Everybody Gets a Screenshot!

Automating Screenshots for Linguistic Testing

Luz Ma. Pineda - Sr Technical Project Manager Localization
Geneviève Bolduc - Localization Developer

MOTOROLA SOLUTIONS

avigilon™

AGENDA

1. Introduction
2. Software Localization Challenges
3. Risk Mitigation
4. Linguistic Testing
5. Automating Screenshots
6. Q & A

Introduction

Who we are

2013

Paula Hunter
Sr Manager,
Localization & Tech Doc

Luz Pineda
Sr Technical Project Manager,
Localization

Eduardo La Selva
Project Manager,
Localization

Geneviève Bolduc
Localization Developer

Sammy Kaspar
Localization Developer
(Co-op/Intern)

weLocalize doing things differently · adapt WORLDWIDE · madtranslations

Avigilon In-Country Reviewers
(Global Sales Team / Partners)

Introduction

Content types we localize

Technical Documentation

- Software User Guides
- Hardware Installation Guides
- Hardware Operations Guides
- Datasheets / Spec sheets
- In-box Materials
  A& Specs

Marketing and Digital Experience

- Websites, Landing pages
- Fact Sheets, Brochures, Catalogs
- Videos
- Ads, Social Media and Emails,
- Support Community, Partner Portal and eCommerce

Software UI

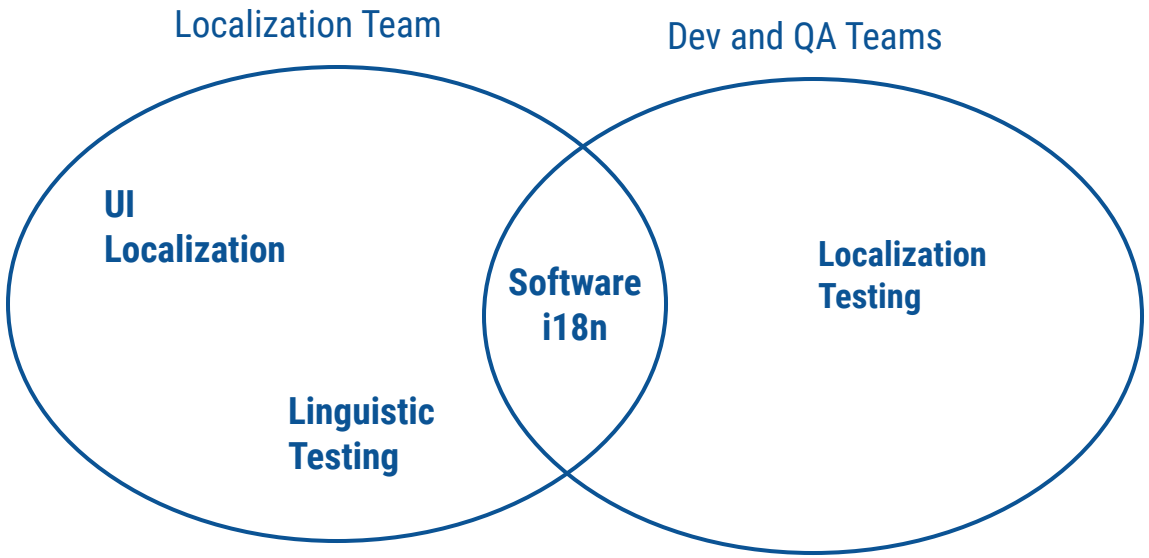
- ACC, ACC Mobile
- ACS + all Microservices
- ACM, ACM Mobile
- CCT, SDT
- Camera WebUI
- MSI Compass
- Open Path

Software Localization

Scope and QA

Our team is responsible for:

- Providing accurate translations
- Verifying that the software UI displays correctly in all the supported locales.

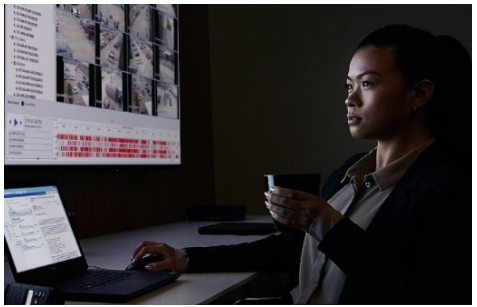


Software Localization Challenges

Level of Quality and Error Tolerance

Quality

The nature of our software requires a high degree of technical accuracy.



