Sage 300 Web Screens SDK

Proxy Tester Utility

October 2024

The MIT License (MIT)

Copyright © 2024 The Sage Group plc or its licensors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

[1. What is the Sage 300 Proxy 5](#_Toc180595510)

[2. What is the Proxy Tester 6](#_Toc180595511)

[3. Web Application Prerequisite 7](#_Toc180595512)

[3.1 Web.config Enhancement 7](#_Toc180595513)

[4. Proxy Tester Prerequisites 8](#_Toc180595514)

[4.1 Required Properties to be changed 8](#_Toc180595515)

[4.1.1 Target Server 8](#_Toc180595516)

[4.1.2 Proxy Tester Server 8](#_Toc180595517)

[4.2 Optional Properties to be changed 9](#_Toc180595518)

[4.2.1 User 9](#_Toc180595519)

[4.2.2 Password 9](#_Toc180595520)

[4.2.3 Company 9](#_Toc180595521)

[4.2.4 ModuleId 9](#_Toc180595522)

[4.2.5 Controller 9](#_Toc180595523)

[4.2.6 Action 9](#_Toc180595524)

[4.2.7 OptionalParameters 9](#_Toc180595525)

[4.2.8 Source 10](#_Toc180595526)

[5. Diagrams 11](#_Toc180595527)

[5.1 The Proxy 11](#_Toc180595528)

[5.1.1 ProxyPublicKey method 11](#_Toc180595529)

[5.1.2 ProxyMenu method 12](#_Toc180595530)

[5.1.3 ProxyScreen method 13](#_Toc180595531)

[5.2 The Proxy Tester 14](#_Toc180595532)

[5.2.1 Testing a Menu 14](#_Toc180595533)

[5.2.2 Testing a screen 15](#_Toc180595534)

[5.3 Elliptical Encryption 16](#_Toc180595535)

[5.3.1 ECDiffieHellmanCng 16](#_Toc180595536)

[5.3.2 Sample Workflow 17](#_Toc180595537)

[6. Running the Utility 18](#_Toc180595538)

[7. Proxy Tester Methods Explained 19](#_Toc180595539)

[7.1 Home Controller 19](#_Toc180595540)

[7.1.1 Get Menu 19](#_Toc180595541)

[7.1.2 Get Screen 19](#_Toc180595542)

[7.1.3 Proxy Public Key 20](#_Toc180595543)

[7.1.4 Proxy Menu 21](#_Toc180595544)

[7.1.5 Proxy Screen 22](#_Toc180595545)

[7.1.6 Encrypt Items 23](#_Toc180595546)

[7.1.7 Prep Model 23](#_Toc180595547)

[7.1.8 Build URL 24](#_Toc180595548)

[7.2 Proxy Tester Behavior JavaScript 25](#_Toc180595549)

[7.2.1 Init Buttons – Test Menu and Test Screen Clicks 25](#_Toc180595550)

[7.2.2 Assign Source 26](#_Toc180595551)

[7.2.3 Ajax Post 26](#_Toc180595552)

1. What is the Sage 300 Proxy

The Sage 300 Proxy (Proxy) is a gateway to the Sage 300 Web Screens in which the consumer of the Proxy signs in without a UI and then is allowed to display a Sage 300 Web Screen in the consumer’s web application.

