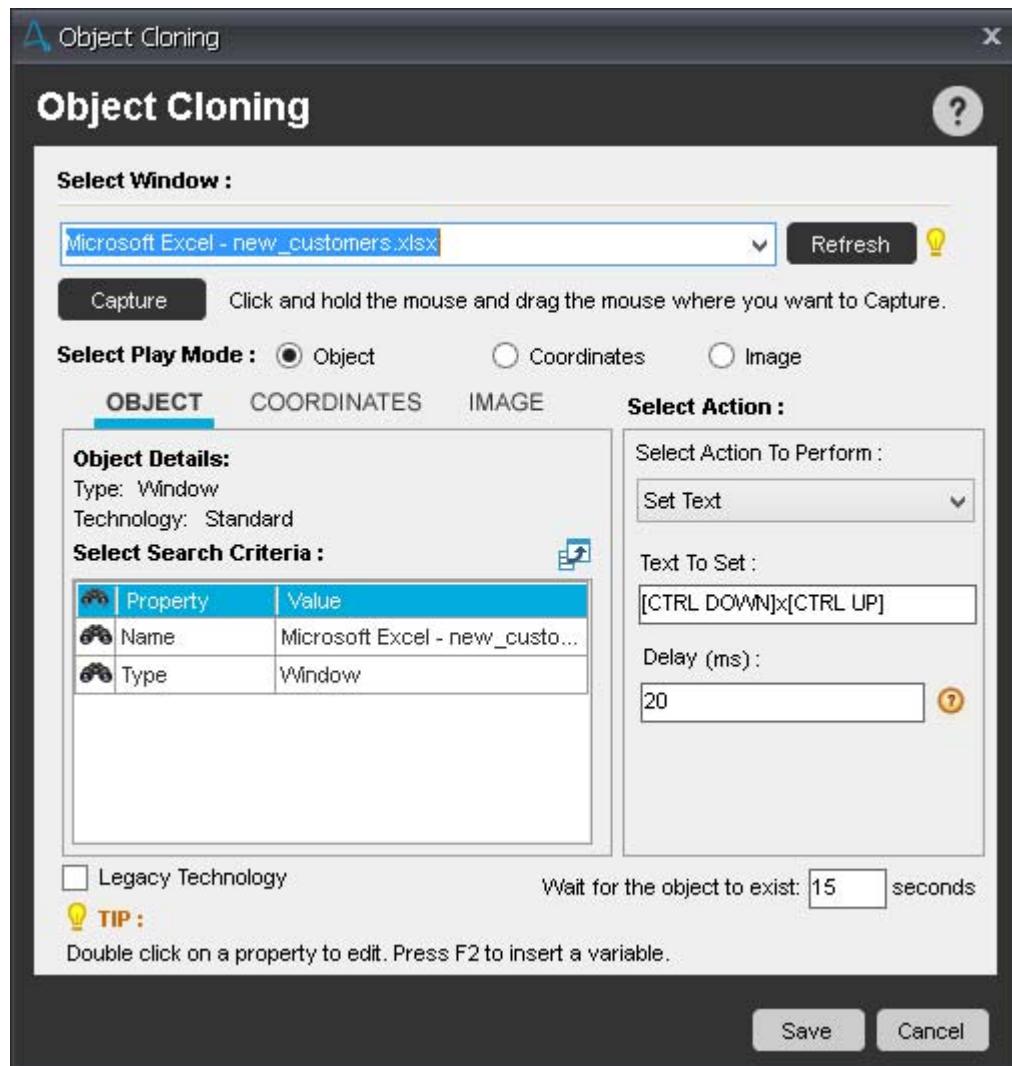


If you recorded extra Object Cloning commands, you can delete them from the Actions List.



If you hold the Ctrl key while recording, the Smart Recorder can record multiple [CTRL DOWN] events. If you record extra keystroke events and the **Text To Set** property for the **Set Text** events do not match the actions that you need, you must edit the **Text To Set** values.

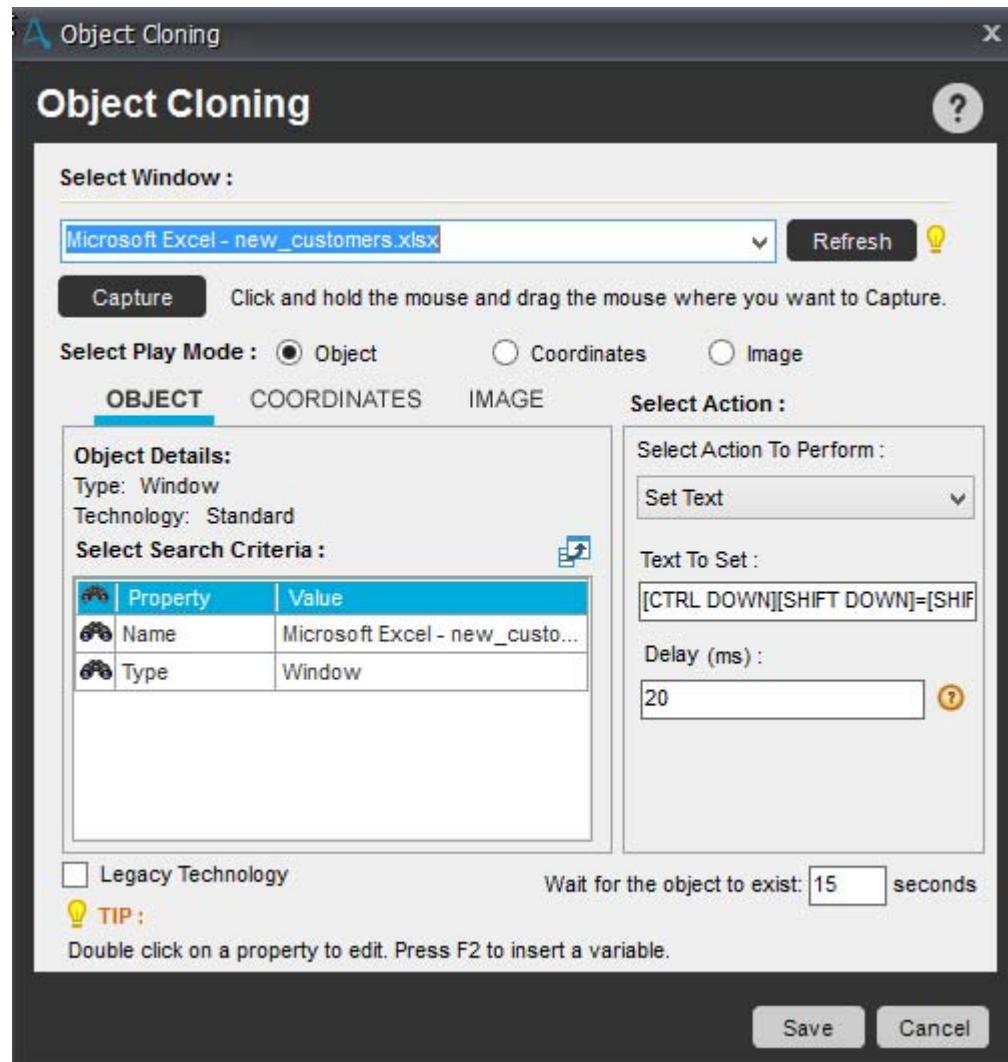
- ___ 9. If needed, update the **Text To Set** for the cut column action that you recorded.
- ___ a. Double-click the second object cloning command (line 14) to open the Object Cloning window. (The code includes two **Set Text** object cloning commands, and you need to modify the first of these commands.)
 - ___ b. In the Select Window section, make sure that **Microsoft Excel – new_customers.xlsx** is selected.
 - ___ c. In the Select Action section, go to the **Text To Set** field and replace the value with the following text:
[CTRL DOWN]x[CTRL UP]



- ___ d. Click **Save**.
- ___ 10. If needed, edit the **Set Text** command for the insert action that you recorded.
 - ___ a. Double-click the second **Set Text** object cloning command (line 16) to open the Object Cloning window.
 - ___ b. In the Select Window section, make sure that **Microsoft Excel – new_customers.xlsx** is selected.
 - ___ c. In the Select Action section, go to the **Text To Set** field and replace the value with the following text:

[CTRL DOWN][SHIFT DOWN]=[SHIFT UP][CTRL UP]

__ d. Click **Save**.



__ 11. Save your work.

This screen capture shows the bot code that you worked on in this section.

```

1  Comment: Evaluate and prepare file for processing
2  IF If File Exists ("C:\Nabfiles\account_opening\new_customers.xls") Then
3    Comment: Rename spreadsheet with a .xls extension to .xlsx
4    Rename Files "C:\Nabfiles\account_opening\new_customers.xls" to "new_customers.xlsx"
5    Comment: Check to see whether the spreadsheet exists and has a .xlsx extension
6  EI Else If File Exists ("C:\Nabfiles\account_opening\new_customers.xlsx") Then
7    Comment: Open the spreadsheet and get the location of the Customer ID column
8    Excel: Open Spreadsheet "C:\Nabfiles\account_opening\new_customers.xlsx". ActiveSheet: "Default". Session: Default
9    Excel: Find "Customer ID"; Range : "All Cells"; Assign to : "$vCustomerIdCell$"; Session: Default
10   IF If $vCustomerIdCell$ Not Equal To (<>) "A1" Then
11     Comment: Move the Customer ID column
12     Excel: Maximize Window: "Microsoft Excel - new_customers.xlsx"
13     Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type: Left Click
14     Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_cu
15     Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xlsx'; Click Type: Left Click
16     Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Excel - new_cu
17   NF End If
18   NF End If

```

2.6. Copying the Excel bot code into the Account Opening with Hardening bot

In this section, you copy the Excel bot code that you completed in the previous section into the Account Opening with Hardening bot.

You paste the Excel bot code at the beginning of the Account Opening with Hardening.atmx file Actions List. You also copy and paste the vCustomerIdCell local variable from the Excel File Prep bot to the Account Opening with Hardening bot.

- ___ 1. Open the Account Opening with Hardening bot.
 - ___ a. Go back to the main Enterprise Client window.
 - ___ b. In the My Tasks list, click **Account Opening with Hardening.atmx** to select it.
 - ___ c. On the toolbar, click **Edit**.

The bot opens in the Workbench.
- ___ 2. Copy the Excel bot code into the Actions List for the Account Opening with Hardening bot.
 - ___ a. Go back to the Workbench, and click the **Excel File Prep** tab.
 - ___ b. Press Ctrl+A to select all of the code in the Actions List, and then press Ctrl+C to copy it.
 - ___ c. In the Workbench, click the **Account Opening with Hardening** tab.
 - ___ d. In the Actions List for the Account Opening with Hardening bot, click the first line.
 - ___ e. Press Ctrl+V to past the Excel bot code into the Account Opening with Hardening bot.
The code is pasted after the first line.
 - ___ f. Click the first line and drag it to the last line of code that you inserted (the second **End If** command) to place it at its correct location.

- ___ 3. Copy the vCustomerIdCell variable from the Excel File Prep bot to the Account Opening with Hardening bot.
 - ___ a. In the Workbench, go to the **Excel File Prep** bot.
 - ___ b. If needed, click the **VARIABLE MANAGER** tab to open it.
 - ___ c. In the Local Variables list, click **vCustomerIDCell**, and then click **Copy**.
 - ___ d. Go to the **Account Opening with Hardening** bot.
 - ___ e. If needed, open the Variable Manager.
 - ___ f. In the Variable Manager, click **Paste**.
 - ___ g. On the toolbar, click **Save** to save the bot.

2.7. *Optional. Testing the bot*

After you copy the Excel bot code to the Account Opening with Hardening bot, you can test the Excel scenarios. For more information about testing the scenarios, see the "["Appendix: Testing bots"](#) on page 33 for this exercise. At this point, you can test the scenarios in ["Section 1, "Excel Scenarios,"](#) on page 10-33.

Section 3. Preparing the error handling code

In this section, you add error handling to the Account Opening with Hardening bot. You start by adding and configuring a global error handling command to the Actions List.

In these steps, you add global error handling to the bot and configure it to log errors. If the bot encounters errors, it writes the errors to the log file. You then use the log file to configure a bot trigger later in this exercise: changes to the log file trigger the Handle Errors bot that you build in the next section.

- ___ 1. In the **Commands** list, expand the **Error Handling** section, and drag the **Begin Error Handling** command to the first line in the bot.
- ___ 2. In the Select Error Handling Action section, select **Stop Task**.
- ___ 3. In the Select Error Handling Options section, select **Log Data into File**. A section then opens where you can configure the log file.
- ___ 4. In the **Log File** field, enter the following file path:

C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Tasks Error Log.txt

- ___ 5. In the **Text** field, enter the following information:

Error: \$Error Description\$ Task: \$AATaskName\$



Note

Both \$Error Description\$ and \$AATaskName\$ are system variables.