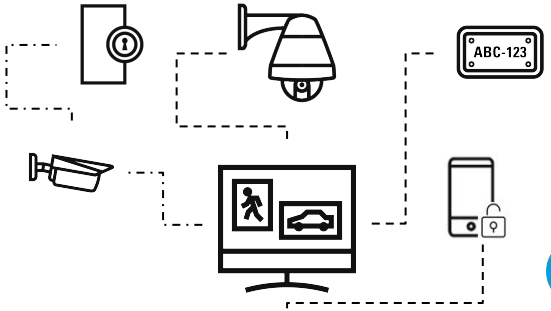
Industry Specific Terminology

Terminology used in our industry leaves room for ambiguity if no context is provided.

Testing Environments

Testing environments similar to the ones end users will use:

- Complexity
- Cost
- Privacy



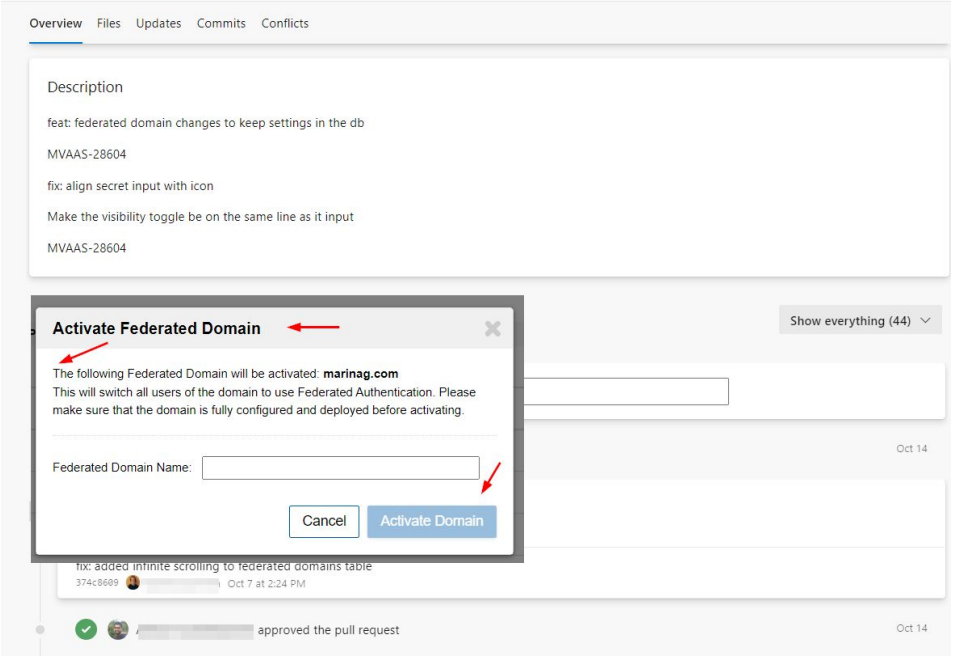
Automation Testing

Risk Mitigation - Early QA

Source Language Review

The Localization team is included in all the pull requests with changes in the resource file. We ask the developers to include:

- Screenshots showing the changes in the UI
- Comments on the strings being introduced