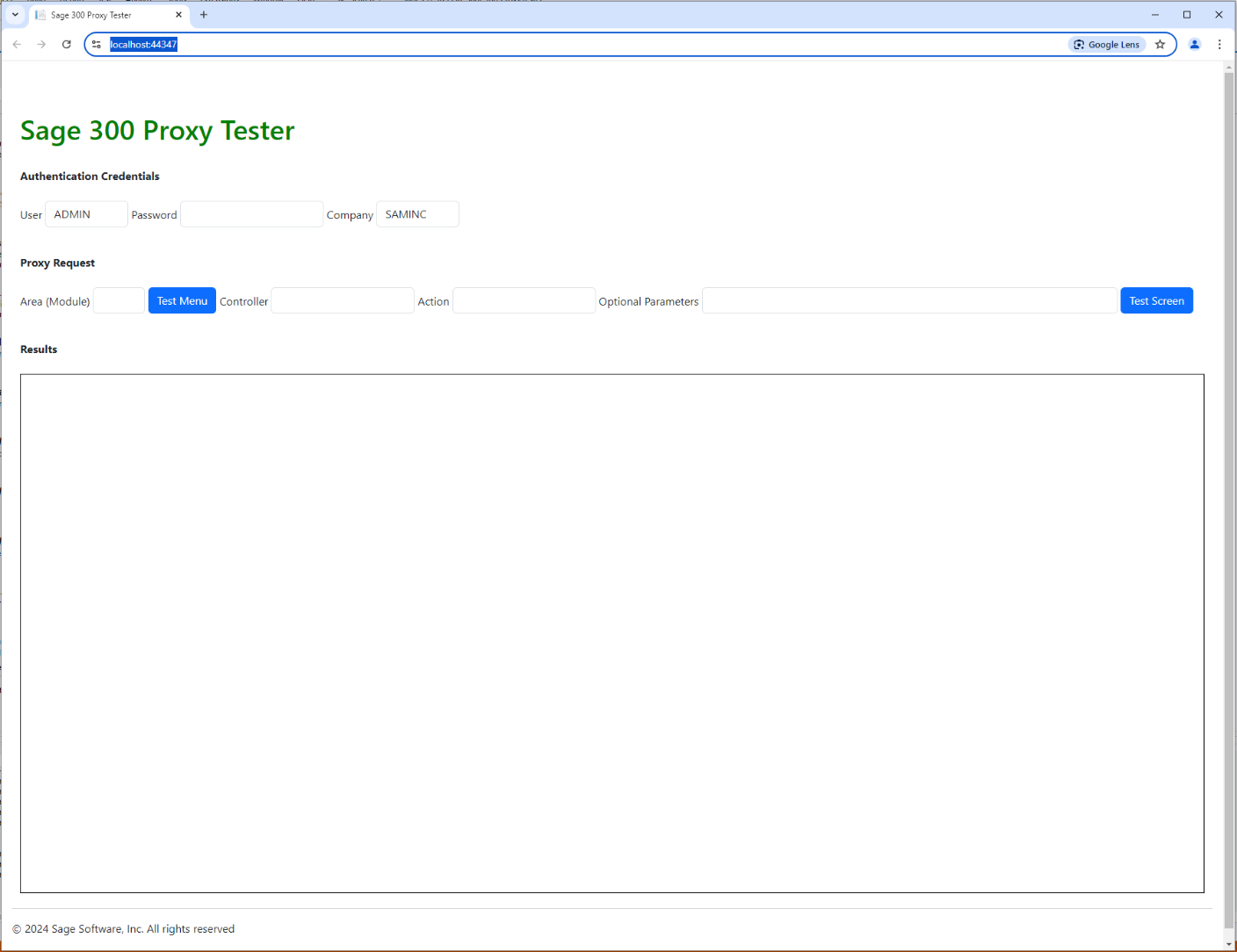
The Proxy is a process that facilitates the encryption and decryption of credentials, logs into the application, and allows for the display of the requested screen.

The Proxy will allow any web application that supplies the correct credentials for a Sage 300 user to have the ability to host a Sage 300 web screen in their iFrame.

The Proxy uses Elliptic-Curse Cryptography (ECC) for asymmetric encryption. Unlike the proxy that Sage CRM uses to interact with Sage 300 where credentials are passed in the query string, this Proxy only passes encrypted credentials and parameters in the request header for added security.

1. What is the Proxy Tester

The Proxy Tester (tester) is an ASP.NET MVC 5 application. It provides the partner with an example of how to invoke the Proxy while providing valuable code snippets that can be used by the partner in the development of code in their application.



The tester provides simple inputs for Sage 300 credentials (User, Password, Company) and the requested web screen information (Area/Module, Controller, Action, and Optional Parameters).

The Proxy also can return a JSON payload of a module’s menu items. This is a requirement of our HRMS team as they require this data, and I do not envision that Sage 300 partners will require this feature. But it is still available!

When the required inputs are satisfied, the request is made to the Proxy and the screen, or JSON payload, is displayed in the iFrame at the bottom of the tester.

1. Web Application Prerequisite

For the Sage 300 Web Application to allow another application to host a screen, the application must be told the domain from which the request is to be made. This is how our Sage 300 – CRM Integration works when the Sage CRM server is on a different domain. This Proxy is no different.

In a future release, the Sage 300 – CRM integration will be enhanced to use this Proxy instead of the current proxy which is CRM specific.