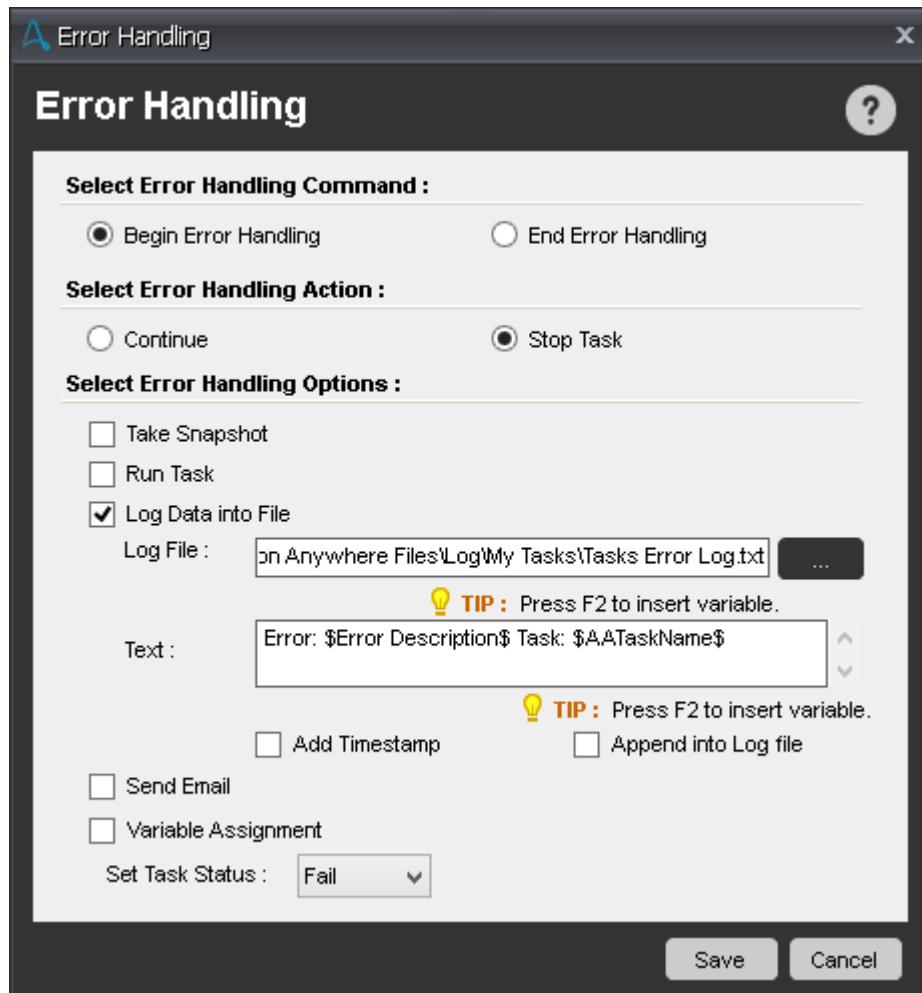
- ___ 6. Clear the **Append into Log file** check box.



Information

When you clear this option, the log file does not retain a historical log of all errors. Instead, only one error is logged in the file, and subsequent errors overwrite any existing error that is in the file. This simplifies the log for testing purposes so that you do not have to scroll through long logs.

- ___ 7. Click **Save**.



- ___ 8. Drag the Evaluate and prepare file for processing comment after the Begin Error Handling command.
- ___ 9. Drag the End Error Handling command to the last line of the bot. The Account Opening with Hardening bot is now covered by the global error handling command.
- ___ 10. Delete the Comment command that was added with the Begin Error Handling command.
- ___ 11. Save your work.

The following screen capture shows the Excel bot code and the Begin Error Handling command that were added to the bot.

```

1  ⚠ Begin Error Handling; Action: Stop Task; Options: Log to File, Task Status: Fail
2  🟢 Comment: Evaluate and prepare file for processing
3  IF 🟢 If File Exists ("C:\labfiles\account_opening\new_customers.xls") Then
4    🟢 Comment: Rename a spreadsheet with a .xls extension to .xlsx
5    F Rename Files "C:\labfiles\account_opening\new_customers.xls" to "new_customers.xlsx"
6    🟢 Comment: Check to see whether the spreadsheet exists and has a .xlsx extension
7  EI 🟢 Else If File Exists ("C:\labfiles\account_opening\new_customers.xlsx") Then
8    🟢 Comment: Open the spreadsheet and get the location of the Customer ID column
9    X Excel: Open Spreadsheet "C:\labfiles\account_opening\new_customers.xlsx". ActiveSheet: "Default". $ 
10   X Excel: Find "Customer ID"; Range : "All Cells"; Assign to : "$vCustomerIdCell$"; Session: Default
11  IF 🟢 If $vCustomerIdCell$ Not Equal To (<>) "A1" Then
12    🟢 Comment: Move the Customer ID column
13    T Maximize Window: "Microsoft Excel - new_customers.xlsx"
14    M Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xls
15    M Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Exc
16    M Object Cloning: Click On Client "new_customers.xlsx" in window 'Microsoft Excel - new_customers.xls
17    M Object Cloning: Set Text of Window "Microsoft Excel - new_customers.xlsx" in window 'Microsoft Exc
18  X End If
19  X End If

```

Here you see the End Error Handling command.

S2	Disconnect from database Session:'Default'
S3	Excel: Save Spreadsheet. Session: NewCustomers
S4	Excel: Close Spreadsheet. Session: NewCustomers
S5	Close Window: "SIB CRM"
S6	⚠ End Error Handling

Section 4. Building the error handling bot

In this section, you create a reusable bot that reads the error log. Depending on the error, the bot sends an email to a specific SIB team.

4.1. Creating a bot and defining local variables

- ___ 1. In the Workbench toolbar, click **New** to create a bot file.
- ___ 2. Create the `vErrorCode` variable. This variable holds the SQL error code.
 - ___ a. If needed, click the **VARIABLE MANAGER** tab to open it.
 - ___ b. In the Variable Manager, click **Add**.
 - ___ c. In the **Name** field, type: `vErrorCode`.
 - ___ d. Click **Save**.
 - ___ e. When you are asked whether you want to assign a null value to the variable, click **Yes**.
- ___ 3. Repeat step 2 to define the following variables.

Variable	Description
<code>vErrorDesc</code>	This variable holds raw data that is pulled from the log file.
<code>vTaskBot</code>	This variable contains the location and name of the bot that was run.
<code>vTaskBotName</code>	This variable contains only the name of the bot that was run.

- ___ 4. Save and name the bot.
 - ___ a. In the Workbench toolbar, click **Save**.
 - ___ b. In the **Filename** field of the Save Task window, enter: `Handle Errors`.
 - ___ c. Click **Save**.

4.2. Opening the log file and copying log entries

In this section, you create the file that is used for logging. You use this log file to configure the bot commands that copy its contents to the Clipboard.

- ___ 1. Open Notepad.
 - ___ a. From the Windows Start menu, expand the list of Apps.
 - ___ b. In the Windows Accessories section, click **Notepad**.
- ___ 2. Use Notepad to save a text file to use for logging.
 - ___ a. In Notepad, click **File > Save As**.
 - ___ b. In the Save As window, go to the following folder:
`C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks`

- ___ c. In the **File name** field, enter: Tasks Error Log.txt.
 - ___ d. Click **Save**.
 - ___ 3. Return to the Workbench, and go to the Handle Errors bot.
 - ___ 4. In the Handle Errors bot Actions List, add the following comment:
Open the log file and copy the log entry
 - ___ 5. Add a command to open the Tasks Error Log.txt file.
 - ___ a. In the Files/Folders section of the **Commands** list, double-click the **Open File** command.
 - ___ b. In the **Open File Name** field, enter the following file path:
C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Tasks Error Log.txt
 - ___ c. Click **Save**.
 - ___ 6. Add a command to copy the contents of the log file.
 - ___ a. In the **Commands** list, double-click the **Insert Keystrokes** command.
 - ___ b. From the **Select Window** list, select **Tasks Error Log.txt – Notepad**.
 - ___ c. Enter the following text in the **Keystrokes** value:
[CTRL DOWN]a[CTRL UP][CTRL DOWN]c[CTRL UP]
-



Information

These keystrokes perform the Ctrl+A (select all) and Ctrl+C (copy) actions to copy the log file contents.

- ___ d. Click **Save**.
- ___ 7. Add a command to assign the copied contents of the log file from the Clipboard to the vErrorDesc variable.
 - ___ a. In the **Commands** list, expand the **Clipboard** section and double-click the **Assign From Clipboard** command.
 - ___ b. In the Select Variable section, from the **Assign value of clipboard to a variable** list, select **vErrorDesc**.
 - ___ c. Click **Save**.
- ___ 8. Save your work.

4.3. Extracting the SQL error code and the bot name

In this section, you use the Before-After String Operation command to extract the error description, the bot name, and any SQL error codes that are included.

The two SQL errors the bot detects are:

1. SQL Error 22001:

- **Error:** [DB2/NT64] SQL0433N Value ")_(*&(*&^&\$^#%@"is too long. SQLSTATE=22001
- **Task:** C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx

2. SQL Error 08001:

- **Error:** SQL30082N Security processing failed with reason "24" ("USER NAME AND/OR PASSWORD INVALID"). SQLSTATE=08001
- **Task:** C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx



Reminder

You use the `Before-After` String Operation command to identify the substring that is located between the identified **Before** and **After** values. Here, you use the `Before-After` command to find the error code data based on the error code description text.

___ 1. Add the following comment to the Actions List:

Extract the SQL error code and task bot name

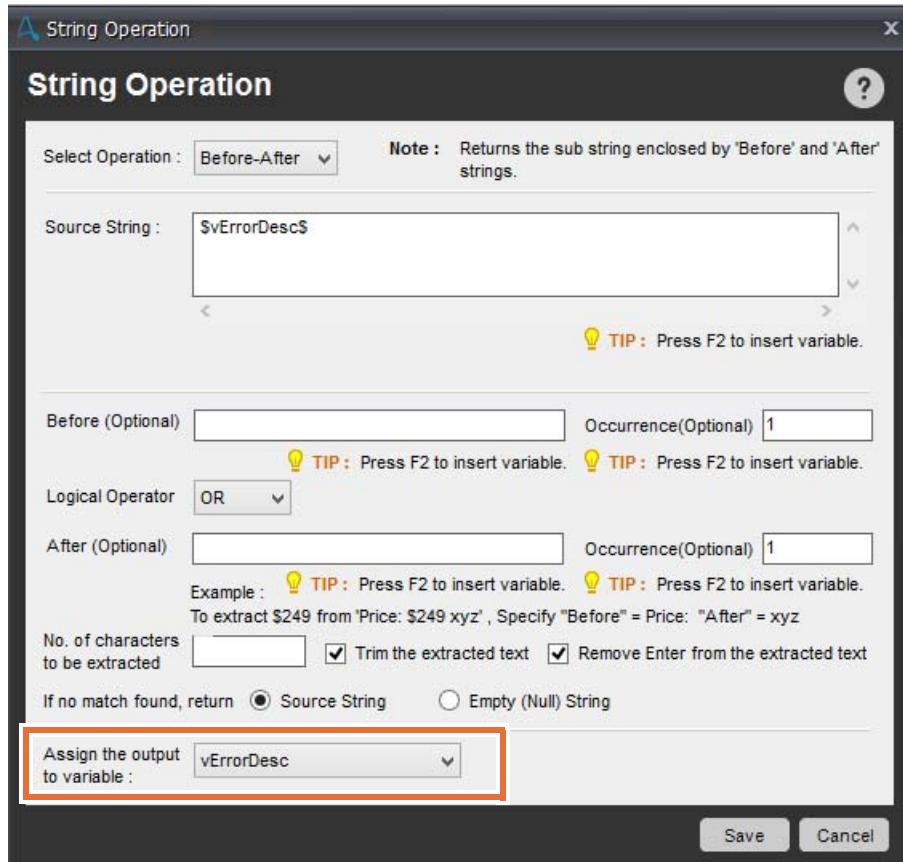
___ 2. Use the `Before-After` String Operation command to extract the error code value from the error code description, and assign it to `vErrorCode`.

___ a. In the **Commands** list, expand the **String Operation** section and double-click the **Before-After** command.

___ b. In the **Source String** field, enter: `$vErrorDesc$`

___ c. Select **Remove Enter from the extracted text**.

- ___ d. From the **Assign the output to variable** list, select **vErrorDesc**.



- ___ e. Click **Save**.
 ___ 3. Repeat step 2 to configure the remaining Before-After commands by using the following parameters:

Source String	Before (Optional)	Logical Operator	After (Optional)	Assign output to Variable
\$vErrorDesc\$	SQLSTATE=	AND	Task	vErrorCode
\$vErrorDesc\$	Task:	OR	<no value>	vTaskBot
\$vErrorDesc\$	Error:	AND	Task	vErrorDesc
\$vTaskBot\$	My Tasks\	AND	.atmx	vTaskBotName

The bot should now have a total of five **String Operation** commands.

- ___ 4. Add a command to close the Notepad window.
 ___ a. In the **Windows Actions** section of the **Commands** list, double-click the **Close Window** command.
 ___ b. From the **Select Window** list, select **Tasks Error Log.txt – Notepad**.
 ___ c. Click **Save**.
 ___ 5. Save your work.

Your code should match this screen capture.

```
1  Comment: Open the log file and copy the log entry
2  Open File "C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Tasks Error Log.txt"
3  Keystrokes: [CTRL DOWN]a[CTRL UP][CTRL DOWN]c[CTRL UP] in "Tasks Error Log.txt - Notepad"
4  Assign value of Clipboard to variable "$vErrorDesc$"
5  Comment: Extract the SQL error code and task bot name
6  String Operation: Before-After "$vErrorDesc$" and assign output to $vErrorDesc$
7  String Operation: Before-After "$vErrorDesc$"; Before: "SQLSTATE="; After: "Task" and assign output to $vErrorCode$"
8  String Operation: Before-After "$vErrorDesc$"; Before: "Task:" and assign output to $vTaskBot$"
9  String Operation: Before-After "$vErrorDesc$"; Before: "Error:"; After: "Task" and assign output to $vErrorDesc$"
10 String Operation: Before-After "$vTaskBot$"; Before: "My Tasks\"; After: ".atmx" and assign output to $vTaskBotName$"
11 Close Window: "Tasks Error Log.txt - Notepad"
```

Section 5. Responding to errors

In this section, you configure the bot so that it can determine whether the automated task encounters an SQL error. If it encounters an SQL error, depending on the error code that it receives, the bot can respond by sending an email to a specific SIB team. If the bot does not encounter any SQL errors, the bot responds to any other error type by sending an email to the SIB support team.

5.1. Responding to the SQL 08001 error

In this part of the exercise, you configure the bot to respond to the SQL 08001 error. This error is returned when the database user name or password is incorrect. The error handling commands in this section send a notification to the SIB User Security team.