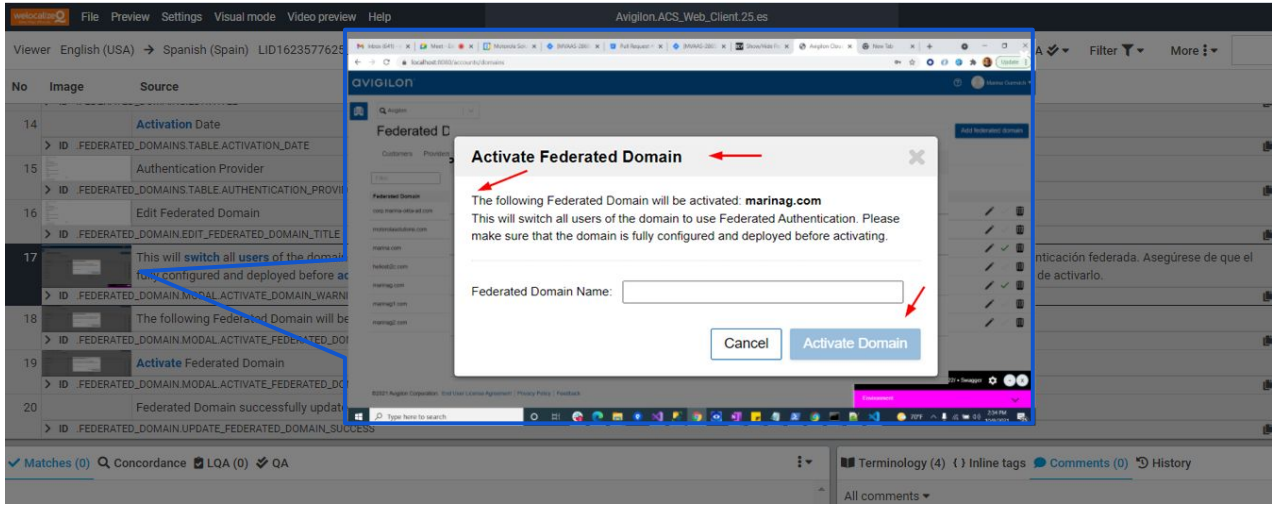


Automation Testing

Risk Mitigation - Context

Context for Translation

The same screenshot is sent to the translators to provide context to the strings within scope



Automation for Linguistic Testing

Pros and Cons

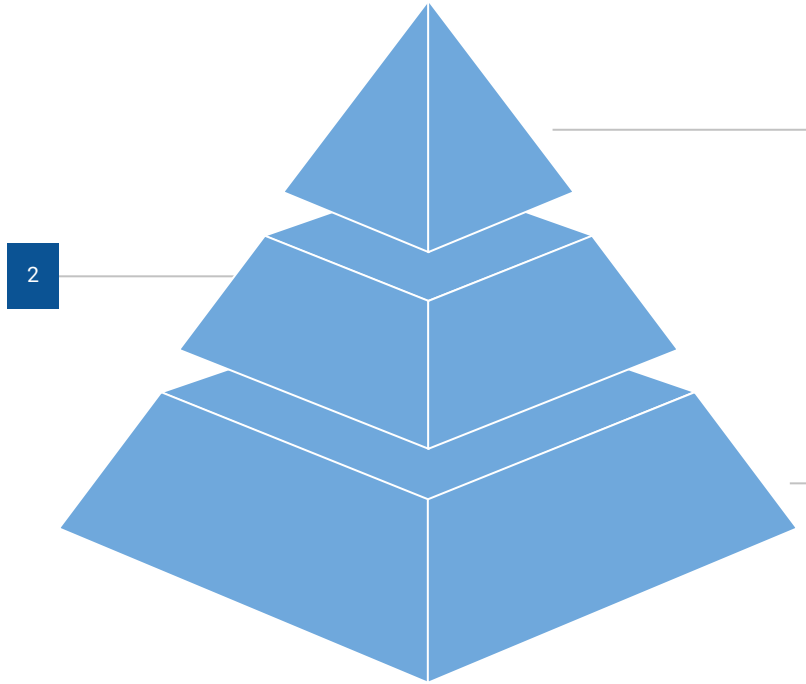
PROS	CONS
It is possible to reuse existing testing environments	It takes time to develop the scripts.
It makes Linguistic Testing affordable	We don’t have the navigational feedback that you get when testers execute test cases manually.
Increase Testing Coverage	It is not possible to cover all the strings with automation.

Software Localization

Linguistic testing - Priorities

Medium Priority

Strings visible when executing basic functionality, warning messages and Configuration views..



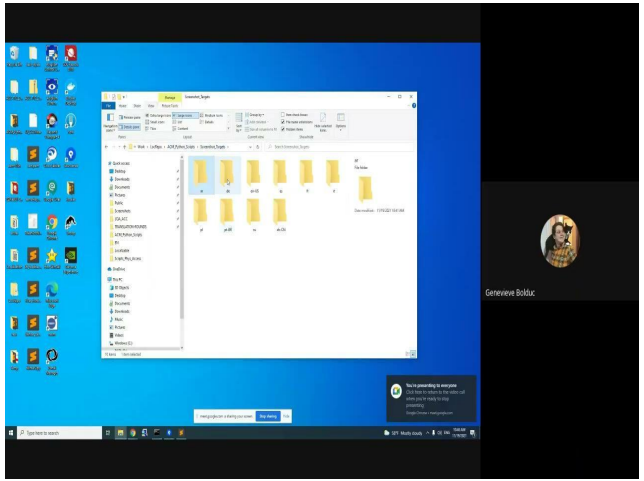
High Priority Strings

Strings visible to any user while simply navigating through the UI, Feature priority and Escalations

Low Priority

Error messages for edge cases, strings visible to a smaller group of users, strings that historically haven't caused layout issues.