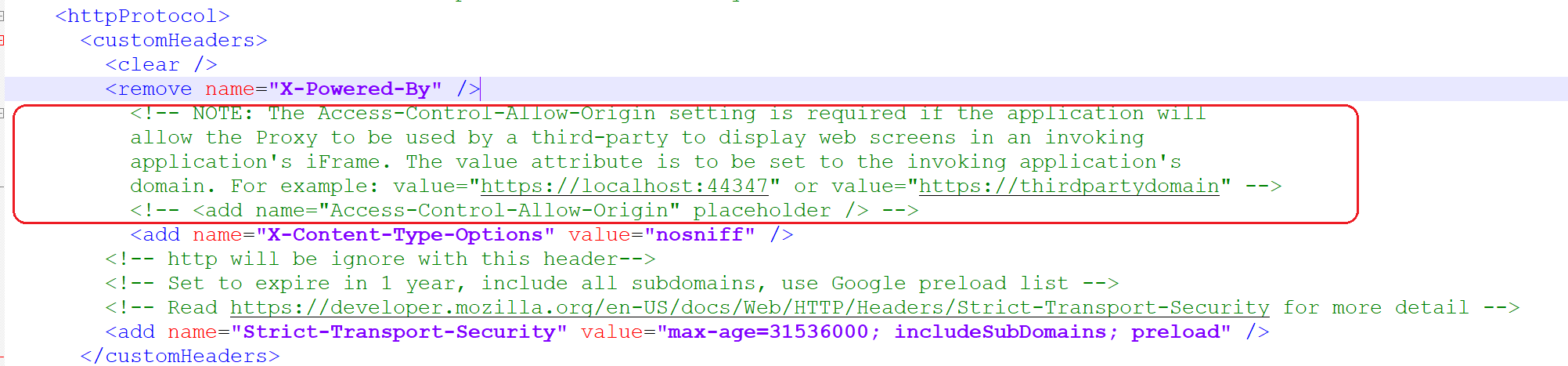
Without this information, the request will fail with a Cross-Origin Resource Sharing (CORS) complaint.

CORS is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, port) other than its own from which a browser should permit loading resources.

* 1. Web.config Enhancement

The Sage 300 web screen’s web.config file must be modified to include the domain which will be allowed by the Proxy.

The web.config file is in the ..\Online\Web folder and has the following commented section:



To enable the allowed domain, it needs to be specified in the ***value*** attribute.

Modify the ***value*** attribute to specify the domain of the tester (in this example <https://localhost:44347>) so that Sage 300 will allow this new domain:

<add name="Access-Control-Allow-Origin" value="https://localhost:44347" />

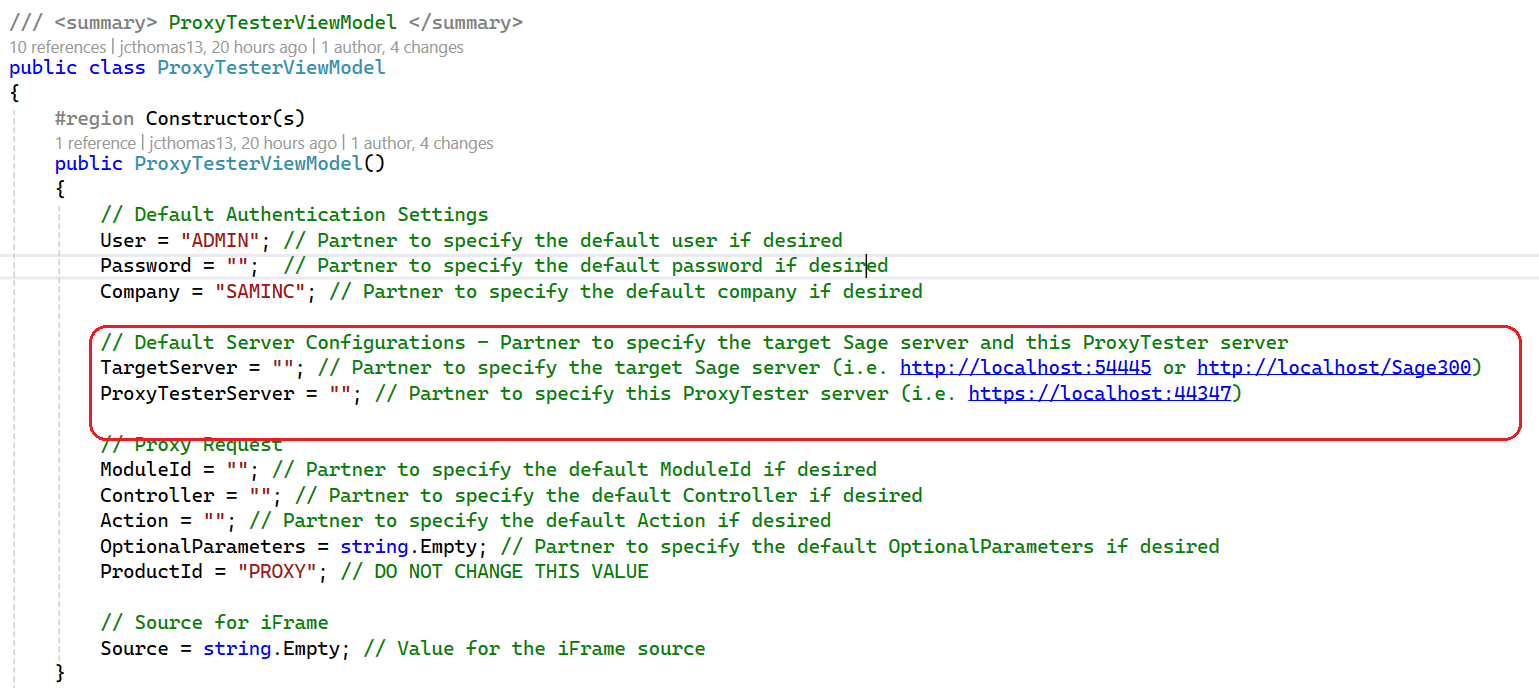
1. Proxy Tester Prerequisites

The tester has several properties that MUST be specified other than those that are provided as inputs on the tester screen.

There are required properties that must be set and optional properties that can be set as defaults for the screen for easier interaction.

The properties were established in this fashion because the tester is a utility for partners to play with the Proxy, grab code snippets, and look at the implementation to assist them with their implementation in their product. Thus, the tester is simplistic with limited error trapping and validations.

The properties that must and should be set prior to running the utility are found in the ***ProxyTesterViewModel*** class:



* 1. Required Properties to be changed
     1. Target Server

The ***TargetServer*** property is the Sage 300 server that your application or this tester will be hitting. To specify the local Sage 300 installation, the ***TargetServer*** is changed to:

TargetServer = "http://localhost/Sage300";

* + 1. Proxy Tester Server

The ***ProxyTesterServer*** property is the server for the tester. The ***ProxyTesterServer*** is changed to (but may be different on your machine):

ProxyTesterServer = "https://localhost:44347";

* 1. Optional Properties to be changed
     1. User

The Sage 300 user id. Defaulted to “ADMIN” but change as desired for screen default.

User = "ADMIN";

* + 1. Password

The Sage 300 password. Defaulted to “” and do not recommend adding a default for this property.

Password = "";

* + 1. Company

The Sage 300 company. Defaulted to “SAMINC” but change as desired for screen default.

User = "SAMINC";

* + 1. ModuleId

The area (url resource segment) / module (menu) (. Defaulted to “” but change as desired for screen default.

ModuleId = "";

* + 1. Controller

The controller segment. Defaulted to “” but change as desired for screen default.

Controller = "";

* + 1. Action

The action segment. Defaulted to “” but change as desired for screen default.

Action = "";

Most Actions will be ***Index***. However, when it comes to Sage 300 Payroll, this segment will be ***Index/0*** for US Payroll and ***Index/1*** for Canadian Payroll. The 0 and 1 are segments in the Payroll Controller url where the 0 and 1 translate into a module id of UP for US Payroll and CP for Canadian Payroll

* + 1. OptionalParameters

The query string of the url to allow for the screen to be displayed in the context of the screen. Defaulted to “” but change as desired for screen default.

OptionalParameters = string.Empty;

* + 1. Source

The value of the iFrame src attribute. Defaulted to “”.

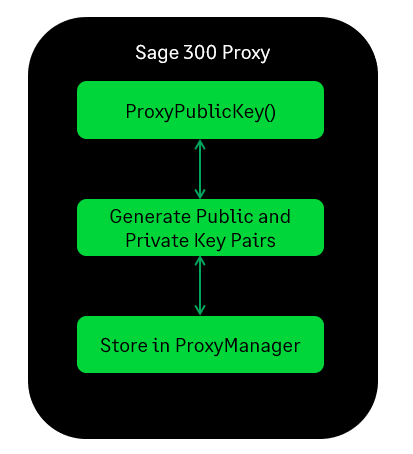
Do not change as this accepts the return content from the proxy.

Source = string.Empty;

1. Diagrams
   1. The Proxy
      1. ProxyPublicKey method

Returns a public key that will be used to encrypt the headers.