Reminder

The error data that is logged for SQL Error 0 8001 is:

```
SQL30082N Security processing failed with reason "24" ("USER NAME AND/OR PASSWORD INVALID"). SQLSTATE=08001
```

- 1. Add the following comment: Respond to the SQL 08001 error
- 2. Determine whether the SQL 08001 error occurred.
 - a. In the **If/Else** section of the **Commands** list, double-click the **Variable** command.
 - b. Next to the **IF Condition** field, click **Edit**.
 - c. In the If Variable window, click the **Variable** field and press F2 (or Fn+F2).
 - d. In the Insert Variable window, select **vErrorCode** and click **Insert**.
 - e. From the **Operator** list, select **Equal To (=)**.
 - f. In the Value section, keep **Fix** selected, and in the field, enter: 08001
 - g. Click **Save**.
- 3. Delete the comment that is placed after the **If** command.
- 4. Add an **Email** command to the If condition so that the bot notifies the User Security team.
 - a. Select the **If \$vErrorCode\$ Equal To (=) "08001" Then** command.
 - b. In the Commands list, double-click the **Send Email** command.
 - c. In the **From** field, enter: taskbot@sib.com
 - d. In the **To** field, enter: userSecurity@sib.com
 - e. In the **Subject** field, enter:

Error: \$vErrorCode\$ in Task Bot: \$vTaskBotName\$

- ___ f. For the Message section, select **HTML** and enter the following text for the body of the email. You can copy the contents and paste them into the body of the email. The email message should retain the formatting.

Dear User Security team,

The following SQL error: \$vErrorCode\$ indicates that the bot login credentials are invalid. The complete error description is:

\$vErrorDesc\$

This error occurred in the following bot:

\$vTaskBot\$

Handle Errors Task Bot



Troubleshooting

If you see formatting issues in the email messages when you select **HTML** for the email message option, select **Text** instead.

- ___ g. Click **Save**.
___ 5. Save your work.

5.2. Responding to the SQL 22001 error

In this part of the exercise, you configure the bot to respond to the SQL 22001 error. This error is returned when the data is incorrect. The error handling commands in this section send an email notification to the SIB Data Integrity team.



Reminder

The error data that is logged for SQL Error 22001 is:

[DB2/NT64] SQL0433N Value ")_(*&(*^&\$^#%@"is too long. SQLSTATE=22001

- ___ 1. Make sure that the `Send Email` command is selected, add the following comment:
 Respond to the SQL 22001 error
- ___ 2. Add a command to determine whether the SQL 22001 error occurred.
- ___ a. In the Actions List, make sure that the Respond to the SQL 22001 error comment is highlighted.
- ___ b. In the **If/Else** section of the **Commands** list, double-click the `Else If` command.

- ___ c. In the Select Condition section, select **Variable**.
 - ___ d. Next to the **IF Condition** field, click **Edit**.
 - ___ e. Click the **Variable** field and press F2 (or Fn+F2).
 - ___ f. In the Insert Variable window, select **vErrorCode** and click **Insert**.
 - ___ g. From the **Operator** list, select **Equal To (=)**.
 - ___ h. In the Value section, keep **Fix** selected, and in the field, enter: 22001
 - ___ i. Click **Save**.
- ___ 3. Add a command to send an email notification to the SIB Data Integrity team.
- ___ a. Make sure that the **Else If** command is selected.
 - ___ b. In the Commands list, double-click the **Send Email** command.
 - ___ c. In the **From** field, enter: taskbot@sib.com
 - ___ d. In the **To** field, enter: dataIntegrity@sib.com
 - ___ e. In the **Subject** field, enter: Error: \$vErrorCode\$ in Task Bot: \$vTaskBotName\$
 - ___ f. In the Message section, select **HTML**, and enter the following text for the body of the email. You can copy the message and paste it into the body of the email.

Dear Data Integrity team,

The following SQL error: \$vErrorCode\$ occurred indicating an insert value is too long for the column. The complete error description is:

\$vErrorDesc\$

This error occurred in the following Bot:

\$vTaskBot\$

Thank you,
Handle Errors Task Bot



Troubleshooting

If you see formatting issues in the email messages when you select **HTML** for the email message option, select **Text** instead.

- ___ g. Click **Save**.

5.3. Responding to all other errors

In this part of the exercise, you configure the bot to send an email if any other error is logged in the log file.

- ___ 1. Make sure that the **Send Email** command for SQL error 22001 is selected, and add the following comment:
Respond to all other errors
- ___ 2. Add a command to send an email to the SIB Support team that notifies them of all other errors.
 - ___ a. Make sure that the **Respond to all other errors** comment is selected.
 - ___ b. In the **If/Else** section of the **Commands** list, double-click the **Else** command. The **Else** command is placed in the code.
 - ___ c. In the Commands list, double-click the **Send Email** command.
 - ___ d. Enter the following values in the corresponding fields and click **Save**:
 - **From:** taskbot@sib.com
 - **To:** support@sib.com
 - **Subject:** Error occurred in Task Bot: \$vTaskBotName\$
 - **Message:** Select **HTML**, and enter the following message for the body of the email. You can copy the contents and paste them into the body of the email.

Dear Support Desk,

The following error occurred while executing the \$vTaskBotName\$ Task Bot.
The error description is:

\$vErrorDesc\$

This error occurred in the following Bot:

\$vTaskBot\$

Thank you,
Handle Errors Task Bot



Troubleshooting

If you see formatting issues in the email messages when you select **HTML** for the email message option, select **Text** instead.

- ___ e. Click **Save**.
- ___ 3. Save your work.

The following screen capture shows the completed Handle Errors bot:

```

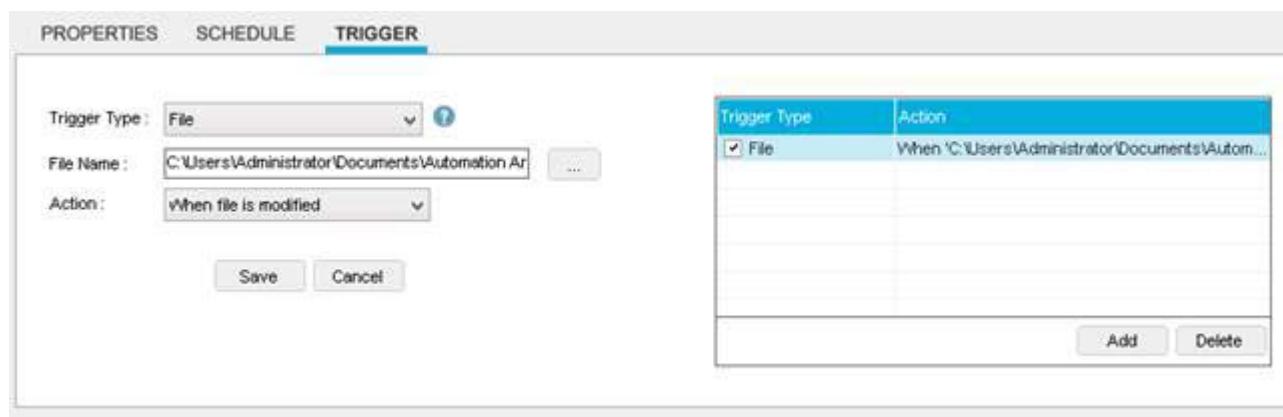
1  Comment: Open the log file and copy the log entry
2  Open File "C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Tasks Error Log.txt"
3  Keystrokes: [CTRL DOWN]a[CTRL UP][CTRL DOWN]c[CTRL UP] in "Tasks Error Log.txt - Notepad"
4  Assign value of Clipboard to variable "$vErrorDesc$"
5  Comment: Extract the SQL error code and task bot name
6  String Operation: Before-After "$vErrorDesc$"; Before: "$vErrorDesc$" and assign output to $vErrorDesc$
7  String Operation: Before-After "$vErrorDesc$"; Before: "SQLSTATE="; After: "Task" and assign output to $vErrorCode$"
8  String Operation: Before-After "$vErrorDesc$"; Before: "Task:" and assign output to $vTaskBot$"
9  String Operation: Before-After "$vErrorDesc$"; Before: "Error:"; After: "Task" and assign output to $vErrorDesc$"
10 String Operation: Before-After "$vTaskBot$"; Before: "My Tasks\"; After: ".atmx" and assign output to $vTaskBotName$"
11 Close Window: "Tasks Error Log.txt - Notepad"
12 Comment: Respond to the SQL 08001 error
13 IF If $vErrorCode$ Equal To (=) "08001" Then
14   Send Email: Subject "Error: $vErrorCode$ in Task Bot: $vTaskBotName$ "
15   Comment: Respond to the SQL 22001 error
16 Else If $vErrorCode$ Equal To (=) "22001" Then
17   Send Email: Subject "Error: $vErrorCode$ in Task Bot: $vTaskBotName$"
18   Comment: Respond to all other errors
19 Else
20   Send Email: Subject "Error occurred in Task Bot: $vTaskBotName$"
21 End If

```

Section 6. Configuring the trigger for the Handle Errors bot

In this section, you configure a trigger for the Handle Errors bot. The trigger runs the bot when the log file is updated. This approach allows other bots to potentially use the same log file with the Handle Errors bot. For example, if the SQL errors occur in other bots, the error code would be the same and the Handle Errors bot can respond as needed.

- ___ 1. Return to the main Enterprise Client window.
- ___ 2. Make sure that the **Handle Errors** bot is selected.
- ___ 3. Configure the trigger.
 - ___ a. Click **trigger** in the Task Relevant Activities pane.
 - ___ b. From the **Trigger Type** list, select **File**.
 - ___ c. Enter the following text for **File Name**:
C:\Users\Administrator\Documents\Automation Anywhere Files\Log\My Tasks\Tasks Error Log.txt
 - ___ d. From the **Action** list, select **When file is modified**.
 - ___ e. Click **Save**. The trigger is shown to the right.



After the trigger is configured, you can run the Account Opening with Hardening bot for each of the six scenarios that are described in the testing bots appendix for this exercise. For scenarios that result in errors, the error log is written, and the Handle Errors bot is triggered.

End of exercise

Exercise review and wrap-up

In this exercise, you were provided with extra requirements for the Account Opening bot. The requirements are about two error scenarios that the bot handles, and two error scenarios that the bot responds to by emailing the appropriate SIB team. You extended the Account Opening bot by adding commands that “hardened” the code so that it can handle the two Excel scenarios. Then, you added error handling to the bot, and configured the Error Handling it to log the error to a file. You then built a reusable bot to handle errors and configured a trigger for that bot to run when the log file is updated. Potentially, you can configure any of the other bots to write a log to the same file. Then, when either of the two SQL errors occur, the handling of the errors would be the same. You also configured the bot to detect all other errors, and to respond by emailing the SIB Support team.

Appendix: Testing bots

This appendix provides the directions to test every scenario. Note that a second appendix includes instructions and all credentials that are needed to log in to Thunderbird for testing.

Section 1: Excel Scenarios

The following scenarios are tested in this section:

- Default scenario: The Excel file is present with the correct .xlsx extension and column format.
- The Excel file has an incorrect column format (the **Customer ID** data is in the wrong column).
- The Excel file has the wrong extension (.xls).

Excel file is present, and has the correct extension (.xlsx) and column format

The Excel file that you use for this scenario does not use any of the code that you worked on this exercise, and the bot should run without any issues. The expected behavior is that the spreadsheet opens, and the bot processes the data without any problems.

- ___ 1. Delete the existing new_customers.xlsx file.
- ___ 2. Prepare the test spreadsheet file for processing.
 - ___ a. Go to: C:\labfiles\hardening_lab.
 - ___ b. Copy the file: GOOD – new_customers.xlsx
 - ___ c. Go to: C:\labfiles\account_opening.
 - ___ d. Paste a copy of the file and name it: new_customers.xlsx
- ___ 3. Run the bot.
 - ___ a. Return to the bot in the Workbench.
 - ___ b. Click **Run**.

The bot runs, opens the spreadsheet, and processing occurs as normal.

Excel file has the wrong extension (.xls)

- ___ 1. Prepare the test file for processing.
 - ___ a. Go to: C:\labfiles\account_opening
 - ___ b. Ensure the new_customers.xlsx spreadsheet in the C:\labfiles\account_opening directory is the copy of the GOOD – new_customers.xlsx file.
 - ___ c. If it is a different version of the spreadsheet, follow the steps for the scenario: **Excel file is present, and has the correct extension (.xlsx) and column format**.
 - ___ d. Rename the spreadsheet file to: new_customers.xls
- ___ 2. Run the bot.
 - ___ a. Return to the bot in the Workbench.

- b. Click Run.

The bot runs and renames the spreadsheet that uses the correct extension: new_customers.xlsx. Processing occurs as normal.

Excel file has incorrect column format (Customer ID is in the wrong column)

- 1. Prepare the test file for processing.
 - a. Go to: C:\labfiles\hardening_lab.
 - b. Copy the file: BAD FORMAT – new_customers.xlsx
 - c. Go to: C:\labfiles\account_opening.
 - d. Paste a copy of the file and name it: new_customers.xlsx
- 2. Run the bot.
 - a. Return to the bot in the Workbench.
 - b. Click Run.

The bot runs, opens the spreadsheet, and moves the **Customer ID** column to column A1 inserting it in front of the **First Name** column. The spreadsheet remains open and processing occurs as normal.

Section 2: Handling error scenarios

The Handle Errors bot looks specifically for two SQL errors and notifies the appropriate department. If it does not see either of the two SQL errors it is looking for, it sends the error to the Support team. The following scenarios are tested in this section:

- SQL 08001 error (incorrect DB2 password)
- SQL 22001 error (bad data)
- All other errors

SQL 08001 error

To simulate the SQL 08001 error, the DB2 credential is changed with one that contains an incorrect password. The expected behavior is that the bot attempts to process the file, but encounters an error when it attempts to access the database. As a result, the Excel file is left open, the bot writes to the log file, and stops running. The Handle Errors bot, which monitors whether the log file is updated, is triggered. It determines whether the error is the SQL 08001 error, and sends an email to the SIB User Security team.