Automating Screenshots



Automating Screenshots

1. Infrastructure
    - Creating a Map of Your UI
    - Working within the Codebase vs. Independent Testing Suites
    - Web Apps and Python/Selenium Automation
    - Desktop Apps and Python/PyWinAuto Automation
2. Iterative Process
    - Collecting Steps & Screenshots Iteratively
3. Limitations

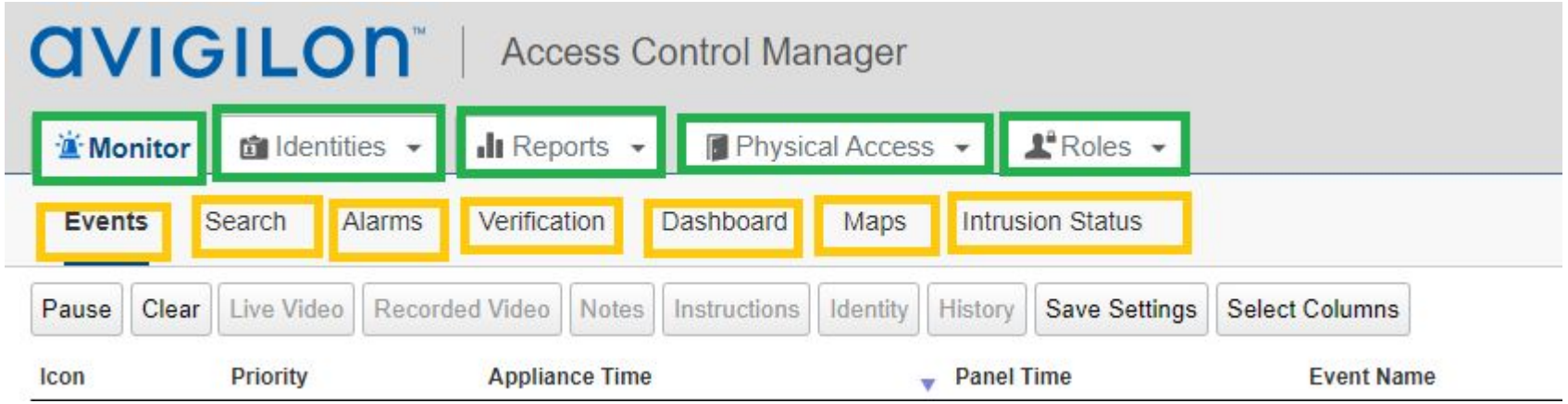
Creating a Map of the UI

1. **Identifying the Primary Views**
   - What are all the pages or views users can immediately access by:
     - Logging in?
     - Clicking on the site menu?
     - Clicking on the visible tabs?

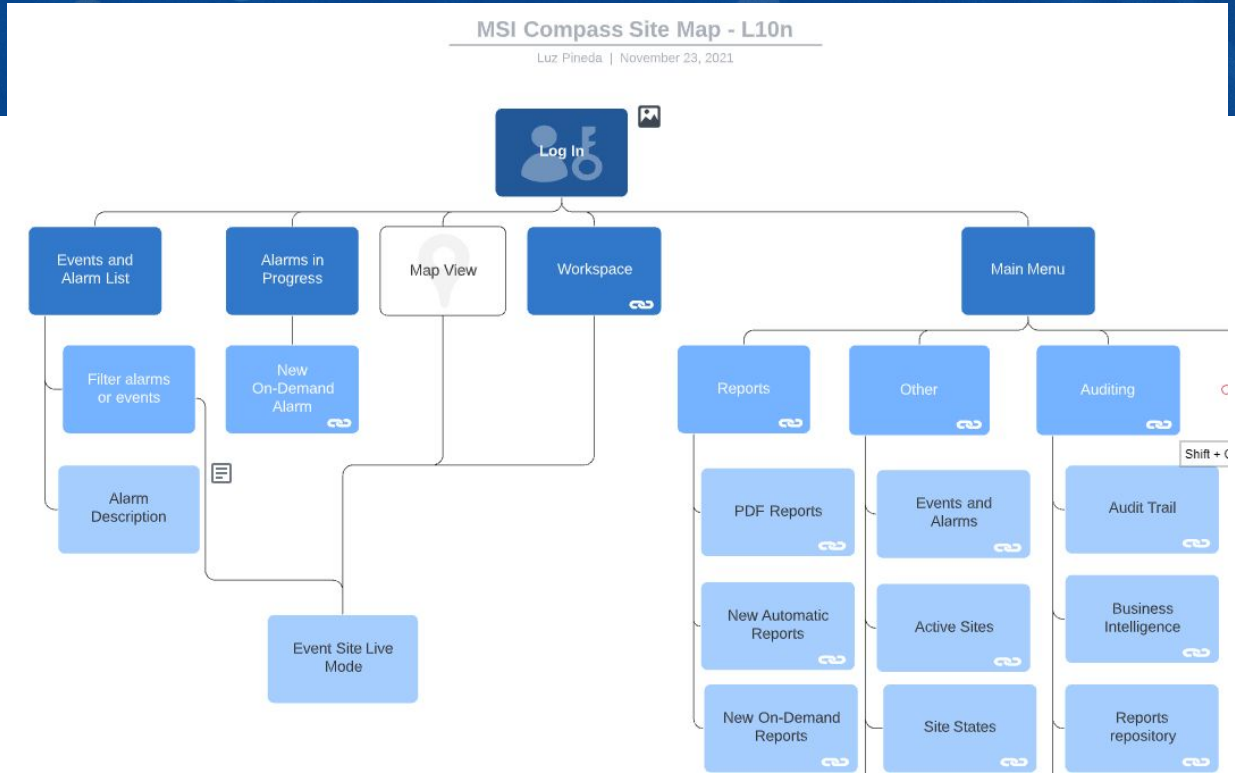
   - How can the text on these pages be maximized?
     - Is there a drop-down menu you can open?
     - Are there tabs you can open up?

2. **Build Test Cases for these Views**
   - From the information surmised above, you can create a list of main views and any triggers to maximize text on the screen.

Creating a Map of the UI



Creating a Map of the UI



Working within the Codebase vs Independent Testing Framework

Codebase

PROS	CONS
Maximizes existing infrastructure (using existing methods, class, etc)	Processes can be slowed by build compilation errors that do not pertain to localization
Codebase is maintained and impact to localization scripts is noticeable by non-localization developers	Dependent on development teams if issues arise

Working within the Codebase vs Independent Testing Framework

Independent Testing Framework

PROS	CONS
Autonomy	Responsibility to maintain and adapt to changes in UI.
No external dependencies for compilation	Higher Setup Overhead
Choice of tools and libraries	No access to pre-existing methods from codebase

Web Apps & Python/Selenium

1. **Python**
   - High level and object-based scripting language
   - User-friendly
     - Uses English keywords that requires very easy interpretation (readable)
     - Has few syntax complications compared to other languages for programming.

2. **Selenium**
   - Set of tools that firmly supports the quick development of test automation of web applications.
   - Offers testing functions that are specially designed to the requirements of testing of a web application.

Web Apps & Python/Selenium

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome('./chromedriver')
driver.get("https://www.python.org")
print(driver.title)
search_bar = driver.find_element_by_name("q")
search_bar.clear()
search_bar.send_keys("getting started with python")
search_bar.send_keys(Keys.RETURN)
print(driver.current_url)
driver.close()
```

Source: https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test

Web Apps & Python/Selenium

Navigating through DOM Elements

Identifying Elements can be done by:

- Name: driver.find_element_by_name("name")
- CSS ID: driver.find_element_by_id(“id-search-field”)
- DOM Path: driver.find_element_by_xpath(“//input[@id=’id-search-field’]”)
- CSS Class: driver.find_element_by_class_name(“search-field”)

In our L10n team, we have found CSS ID and Class to be the most robust:

- Name can change with language change
- Xpath can change if different positionings surface

Desktop Apps & Python/pywinauto

1. **Python**
   - High level and object-based scripting language
   - User-friendly
     i. Uses English keywords that requires very easy interpretation (readable_
     ii. Has few syntax complications compared to other languages for programming.

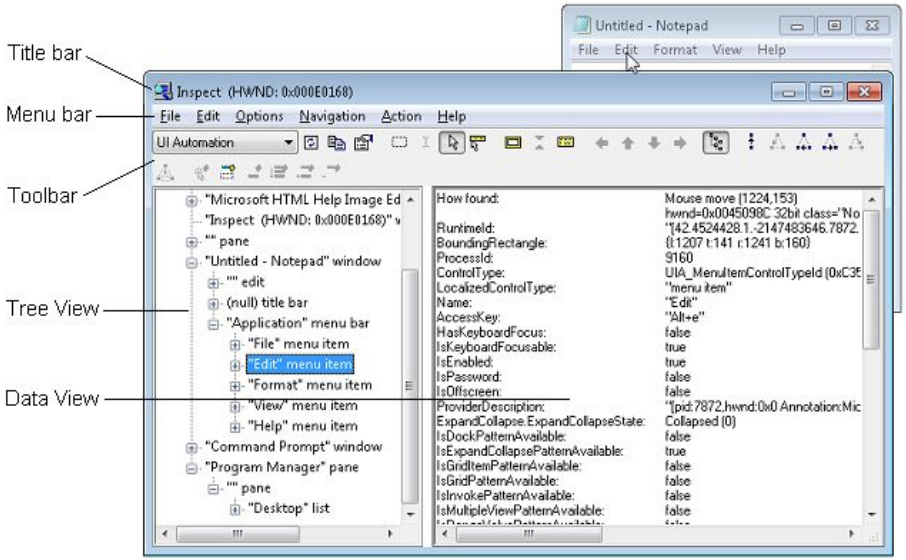
2. **pywinauto**
   - pywinauto is a set of python modules to automate the Microsoft Windows GUI.
   - Simple: Allows you to send mouse and keyboard actions to windows dialogs and controls
   - Complex: Offers support for more complex actions like getting text data.

Desktop Apps & Python/pywinauto

GUI Tools

GUI Tools:

- Inspect.exe
- Spy++
- Py_inspect



Desktop Apps & Python/pywinauto

Finding the Elements

Pywinauto Methods:

- print_control_identifiers()

How to know magic attribute names