- 1. If needed, update the spreadsheet file in C:\labfiles\account_opening so that it contains the default scenario spreadsheet.
 - a. Go to: C:\labfiles\hardening_lab.
 - b. Copy the file: GOOD – new_customers.xlsx
 - c. Paste a copy of the file in the C:\labfiles\account_opening folder and name it: new_customers.xlsx

- __ 2. In the Control Room, edit the **DB_CONNECT_STRING** so that it contains an incorrect value for the database password.
 - __ a. Go to the web browser window with the Control Room by using the link <http://vclassbase:81> and login as **devUser1** in the **Username** and **Password** fields.

Information

You need to login as **devUser1** because **devUser1** is the owner of the credential.

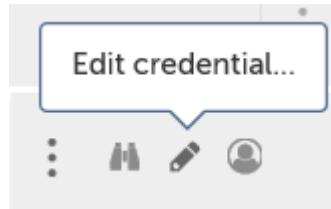
- __ b. Go to Bots > Credentials.

Credentials

 Create credential...

	MY CREDENTIALS	MY LOCKERS	CREDENTIAL REQUESTS
Search name <input type="text"/> 			
Credentials {2 of 2}   			
	TYPE	NAME ↑	LOCKER NAME
	Standard	ACCESS_CONNECT_STRING	IBM Business Automation Workflow Systems
	Standard	DB_CONNECT_STRING	IBM Business Automation Workflow Systems

- __ c. Mouse over the ellipsis for the **DB_CONNECT_STRING** credential.
- __ d. Click the **Edit credential** icon.



- __ e. Scroll down to the **Attributes** section.
- __ f. For the **DB_CONNECT_STRING** attribute, delete the value, and replace it with:

```
Provider=IBMOLEDB.DB2COPY1;Password=WRONG;Persist Security Info=True;UserID=student;Data Source=SIB;Location="" ;Extended Properties=""
```

___ g.

Attributes

ATTRIBUTE NAME	DESCRIPTION	TYPE	VALUE
DB_CONNECT_STRING	DB2 connection string	Standard	Password=WRONG;P@ssw0rd

___ h. Scroll up and click **Save changes**.

Bots > Credentials > Edit credential

Edit credential

[Cancel](#)

[Save changes](#)

Only the owner of a credential can edit the attribute name and descriptions;
A consumer of a credential can only edit the user-specific values of a credential.

- ___ 3. Run the bot.
 - ___ a. Go back to the Workbench and return to the **Account Opening with Hardening** bot.
 - ___ b. Click **Run**.

The bot runs, attempts to log in to the database, encounters an error, writes to the log file, and then stops running. After the log file is updated, the Handle Errors bot starts. The Handle Errors bot then copies the error from the log file, extracts the error code and task name, and sends an email to the SIB User Security team.
- ___ 4. Verify that the bot sent the email to userSecurity@sib.com.
 - ___ a. Log in to Thunderbird as the SIB User Security user with the password userSecurity.
 - ___ b. Locate email with the subject line: **Error: 08001 in Task Bot: Account Opening with Hardening**.
 - ___ c. Verify the email contents:

Dear User Security team,

The following SQL error: 08001 indicates the bot's login credentials are invalid. The complete error description is:

SQL30082N Security processing failed with reason "24" ("USER NAME AND/OR PASSWORD INVALID"). SQLSTATE=08001

This error occurred in the following Bot:

C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx

Thank you,
Handle Errors Task Bot

___ 5. Return the database credential to its original state for more testing.

Because you changed the database credential, it needs to be set back to the correct value before you perform more testing.

- ___ a. Go to the Control Room window.
- ___ b. Click **Credential Manager** on the left navigation pane.
- ___ c. Select the **DB_CONNECT_STRING** credential.
- ___ d. Click **Edit**.
- ___ e. Delete the current **ATTRIBUTE VALUE** and replace with:

```
Provider=IBMOLEDB.DB2COPY1;Password=Education1;Persist Security
Info=True;User ID=student;Data Source=SIB;Location="" ;Extended
Properties=" "
```

- ___ f. Click **Save**.
- ___ g. Close Thunderbird.
- ___ h. Close the Excel Spreadsheet.

___ 6. *Optional.* Verify that the bot functions correctly.

Because the default scenario file is used here, you can rerun the bot after updating the credential to verify that it functions correctly. The bot should run normally with no errors.

SQL 22001 error

To simulate the SQL 22001 error, you use the `BAD DATA - new_customers.xlsx` spreadsheet, which contains “bad” data. The bot attempts to copy the bad data for the second Customer ID entry into the database column, which does not support the data. This scenario produces the SQL 22001 error. Thus, the bot attempts to process the data transfer but encounters an error while it is running. After the bot encounters the error, it leaves both the spreadsheet and SIB CRM application open, and stops running.

___ 1. Prepare the test spreadsheet file for processing.

- ___ a. Go to: `C:\labfiles\hardening_lab`.
- ___ b. Copy the file: `BAD DATA - new_customers.xlsx`
- ___ c. Go to: `C:\labfiles\account_opening`.
- ___ d. Paste a copy of the file and name it: `new_customers.xlsx`

___ 2. Run the bot.

- ___ a. In the Workbench, return to the **Account Opening with Hardening** bot in the Workbench.
- ___ b. Click **Run**.

The bot runs, opens the spreadsheet and SIB CRM application and begins transferring data. When it attempts to write the bad data to the database, the error occurs, and the bot writes the error to the log file. After the log file is updated, the Handle Errors bot

starts. It then copies the error from the log file, parses it into the error code and task name, and sends an email to the Data Integrity support desk.

- ___ 3. Verify the email.
 - ___ a. Log in to Thunderbird as the SIB Data Integrity user with the password `dataIntegrity`.
 - ___ b. Locate email with the subject line: **Error: 22001 in Task Bot: Account Opening with Hardening.**
 - ___ c. Verify the email.

Data Integrity Desk,

The following SQL error: 22001 occurred indicating an insert value is too long for the column. The complete error description is:
`[DB2/NT64] SQL0433N Value " ")_(*&(*^&$^#%@"is too long. SQLSTATE=22001`

This error occurred in the following Bot:

`C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx`

Thank you,
 Handle Errors Task Bot

- ___ 4. Prepare the environment for more testing.
 - ___ a. Close Thunderbird.
 - ___ b. Close the Excel spreadsheet.
 - ___ c. Close the SIB CRM application.

All other errors

Because it is impossible to know all possible errors that might occur, it is a good practice to enable error handling to address any errors that the bot might encounter. In this exercise, you can test two general error scenarios. In the first scenario, the SIB CRM application is not in the folder. In the second scenario, the spreadsheet file is not in the folder. The expected behavior for both scenarios is that the bot sends an email to the SIB Support team with a description of the error.

General error scenario 1: The SIB CRM application is not in the folder

- ___ 1. Prepare the environment to simulate the error.
 - ___ a. Go to: `C:\labfiles\account_opening`.
 - ___ b. Rename `SIB_CRM_app.exe` to: `SIB_CRM_app_Copy.exe`
- ___ 2. Run the bot.
 - ___ a. Return to the **Account Opening with Hardening** bot in the Workbench.
 - ___ b. Click **Run**.

The bot runs, opens the spreadsheet, and attempts to open the SIB CRM application then stops. A moment after the bot completes, the Handle Errors bot is triggered and runs.

- ___ 3. Verify the email.
 - ___ a. Log in to Thunderbird as the **SIB Support** user with the password support.
 - ___ b. Locate the email message with the subject line:
Error occurred in Task Bot: Account Opening with Hardening.
 - ___ c. Verify the email.

Dear Support Desk,

The following error occurred while executing the Account Opening with Hardening Task Bot. The error description is: The Program/File 'C:\labfiles\account_opening\SIB_CRM_app.exe' that you are trying to open, could not be found.

This error occurred in the following Bot:

C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx

Thank you,
Handle Errors Task Bot

- ___ 4. Prepare the environment for more testing.

Because you changed the name of the `SIB_CRM_app.exe` application, you must restore the original file name before you do more tests.

- ___ a. Go to: C:\labfiles\account_opening.
- ___ b. Rename the file `SIB_CRM_app_Copy.exe` to: `SIB_CRM_app.exe`
- ___ 5. Close Thunderbird.

General error scenario 2: The Excel file is not in the folder

- ___ 1. Prepare the Excel file for processing.
 - ___ a. Go to: C:\labfiles\account_opening.
 - ___ b. Delete the file: `new_customers.xlsx`
- ___ 2. Run the bot.
 - ___ a. Return to the **Account Opening with Hardening** bot in the Workbench.
 - ___ b. Click **Run**.

The bot runs, opens the spreadsheet, attempts to open the SIB CRM application, and then stops. A moment after the bot stops, the Handle Errors bot is triggered and runs.

- ___ 3. Verify the email.
 - ___ a. Log in to Thunderbird as the **SIB Support** user.

- ___ b. Locate the email with the subject line:

Error occurred in Task Bot: Account Opening with Hardening.

- ___ c. Verify the email.

Dear Support Desk,

The following error occurred while executing the Account Opening with Hardening Task Bot. The error description is:

'C:\labfiles\account_opening\new_customers.xlsx' could not be found.
Check the spelling of the file name, and verify that the file location is correct. If you are trying to open the file from your list of most recently used files, make sure that the file has not been renamed, moved, or deleted.

This error occurred in the following Bot:

C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks\Account Opening with Hardening.atmx

Thank you,
Handle Errors Task Bot

- ___ 4. Close Thunderbird.

Appendix: Logging in to Thunderbird

You need to log in to Thunderbird to verify that the correct emails are sent. The following instructions explain how to log in to Thunderbird by using the SIB Support credentials. A table of user IDs and passwords for each relevant account is also provided.

- 1. On the desktop, double-click the **Mozilla Thunderbird** shortcut.



- 2. Select the **SIB Support** user name.



- 3. Click **Start Thunderbird** and enter support for the password.

You can refer to the following table when you log in to Thunderbird as various users.

Email account	Password	Email address	Purpose
SIB Support	support	support@sib.com	General support
SIB User Security	userSecurity	userSecurity@sib.com	Handles user security issues
SIB Data Integrity	dataIntegrity	dataIntegrity@sib.com	Handles data integrity issues

Exercise 11. Managing bots

Estimated time

01:00

Overview

This exercise demonstrates how to create dependencies between bots so that one bot can run other bots and pass parameters. You are also introduced to bot management features in the Enterprise Client.

Objectives

After completing this exercise, you should be able to:

- Create dependencies between bots
- Pass parameters from one bot to another
- Upload bots to the Server Repository and compare bot code
- Use bot management tools in the Enterprise Client

Introduction

In this exercise, you build a simple bot that passes parameters to another bot. You create a dependency between the two bots, deploy the bot to the server repository, and compare the code. You also work with bot management features in the Enterprise Client.

The exercise includes these sections:

- [Section 1, "Preparing to code bots"](#)
- [Section 2, "Building code for client bot"](#)
- [Section 3, "Uploading Server bot for code comparison"](#)
- [Section 4, "Reviewing notifications and system logs"](#)
- [Section 5, "Creating a report in Report Designer"](#)

Section 1. Preparing to code bots

In this section, you prepare two bots for coding. First, you create a copy of the original Data Consistency bot, which becomes the Server bot. Then, you create a new bot to act as the Client bot.

1.1. Making a copy of original Data Consistency bot

- ___ 1. In Enterprise Client, right-click the **Data Consistency** bot and select **Copy**.
- ___ 2. Name the new bot **Data Consistency Server**.
- ___ 3. Click **Save**.

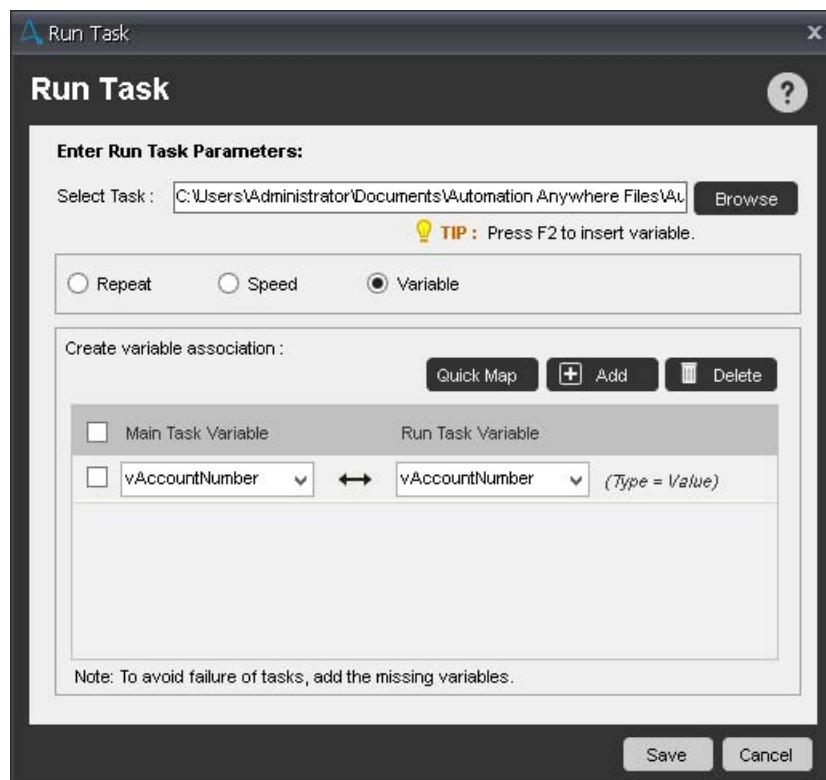
1.2. Creating the client bot that passes a variable to the server bot

- ___ 1. Click **New** on the toolbar to create a bot.
- ___ 2. Click **Task Editor**.
- ___ 3. Add a variable to house the Account Number that the Client bot passes to the Server bot.
 - ___ a. In the Variable Manager, click **Add**.
 - ___ b. In the **Name** field, enter `vAccountNumber`.
 - ___ c. Enter **1234** for Value.
 - ___ d. Click **Save**.
- ___ 4. Save the bot.
 - ___ a. Click **Save** on the toolbar.
 - ___ b. Provide a name to be used for the bot: **Data Consistency Client**
 - ___ c. Click **Save**.

Section 2. Building code for client bot

In this section, you add the code to run the other bot and pass the Account Number as a parameter.

- ___ 1. In the **Commands** list of the Task Editor, expand the **Task** command and double-click **Run Task**.
- ___ 2. For the **Select Task** field, click **Browse**, and choose Data Consistency Server.atmx in the C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere\My Tasks folder.
- ___ 3. Select the **Variable** option.
- ___ 4. In the **Create variable association** section, select **vAccountNumber** for both the **Main Task Variable** and the **Run Task Variable**.
- ___ 5. Click **Save**.



2.1. Creating bot dependency

Now that you have a Client bot and a Server bot, you must create a dependency between them. The Control Room can then track that dependency.

- 1. In the Task Editor, open the **BOT DEPENDENCIES** tab.



- 2. Click **Add**.
- 3. In the Open window, expand the **My Tasks** folder, and double-click **Data Consistency Server.atmx**.

The bot dependency is listed on the **BOT DEPENDENCIES** tab.



- 4. Save the bot by clicking **Save** on the toolbar.



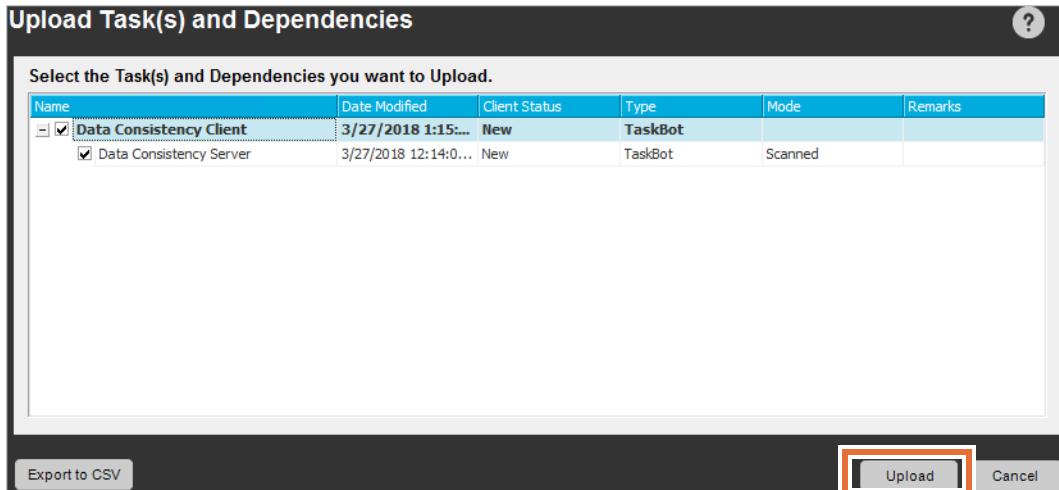
Information

You complete the code changes for the **Data Consistency Server** bot after you upload the bot to the Server Repository so that you can view a code comparison between the Server and Client versions.

Section 3. Uploading Server bot for code comparison

3.1. Uploading the bot to Server Repository

- ___ 1. Return to the Enterprise Client and locate the Client bot.
- ___ 2. Right-click the **Data Consistency Client** bot and click **Upload**.
- ___ 3. Click **Upload**.



- ___ 4. When the “Upload is successful” message opens, click **OK** to close it.
- ___ 5. In the Enterprise Client, right-click the **Data Consistency Server** bot and click **Edit** to open it in the Task Editor.
- ___ 6. In the Task Editor, highlight the first two lines of code, right-click them and click **Delete**.

```

1  ■ Comment: Prompt the user to enter an account number and assign it to the vAccountNumber variable
2  ⚠ Prompt Message: "Enter the customer account number:" in "Taskbar" Assign value to Variable: $vAccountNumber$
3  ■ Comment: Source: SIB Intranet
4  ■ Comment: Use the Web Recorder to check the SIB customer list for the account number that is entered by the user
5  ⌚ Open "http://www.sib.com/cid.html"
6  ⌚ Extract table from "http://www.sib.com/cid.html" website's '1' table

```

- ___ 7. When prompted to confirm deletion, click **Yes**.
- ___ 8. Click **Save** on the toolbar to save the bot.

3.2. Comparing code between the Client and Server bots

- ___ 1. Return to the Enterprise Client.
- ___ 2. In the lower left of the Features pane, click **MANAGE**.



The screenshot shows the Bluebeam software interface. At the top, there are several icons: New (yellow sun), Record (red circle), Run (green play button), Edit (yellow wrench), and ROI (chart with dollar sign). Below the toolbar, there are two main navigation sections: 'AUTOMATE' and 'MANAGE'. Under 'MANAGE', there is a list of items: Repository, Schedules, Triggers, Reports, and My Reports. The 'My Reports' item is highlighted with a blue selection bar. To the right of the 'MANAGE' section, a table titled 'My Reports' is displayed. The table has two columns: 'File Name' and 'Total clicks'. There are four rows in the table, but all fields are empty.

File Name	Total clicks

3. In the **Manage** list, click **Repository**.

The Repository screen opens. Your Server and Client bots are listed in the Server Repository. The Server bot was uploaded automatically because of the dependency with the Client bot. Other bots that you developed are listed in the Client Repository.

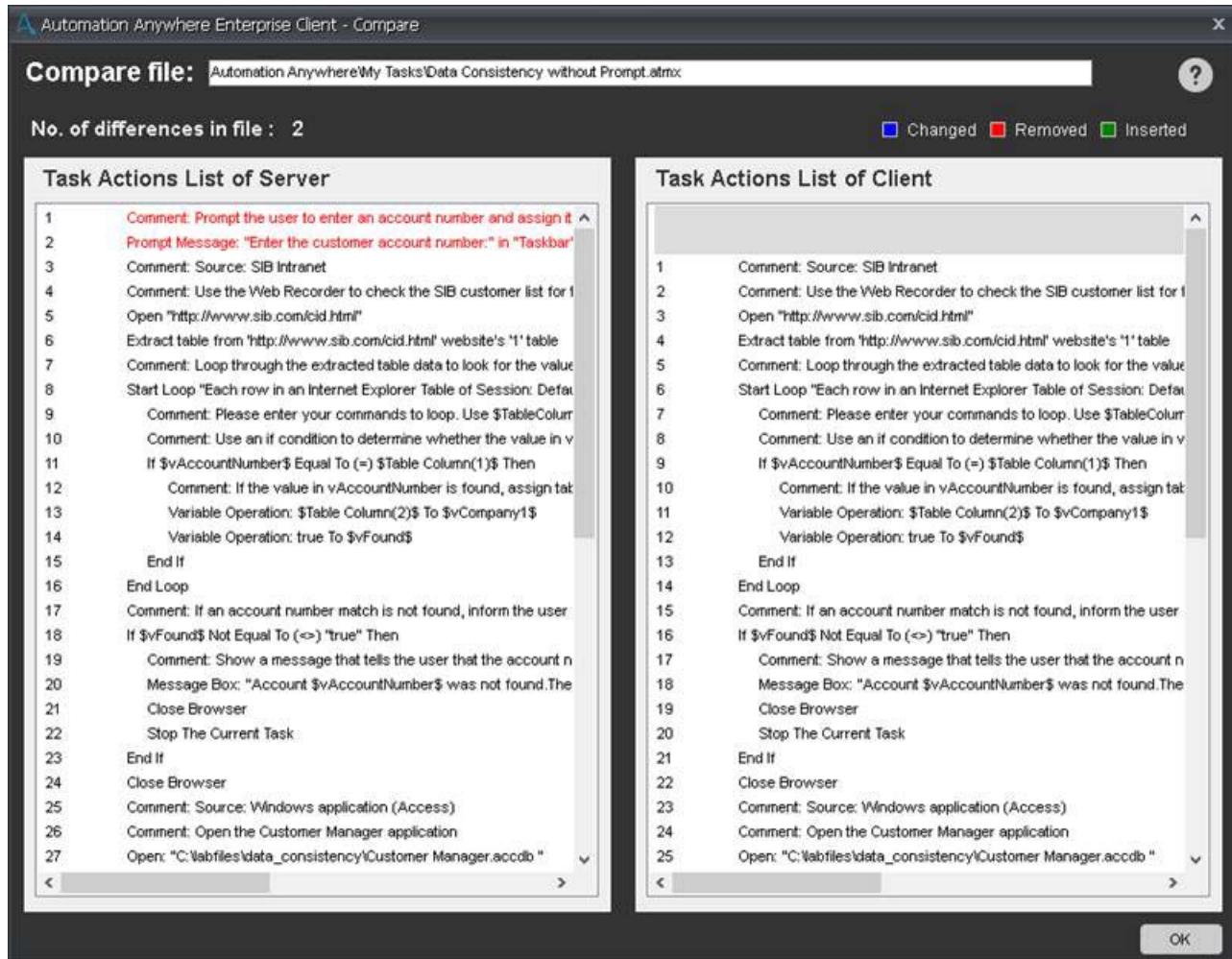
Name	Type
Data Consistency Server.atmx	Task File
Data Consistency Client.atmx	Task File

Name	Type
Account Opening with Hardening.at...	Task File
Account Opening.atmx	Task File
Check Declines.atmx	Task File
Data Consistency Client.atmx	Task File
Data Consistency Server.atmx	Task File
Data Consistency with Login.atmx	Task File
Data Consistency.atmx	Task File
Excel File Prep.atmx	Task File
Handle Errors Original.atmx	Task File

- ___ 4. Highlight the **Data Consistency Server** bot in both repositories.
- ___ 5. Click **Compare**.



The Code Comparison window displays the differences between the Server and Client versions of the bot.



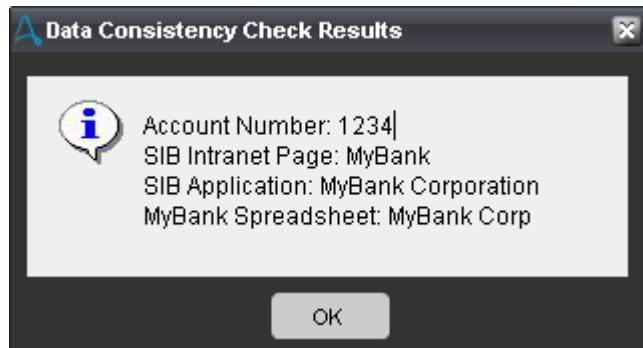
- ___ 6. Click **OK**.
- ___ 7. Click the X in the upper-right corner to close the Repository screen.

3.3. Running the new bot

- ___ 1. In the Enterprise Client, from the **AUTOMATE** tab, right-click the **Data Consistency Client** bot and click **Run**.

Two bots run in one session. Because the first bot runs quickly, you might not see it in the Run Time Window. Depending on the timing, you might see the dependent bots in the Run Time Window.

After the bot completes, a message displays the results.



- __ 2. Click **OK**.

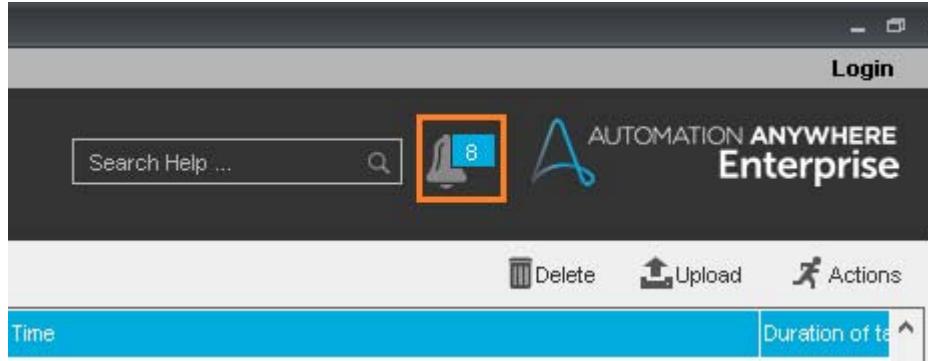
Section 4. Reviewing notifications and system logs

4.1. Viewing notifications

When you have a scheduled bot or a trigger, Automation Anywhere notifies you when your task encounters an error or fails to run.

When you open Notifications, you see errors generated from the prior bots that were triggered then had an error during execution. The last run is not listed.

- ___ 1. Click the **Notifications** icon to see the list of notifications.



Troubleshooting

If you did not have errors during your scheduled or triggered bot runs, you might not have any notifications.

- ___ 2. Close the Notifications window by clicking the X in the upper-right corner.

4.2. Viewing system logs

- ___ 1. On the **Tools** menu, click **System Logs**.

The System Logs window opens. You can select a start date and an end date.

- ___ 2. Click **Generate Logs**.

By default, all logs for the current day are listed.