There are several principles how "best match" gold names are attached to the controls. So if a window specification is close to one of these names you will have a successful name matching.

1. By title (window text, name): `app.Properties.OK.click()`
2. By title and control type: `app.Properties.OKButton.click()`
3. By control type and number: `app.Properties.Button3.click()` (*Note*: Button0 and Button1 match the same button, Button2 is the next etc.)
4. By top-left label and control type: `app.OpenDialog.FileNameEdit.set_text("")`
5. By control type and item text: `app.Properties.TabControlSharing.select("General")`

Source: https://pywinauto.readthedocs.io/en/latest/getting_started.html#how-to-know-magic-attribute-names

Desktop Apps & Python/pywinauto

```
#External Directory
mn.Go_To_Users_Groups_Ext_Directory(acc, lang)
mu.Screenshot(acc, lang, "External_Directory")
close_bttn = mu.Localize_String("Close", lang)
acc.window(best_match= close_bttn).click_input()
```

```
def Localize_String(string, language):
    en_path = ACC_Resources + "\\strings.txt"
    element_key = ""
    with open(en_path, 'r', encoding="UTF-8-sig") as f:
        lines = [line for line in f.readlines() if line.strip()]
        for line in lines:
            if not line.startswith("#"):
                line = line.strip()
                key = line.split(',', 1)[0]
                val = line.split(',', 1)[1]
                if val == string:
                    element_key = key
                    break
    language_path = ACC_Resources +"\\strings-" + language + ".txt"
    with open(language_path, 'r', encoding="UTF-8-sig") as g:
        lines = [line for line in g.readlines() if line.strip()]
        for line in lines:
            if not line.startswith("#"):
                line = line.strip()
                key = line.split(',', 1)[0]
                val = line.split(',', 1)[1]
                if key == element_key:
                    string = val
                    break

    return string
```

Common Methods Across Platforms

Methods to Get Started

1. **Application Methods**
    - Method to launch application/browser
    - Method to Login
    - Method to Select the Language
    - Method to Take Screenshot
        i. Libraries: Screenshot, pyautogui
2. **Manipulating Elements Methods**
    - Hovering over Elements
    - Clicking
    - Sending Keys
    - Selecting Items from a Dropdown

Iterative Processes

Keeping up with Agile

Once this infrastructure and these views are in place the goal is to keep in sync with production. How can this be achieved?

1. **As soon as you know the string is coming (design phase or PR):**
    - Collect the steps
    - Collect any reference images
    - Write the Script
    - There is NO NEED to wait until the strings are translated to write these
2. **When the strings are translated:**
    - Run the script

Limitations & Path Forward

1. **Limitations:**
   - **Writing the Script is Manual Process**
   - **Running the Script is Manual Process**
   - **Readiness of Script depends on Early Access of Strings**

2. **Path Forward:**
   - **Adding Screenshot Automation for Mobile apps**
   - **Automated Triggers**

Q & A