The screenshot shows the 'System Logs' window with the following details:

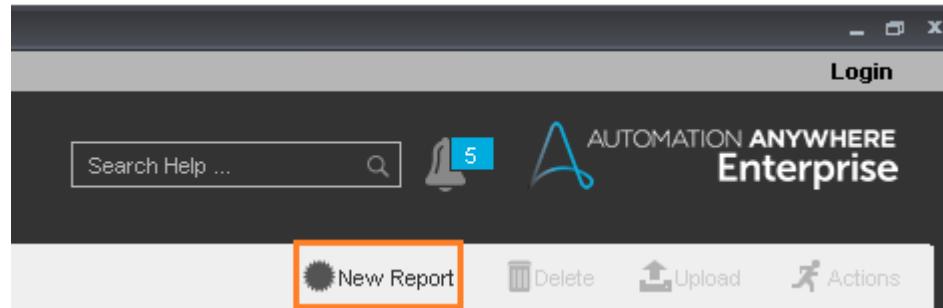
- Header:** 'System Logs' with a help icon and a close button.
- Filter Options:**
 - 'Select Log Type:' dropdown set to 'All'.
 - 'Start Date:' and 'End Date:' both set to '03/07/2018'.
 - 'Date Format - "MM/DD/YYYY"' label.
 - 'Generate Logs' button.
- Logs Table:**

Sr. No.	Task/Variable Name	Date	Time	Description
1	Data Consistency solution.atmx	03/07/2018	10:47:16	Task Run, Completed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
2	Data Consistency solution.atmx	03/07/2018	10:47:19	Task Run, Started, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
3	Data Consistency solution.atmx	03/07/2018	10:49:44	Task Run, Completed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
4	Data Consistency solution.atmx	03/07/2018	10:58:23	Task Run, Started, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
5	Data Consistency solution.atmx	03/07/2018	11:00:33	Task Run, Completed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
6	Run Data Consistency Bot.atmx	03/07/2018	11:00:57	Task Run, Started, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
7	Data Consistency solution.atmx	03/07/2018	11:00:57	Task Run, Started, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
8	Data Consistency solution.atmx	03/07/2018	11:01:38	Task Run, Failed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
9	Run Data Consistency Bot.atmx	03/07/2018	11:01:40	Task Run, Completed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
10	Data Consistency solution.atmx	03/07/2018	11:02:16	Task Modified, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
11	Data Consistency solution.atmx	03/07/2018	11:02:22	Task Run, Started, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
12	Data Consistency solution.atmx	03/07/2018	11:03:37	Task Run, Failed, C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018\Logs\03-07-2018\03-07-2018.log
- Export Options:**
 - 'Export to CSV file:' field containing 'C:\Users\Administrator\Documents\Automation Anywhere Files\03-07-2018.csv'.
 - 'e.g.C:\Reports\first.csv' placeholder.
 - 'Export' button.
 - 'OK' button at the bottom right.

3. Click **OK** to close the System Logs window.

Section 5. Creating a report in Report Designer

- __ 1. Click **MANAGE** in the Features panel.
- __ 2. Click **New Report**.



The Report Designer opens with default values.

Report Designer [New Report]*

Step 1: Select Report Type

- Task Run Task Timeline
- Workflow Run Workflow Timeline
- ROI Visual Logs

Step 2: Select Task

- All Tasks
- Select Specific Folder ...
- Select Task(s) ...

Step 3: Select Date Range

- Yesterday
- 05/03/2018 - 05/03/2018

Save and Run Report Run Report

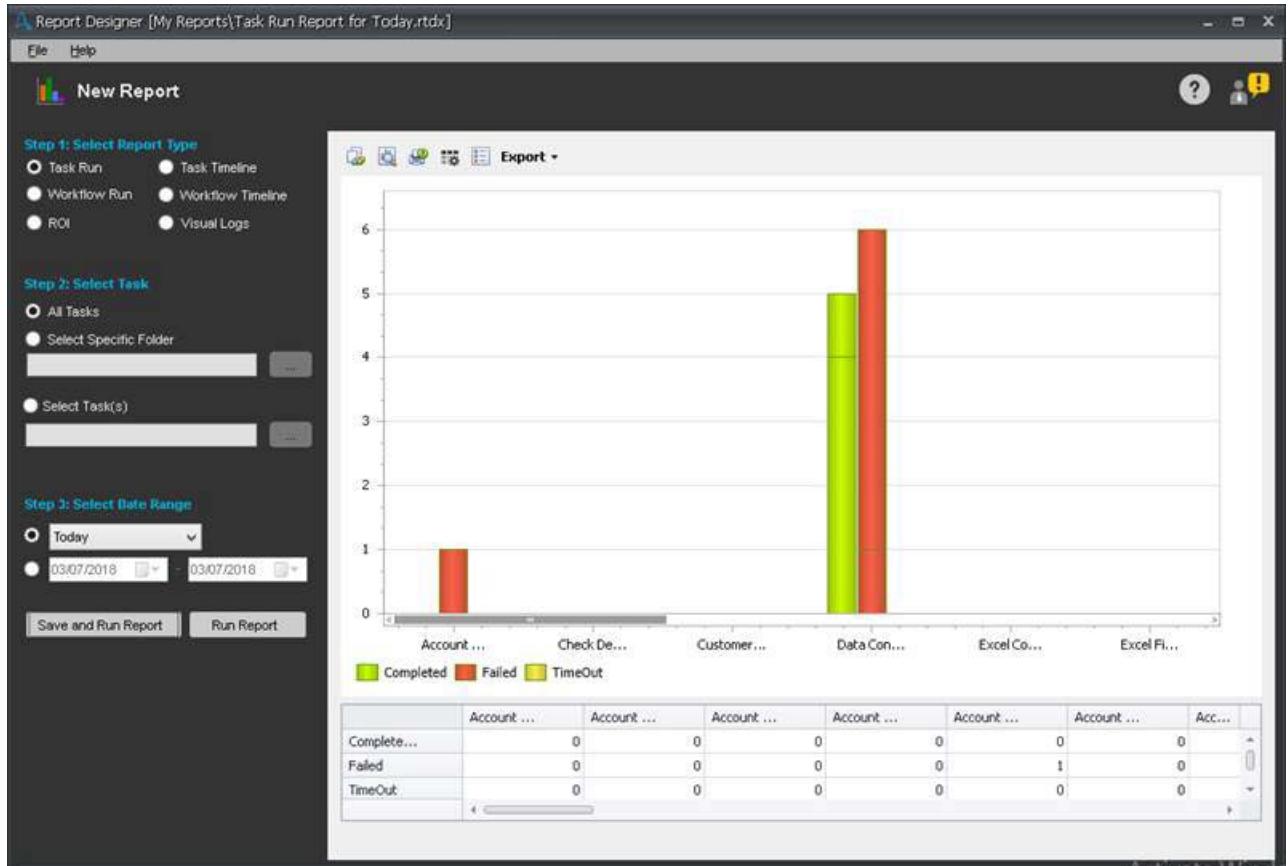
Report Designer provides a graphical interface for managing tasks and workflows as reports.

Select the type of report and a time range to generate a report.

- ___ 3. In the **Step 3: Select the Date Range** section, select **Last 7 Days**.
- ___ 4. Click **Save and Run Report**.
- ___ 5. Provide a **Filename** for the Report: **Task Run Report**.
- ___ 6. Click **Save**.

The report is generated and opens. Note that you might need to scroll sideways to see the whole report.

- __ 7. Click the X in the upper-right corner to close the Report Designer.
The new report is listed in the My Reports section of the Enterprise Client.
- __ 8. Select your report, and on the toolbar, click **Run**.
The report reopens.



End of exercise

Exercise review and wrap-up

In this exercise, you created a bot that calls another bot and passes a parameter. You created a dependency between the bots, uploaded them to the Server Repository, and compared the code. You also viewed system logs and created a report that displays bot executions.

Exercise 12. Administering bots in the Control Room

Estimated time

00:30

Overview

This exercise provides an overview of central bot administration tasks that you can configure and access through the Control Room. You learn about how an administrator can use various Control Room features to manage, schedule, and run bots on remote user (BotRunner) machines.

Objectives

After completing this exercise, you should be able to:

- Upload bot files to the Control Room. Run bots from the Control Room on a bot runner machine
- View the status of bots in various phases (completed, running, scheduled)

Introduction

IBM Robotic Process Automation with Automation Anywhere is based on a client/server model. The Control Room serves as a central repository and management tool for bots. Developer users can upload bots to the Control Room, where they can be downloaded by other users, including Bot Runner (bot user) clients or other developers. You can also use the Control Room to run or schedule bots remotely on Bot Runner client machines.

In a typical RPA deployment, Bot Creators (developers) develop and upload bots to the Control Room. An RPA administrator, or a user who has the appropriate rights, can deploy the bots to remote Bot Runner machines (unattended bot). If a Bot Runner user has download permissions, they can download a copy of the bot from the Control Room Repository to their local machine, and run it locally (attended bot).

As the central server for IBM Robotic Process Automation with Automation Anywhere, the Control Room also provides the following administrative features: configuring server settings; viewing bot performance through dashboards; managing bots, schedules, and triggers; user and role management; and credentials and security. The Audit Trail maintains an audit log of server-based events, such as bot deployments, user logins, and so on.

Requirements

This lab requires the computer lab environment that was developed for this course.

Section 1. Uploading bots to the Control Room

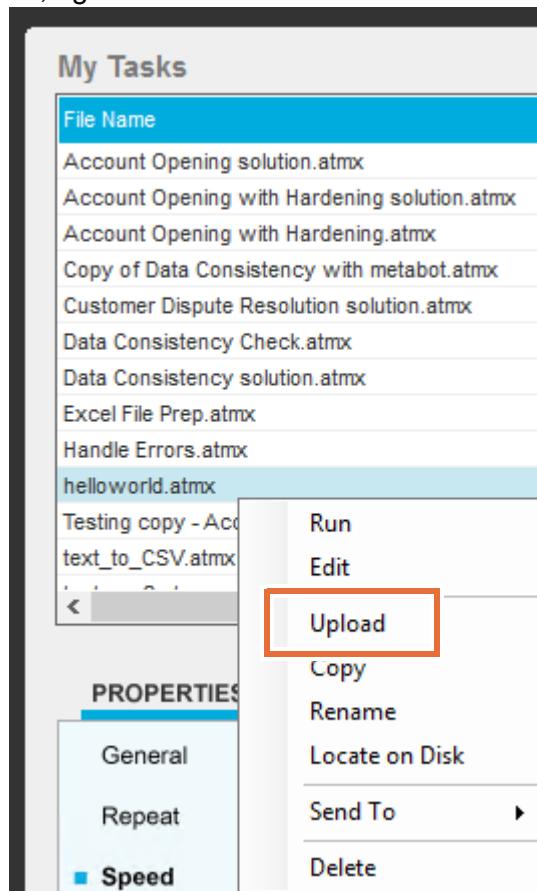
In this section of the exercise, you learn how to upload bots to the Control Room. You also prepare the environment for the next section by moving the My Tasks folder to a temporary location. The bot does not need to be available to the local Bot Runner Enterprise Client installation to run, as it can be run from the Control Room.

1.1. Uploading bots to the Control Room repository

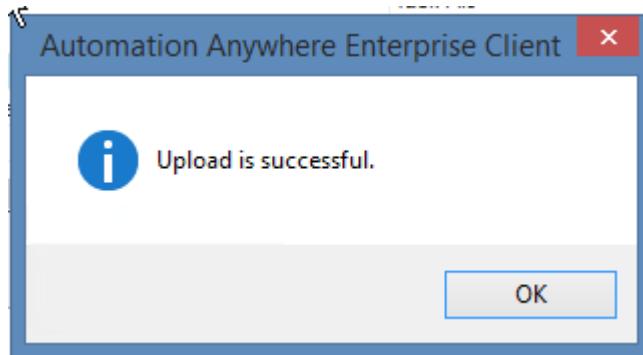
In this part of the exercise, you work as the development user devUser1 to upload the helloworld.atmx bot to the Control Room. This action makes the helloworld.atmx bot available in the Control Room Repository, which can then be deployed to Runtime client computers.

You also upload the Data Consistency with login.atmx bot so that you can upload a bot with a dependency to the Control Room.

- ___ 1. Log in to the Enterprise Client as devUser1.
- ___ 2. Upload helloworld.atmx to the Control Room.
 - ___ a. In the **My Tasks** list, right-click **helloworld.atmx** and click **Upload**.



- ___ b. When you see the **Upload is successful** message, click **OK**.



- ___ 3. Upload the Data Consistency with login.atmx file.

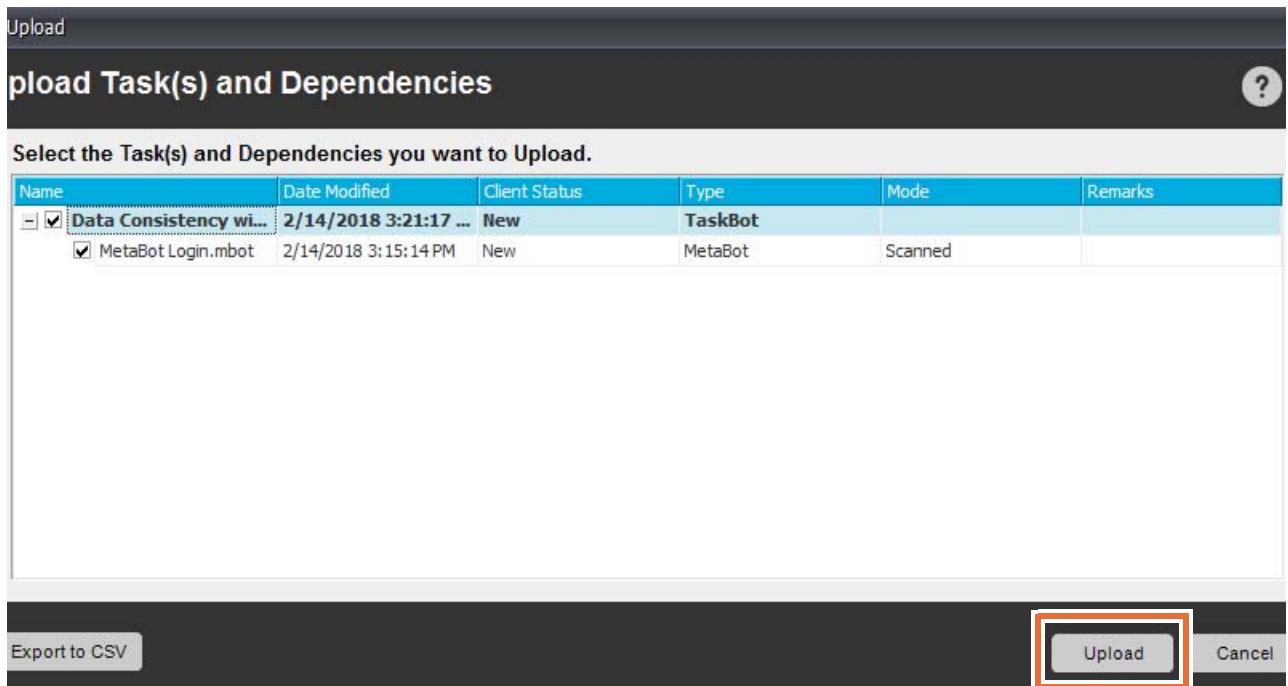
- ___ a. In the My Tasks list, right-click **Data Consistency with login.atmx**, and click **Upload**.

The “Upload Task(s) and Dependencies” window opens. This window lists the various bot dependencies so that you can upload the main bot and any dependent bots or MetaBots in one operation.

The Data Consistency with login bot uses the MetaBot Login.mbot file, so this file is listed in this window. Make sure that MetaBot Login.mbot is selected.

A screenshot of the "Upload Task(s) and Dependencies" window. The title bar says "Upload" and the main title is "Upload Task(s) and Dependencies". Below that is a sub-instruction: "Select the Task(s) and Dependencies you want to Upload." A table lists two items: "Data Consistency wi..." (TaskBot, New) and "MetaBot Login.mbot" (MetaBot, Scanned). Both items have checkboxes next to them; the first one is checked and highlighted with a dashed border.

- ___ b. In the “Upload Task(s) and Dependencies window, click **Upload** to upload the MetaBot Login.mbot file and the Data Consistency with login.atmx file.



- ___ c. When you see the **Upload is successful** window, click **OK** to close the window.
- ___ 4. Upload the Transactions.atmx file.
- ___ a. In the My Tasks list, right-click **Transactions.atmx**, and click **Upload**.
- ___ b. When you see the **Upload is successful** message, click **OK**.

1.2. Preparing the environment and logging in to the Enterprise Client as a runtime user

When you run a bot from the Control Room, you can run it on a client machine that has IBM Robotic Process Automation with Automation Anywhere installed, with a user who is assigned a Runtime license.

In this part of the exercise, you prepare the lab environment and move the **My Tasks** folder to a temporary location. You see how the Control Room deploys and runs a bot on a Bot Runner machine even when the client machine does not have a copy of the bot in the **My Tasks** folder.

You then log in to the Enterprise client as the `botUser1` user, who was assigned a Bot Runner (Runtime) license.

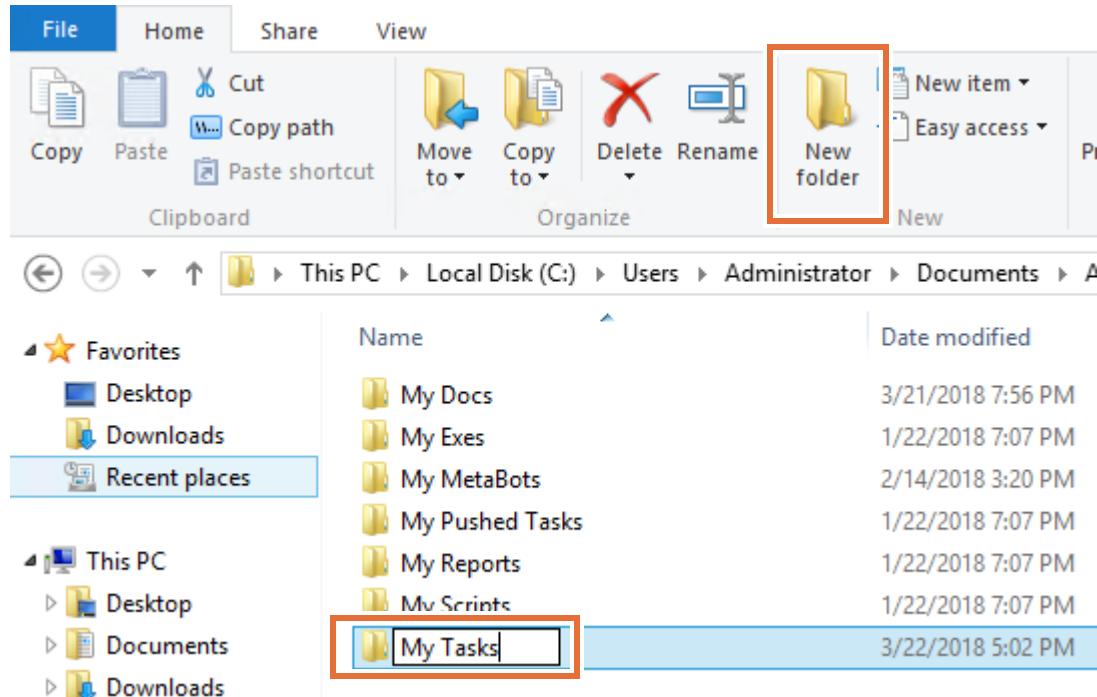
- ___ 1. Move the **My Tasks** folder to a temporary location.
- ___ a. Go to `C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere`.
- ___ b. Move the **My Tasks** folder to a temporary location, such as the Desktop.
- If you want, you can move this folder back to its original location after you complete this exercise.
- ___ 2. Create a folder that is called **My Tasks** to replace the folder that you just moved.

You create this empty **My Tasks** folder so that the Enterprise Client can recognize it. When you run the `helloworld.atmx` bot from the Control Room, you see that the Enterprise Client does not need to have a local copy of the bot installed to run it. Also, later in this exercise, you download the `helloworld.atmx` file from the Control Room Repository to the local **My Tasks** folder.

- __ a. Make sure that you are in the following directory:

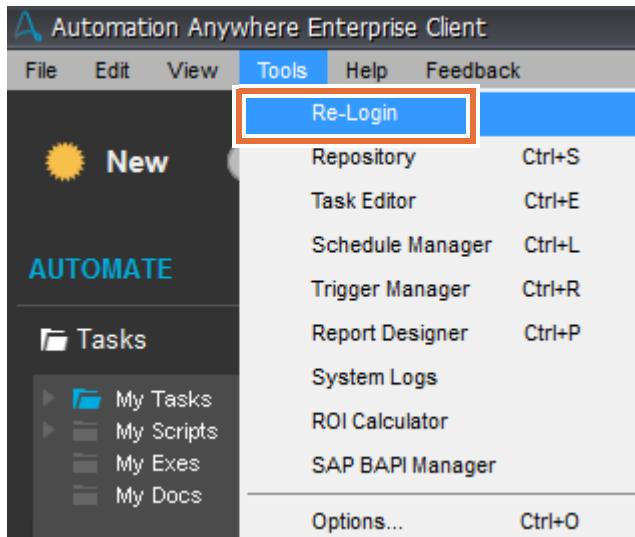
C:\Users\Administrator\Documents\Automation Anywhere Files\Automation Anywhere

- __ b. In the Explorer Window Home tab toolbar, click **New folder**, and name the folder: **My Tasks**



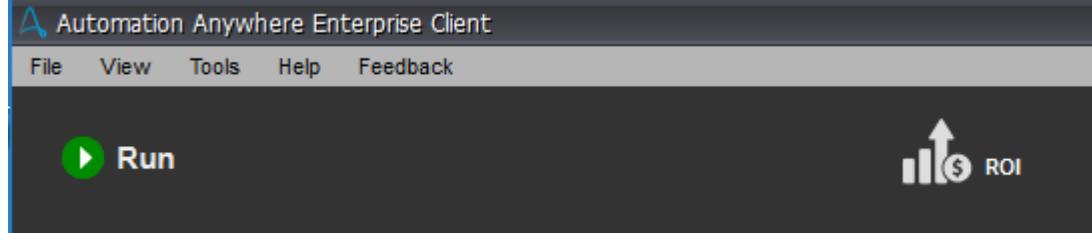
- ___ 3. Log in to the Enterprise Client as `botUser1`, who is assigned a Runtime (Bot Runner) license.

- ___ a. In the Enterprise Client menu, click **Tools > Re-Login**.



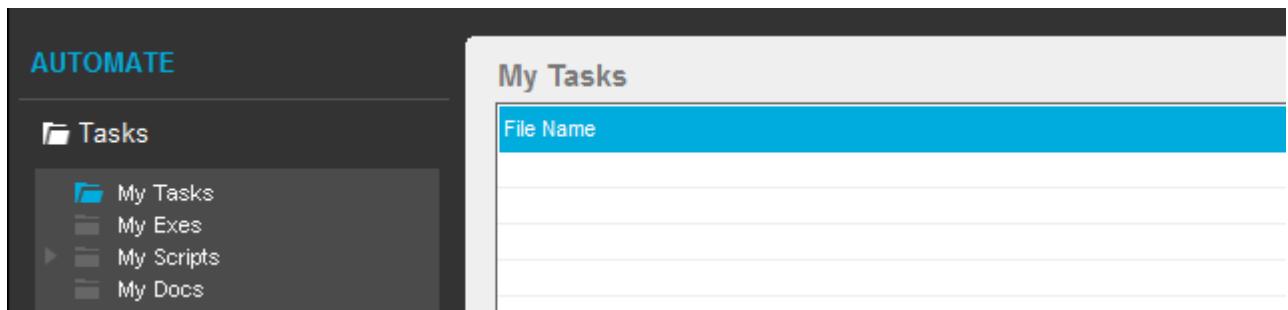
- ___ b. In the “Login to Control Room” window, enter `botUser1` in the **Username** and **Password** fields, and click **Login**.
- ___ c. When you see the **Login is successful message**, click **OK** to close the window.

Notice that the user interface for `botUser1` is different than the user interface that `devUser1` sees. The toolbar has only one active option, which is **Run**.



As a Runtime license user, `botUser1` does not have development privileges, and thus they cannot create or edit bots.

- ___ 4. In the Enterprise Client, confirm that the **My Tasks** folder is empty.



- ___ 5. Minimize the Enterprise Client window.

**Important**

You must stay logged in to the Enterprise Client as `botUser1` so that the `botUser1` user account is active. If the `botUser1` account is not active, it cannot be selected in the Control Room.

Section 2. Running bots from the Control Room

In this section of the exercise, you learn how to run bots from the Control Room. The bot does not need to be available to the local Bot Runner Enterprise Client installation to run, as it can be run from the Control Room. You run a bot four different ways:

- Run bot now - complete
- Run bot now - do not complete (leave running)
- Schedule bot to run later
- Run both with queue

In the next section you view the status of these bots.

2.1. Run bot now from the Control Room

In this section, you log in to the Control Room as `rpaAdmin`, and then run the `helloworld.atmx` bot on the `botUser1` account.

Note: You can deploy and run a task to a client machine without having the task file downloaded locally to the machine.

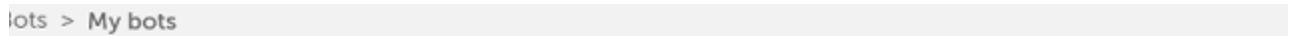
- 1. If you are not already logged in to the Control Room, sign in as `rpaAdmin`.
- 2. Click **Bots > My Bots**.

You see the `helloworld.atmx`, `Data Consistency with Login.atmx` and `Transactions.atmx` files that you uploaded in the previous part of the exercise.

	TYPE	NAME	SIZE	LAST MODIFIED
<input type="checkbox"/>	Task Bot	Data Consistency with Login...	2.69 MB	14:20
<input type="checkbox"/>	Task Bot	helloworld.atmx	59.5 KB	15:20
<input type="checkbox"/>	Task Bot	Transactions.atmx	16.04 KB	15:20

___ 3. Run the helloworld.atmx bot on the BotRunner account.

___ a. Click **Run bot now...**



My bots Import bots... Export bots... **Run bot now...** Run bot with queue... Schedule

You can see files here, upload them from your Bot creator or your Bot runner. You will only see files that you have permission to see.

___ b. In the **Run Bot Now** page, select the helloworld.atmx Task Bot and click the yellow right arrow. The helloworld task bot is moved to the right.

___ c. Click **Next** at the bottom of the page (you may need to scroll down).

TYPE	NAME
<input type="radio"/>	Task Bot Data Consistency with Login.atmx

___ d. Select the connected bot runner device (**VCLASSBASE**) and click the yellow right arrow. The bot runner device is moved to the right.

If the device is showing as disconnected, make sure you are logged into the Enterprise Client as the bot runner user.

This pane shows available bot runners.

- Bot Runner clients are indicated with a Play symbol on a computer.
- Bot Developer clients are indicated with a star and arrow.
- Connected Bot Runner clients are indicated by a green circle with a checkmark.

<input type="checkbox"/>		Disconnected	VCLASSBASE	devuser1
<input type="checkbox"/>		Connected	VCLASSBASE	botuser1

You can deploy and run a bot on multiple Bot Runner clients. For the purposes of this training, you run the bot with only one Bot Runner client.

The **Run Bot Runner Session on control room** option is available so that an administrator can use a Remote Desktop Protocol (RDP) to run a bot on a Bot Runner machine.

- The RDP connection occurs on the Control Room machine as a background process
- The bot does not run in a visible way on the Control Room machine, but then you can see the task in the Task Manager list of running processes.
- The RDP connection ends after the bot runs
- In this case, the BotRunner account is the only one available, and it is connected to the Control Room.

Devices

Run bot runner session on control room

Search name

Available devices (0 of 1)

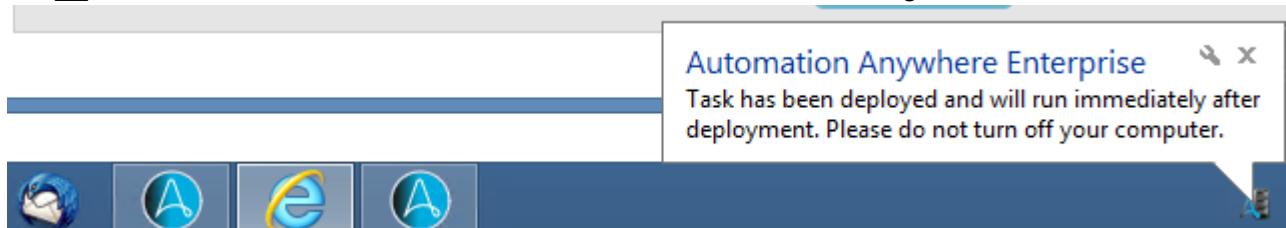
<input type="checkbox"/>	STATUS

Selected (1)

<input type="checkbox"/>	STATUS
<input checked="" type="checkbox"/>	Connected

___ e. Click **Next**.

___ f. Click **Run now**. You see a notification that a task is running.

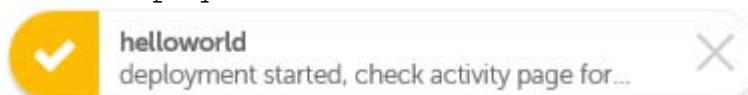


___ g. The helloworld bot runs.

___ h. If prompted, in the Confirm Save As window, click **Yes**.

You should also see a Run Task window pop up with the message:

helloworld deployment started...



You are returned to the **My bots** page.

2.2. Run bot now from the Control Room - leave running

Here, you start the Data Consistency with Login.atmx task, but do not complete running it so that you can see it as an in-progress task. To do this, repeat the steps from the last section but select the Data Consistency with Login task bot.

- __ 1. Click **Bots > My Bots** if not already there.
- __ 2. In the My bots page, run the Data Consistency with Login.atmx bot on the BotRunner account.
 - __ a. Click **Run bot now...**
 - __ b. In the **Run Bot Now** page, select the Data Consistency with Login.atmx Task Bot and click the yellow right arrow. The Data Consistency with Login.atmx task bot is moved to the right.
 - __ c. Click **Next** at the bottom of the page (you may need to scroll down).
 - __ d. Select the connected bot runner device (**VCLASSBASE**) and click the yellow right arrow. The bot runner device is moved to the right.
 - __ e. Click **Next**.
 - __ f. Click **Run now**. You see a notification that a task is running. A prompt window opens and the Run Time Window appears at the bottom right.
 - __ g. Move the “Prompt in ‘Taskbar’” window to the side (it is modal and will stay on top).
 - __ h. In the Run Time Window, click the minimize button.

2.3. Schedule bot to run later

In this section, you schedule the helloworld bot to run in the future. The steps are similar to the last two sections

- __ 1. Click **Bots > My Bots** if not already there.
- __ 2. Schedule the helloworld.atmx bot on the BotRunner account to run later.
 - __ a. Click **Schedule bot...**
 - __ b. In the **Schedule bot** page, select the helloworld.atmx Task Bot and click the yellow right arrow. The helloworld.atmx task bot is moved to the right.
 - __ c. Click **Next** at the bottom of the page (you may need to scroll down).

- ___ d. From the Schedule page, you can elect to run the schedule repeatedly or just once. For the purposes of this exercise, leave **Run once** selected and select a time 1 hour from the present.

Schedule

<input checked="" type="radio"/> Run once	Start date 2018-11-19	Start time 3:30 PM
<input type="radio"/> Run repeatedly		

- ___ e. Select the connected bot runner device (**VCLASSBASE**) and click the yellow right arrow. The bot runner device is moved to the right.
 ___ f. Click **Next**.
 ___ g. Click **Schedule bot**. You are returned to the My bots page.

2.4. Run bot with queue

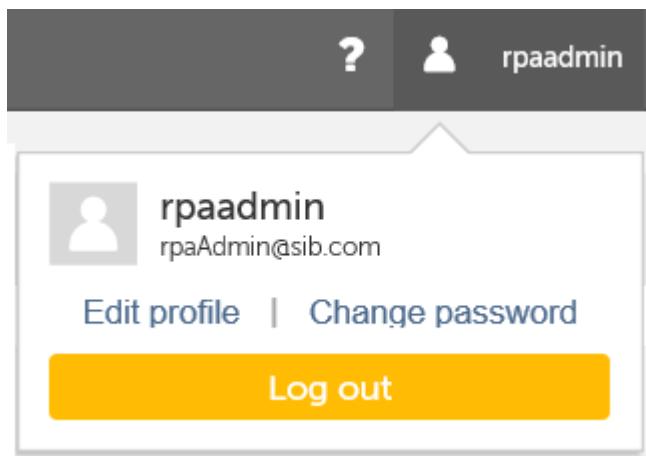
In this section, you log in to the Control Room as `devUser1` (owner of the Transactions queue) and run the `Transactions.atmx` bot using the Transactions queue. You then add two work items to the queue. The bot runs when the work item threshold is met (2).



Information

A Transactions queue, device pool, and work item have been built for use in this exercise.

- ___ 1. Log in to the Control Room as `devUser1`.
- ___ a. Click on **rpaadmin** user id in upper right corner.
 - ___ b. Click **Log out**.



- ___ c. Enter `devUser1` for Username and Password.

- ___ d. Click **Log in**.
- ___ 2. Select bot and review dependencies.
 - ___ a. Navigate to **Workload > Queues**.
The Transactions queue should be in a status of “Not in use.”
 - ___ b. Click **Run bot with queue....**
 - ___ c. In the **Select a Task Bot** section, select **Transactions**.
 - ___ d. Click the **yellow arrow** that is highlighted after selecting the Transactions bot. The bot is moved to the right.

Select a Task Bot

TYPE	NAME
Task Bot	Data Consistency with Login.atmx
Task Bot	helloworld.atmx

→



Transaction

- ___ e. Optionally, you can view any dependencies under the **Review dependencies for Transactions** section. In this case, there are no dependencies for this bot.
- ___ f. Click **Next** (may need to scroll to the bottom of the page).
- ___ 3. Select Queue and Device pool.
 - ___ a. In the **Queue** section, select the **Transactions** queue.
 - ___ b. Click the **yellow arrow** that is highlighted after selecting the **Transactions** bot. The bot is moved to the right.
 - ___ c. In the **Device pool** section, select the **Transactions** device pool.

- ___ d. Click the **yellow arrow** that is highlighted after selecting the **Transactions** device pool. The device pool is moved to the right.

Queue

Select a queue

Queue name ▼	Search queue name	
--------------	-------------------	--

Available queues (0 of 1)

STATUS	QUEUE NAME ↑	MY ACCESS	AUTOMATION NAME	



Transactions

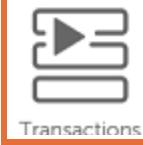
Device pool

Select a device pool

Search device pool name	
-------------------------	--

Available device pools (0 of 1)

STATUS	DEVICE POOL NAME ↑	DETAILED STATUS	# OF AUTOMATIONS	



Transactions

- ___ e. Click **Next**.

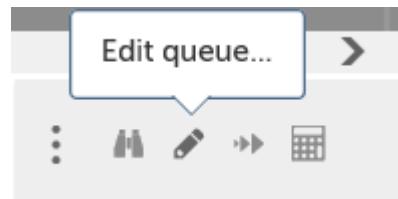
- ___ f. Click **Run now**. You are returned to the Queues page displaying the Transactions queue status as “In use.”

	STATUS	QUEUE NAME ↑	MY ACCESS	AUTOMATION NAME
	In use	Transactions	Queue owner	Transactions.18.11.26.11.23.

- ___ 4. Edit queue and add work item.

- ___ a. Mouse over the ellipsis for the **Transactions** queue.

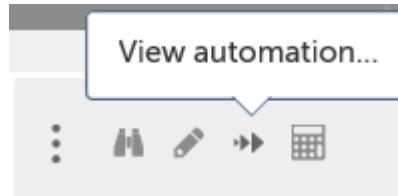
- ___ b. Click the **Edit queue...** icon.



- ___ c. On the **Edit queue** page, click the **General** tab. Notice the minimal number in queue to resume processing is set to **2** work items.

- ___ d. Click **Work item structure**. Here you can view the structure of the work item to be inserted into the queue. This was predefined for this class. Notice it matches the structure used by the **Transactions** bot when processing items from a spreadsheet.

- ___ e. Click **Add work items**.
 - ___ f. Click **Browse**.
 - ___ g. Navigate to: C:\labfiles\transactions
 - ___ h. Double-click the one_work_item.csv file.
 - ___ i. Click **Save changes**. You are taken back to the Queues page.
- ___ 5. View status of Transactions automation.
- ___ a. Mouse over the ellipsis for the **Transactions** queue.
 - ___ b. Click the **View automation...** icon.



- ___ c. In the **RUN DETAILS** section, notice **Number of pending work items** equals 1. You may need to wait and refresh the page before it shows up. Note you can also Pause or Stop the queue from this page.

RUN DETAILS	
Number of active work items	Number of failed work items
0	0
Number of pending work items	Number of completed work items
1	0

- ___ d. Click **Back**.
- ___ 6. Edit queue and add another work item.
- ___ a. Mouse over the ellipsis for the **Transactions** queue.
 - ___ b. Click the **Edit queue...** icon.
 - ___ c. Under the **Work items** section. Notice the work item you inserted. Since the item has not been processed yet, it is in a **Ready to run** state.

	ID	STATUS ↑	DATE	ACCT_NUM	TYPE	AMOUNT
<						
<input type="checkbox"/>	42	Ready to run	1/26/2018	87431	DEBIT	30000

- ___ d. Click the **Add Work Items** tab.
- ___ e. Click **Browse**.

- ___ f. Navigate to: C:\labfiles\transactions
 - ___ g. Double-click the one_work_item.csv file.
 - ___ h. Click **Save changes**. You are taken back to the Queues page.
- ___ 7. View status of Transactions automation.
- ___ a. Mouse over the ellipsis for the **Transactions** queue.
 - ___ b. Click the **View automation...** icon.
 - ___ c. In the **RUN DETAILS** section, notice **Number of active work items** equals 2. You may need to wait and refresh the page before it shows up.

RUN DETAILS	
Number of active work items 2	Number of failed work items 0
Number of pending work items 0	Number of completed work items 0

- ___ d. Click **Back**.
- ___ e. Mouse over the ellipsis for the **Transactions** queue.
- ___ f. Click the **View queue...** icon.



- ___ g. After the Transactions bot runs, hit the **F5** key to refresh the browser. You now see the work items in a **Active** status.

<input type="checkbox"/>	ID	STATUS ↑	DATE	ACCT_NUM	TYPE	AMOUNT
<input type="checkbox"/>	42		Active	1/26/2018	87431	DEBIT
<input type="checkbox"/>	43		Active	1/26/2018	87431	DEBIT

- ___ h. Click **Back**.
- ___ i. Mouse over the ellipsis for the **Transactions** queue.
- ___ j. Click the **View automation...** icon

- ___ k. Click **Stop** to stop the automation. This frees up the queue and any attached device pools. When presented with a dialog box, click **Accept**.

Section 3. Viewing bot status

3.1. Viewing bot status

After running the bots, you can view their status under the Activity section. Within the Activity section, you can view bots in progress, bots that are scheduled, and historical activity.

- ___ 1. View bots in progress
 - ___ a. Click **Activity > In progress**
 - ___ b. You should see the Data Consistency with Login.atmx bot in a status of “Paused for input.” Note, from this page you can pause, resume, and stop bots that are in progress.

The screenshot shows the 'In progress activity' section of the Control Room. At the top, there is a navigation bar with 'Activity > In progress'. Below it is a header with the title 'In progress activity' and a 'Run bot with queue...' button. A search bar with 'Status' and 'Choose status' dropdowns is present. The main area is titled 'Activity (1 of 1)' and contains a table with columns: STATUS, PROGRESS, and ACTIVITY. The table shows one row for a bot named 'Paused for input' which is at 3% completion. There are icons for pausing, resuming, and stopping the bot.

Status	PROGRESS	ACTIVITY
<input type="checkbox"/> Paused for input	3% Paused for input	Run bot

- ___ c. Enter 1234 in the “Prompt in ‘Taskbar’” window and click **OK**.

Note there is only one bot runner on this machine. Because of this, the bot runner is attempting to actively run

- ___ d. The status will turn to Active and the bot runs as normal.
 - ___ e. When the bot completes, it presents a dialog box to the user and is in a “Paused for input” status again.
 - ___ f. Click **OK** in the dialog pop up window. The window disappears along with the bot from the In progress screen.
- ___ 2. View bots that are scheduled.
- ___ a. Click **Activity > Scheduled**

- ___ b. You should see the `helloworld.atmx` bot scheduled to be run one time.

Activity > Scheduled

Scheduled activity

Run bot now... **Run bot with queue...** **Cancel**

You can put a scheduled bot on hold and it will not run for any future occurrence. If it is currently running, you must go to the In progress tab to pause it. If a scheduled bot is on hold, you can remove the hold and return it back to the "Ready to run" state.

Activity (1 of 1)					
Type	Next Occurrence	Activity Name	Bot Name	Scheduled	
<input type="checkbox"/> One time	15:30:00 EST 2018-11-19	helloworld.18.11.19.15...	helloworld.atmx	On 2018-11-19 at 15:30:00 EST	

Depending on timing, if you are close to the one hour to run the bot, you can see the bot run and the scheduled activity will then be deleted. Once the bot has completed, you can see the results of the bot run in the Historical Activity section.

- ___ 3. View historical activity for all bots.

- ___ a. Click **Activity > Historical**

- b. If the scheduled **helloworld** bot completed, you will see that in the Historical activity pane along with the history of all other bot runs.

Historical activity

Status	▼	Choose status			
Activity (7 of 7)					
■	STATUS	DEVICE NAME	AUTOMATION NAME	BOT NAME	USE
<input type="checkbox"/>	Completed	VCLASSBASE	Transactions.18.12.0...	Transactions.at...	bot
<input type="checkbox"/>	Completed	VCLASSBASE	Data Consistency wit...	Data Consisten...	bot
<input type="checkbox"/>	Completed	VCLASSBASE	helloworld.18.12.03.1...	helloworld.atmx	bot
<input type="checkbox"/>	Completed	VCLASSBASE	Transactions.18.11.30...	Transactions.at...	bot

End of exercise

Exercise review and wrap-up

In this exercise, you learned about some of the administrative features in the Control Room. You ran a bot from the Control Room and scheduled a bot to run on the runtime user account. You also configured a bot to run in a queue and added two work items. After running the bots, you also viewed the status of bots in various stages of execution.



IBM Training



© Copyright International Business Machines Corporation 2018, 2019.