Must be invoked for every menu or screen request as headers are encrypted every time with a new public key that is only valid for a single use.



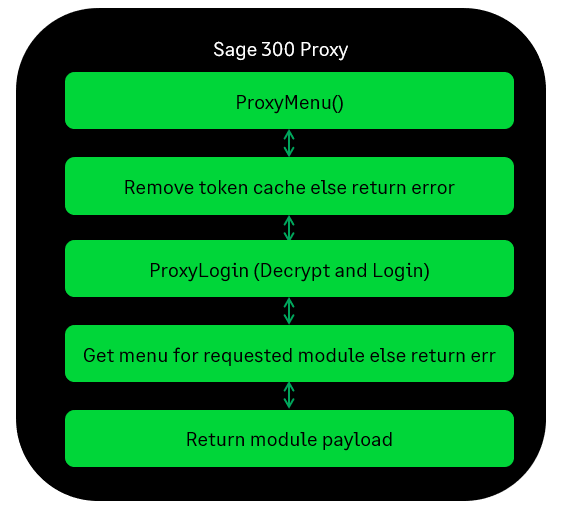
* + 1. ProxyMenu method

Returns a JSON payload of menu items for the requested module.

If the public key is invalid or non-existent the request fails.

If the login fails, the request fails.

The public key is removed/invalidated after every usage and has a limited lifecycle of 5 minutes once created.



* + 1. ProxyScreen method

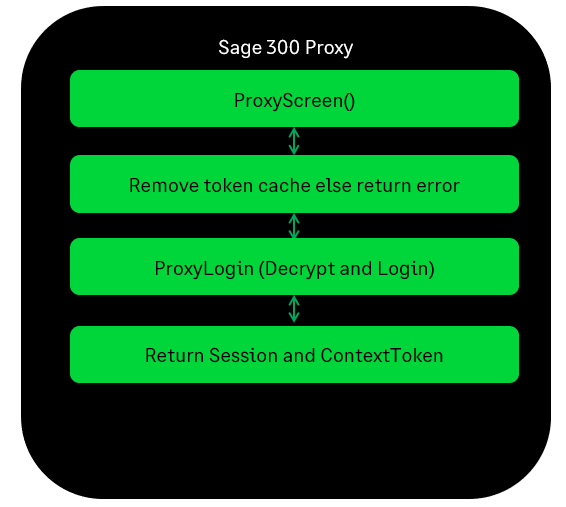
Returns a session id and ContextToken to be used for the requested screen.

If the public key is invalid or non-existent the request fails.

If the login fails, the request fails.

The public key is removed/invalidated after every usage and has a limited lifecycle of 5 minutes once created.

The consumer is responsible for redirecting to the requested URL as well as setting the src attribute in the consumers iFrame.



* 1. The Proxy Tester
     1. Testing a Menu

The tester has a ***Test Menu*** button for testing the Proxy’s ***ProxyMenu*** gateway for requesting a JSON payload for a module’s menu items.

An ajax call is made to the tester for requesting the menu items from the Proxy.

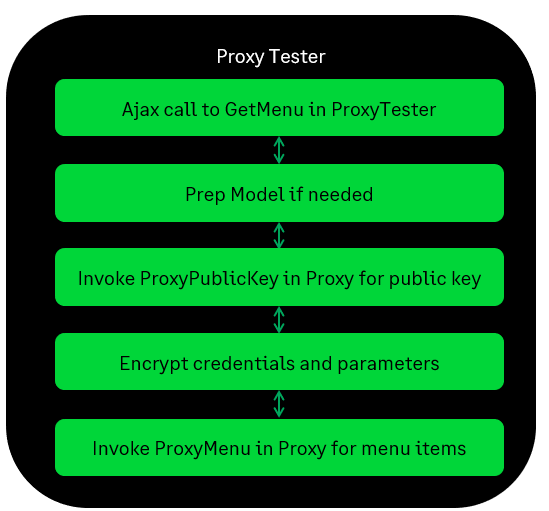
The model is prepped as necessary (i.e. upper casing values, etc.)

The Proxy’s ***ProxyPublicKey*** method is invoked to get a public key used for encryption.

The credentials and any parameters are encrypted using the public key.

The Proxy’s ***ProxyGetMenu*** method is invoked to get a JSON payload of menu items for the requested module

The payload is assigned to the src attribute of the iFrame which displays the contents.



* + 1. Testing a screen

The tester has a ***Test Screen*** button for testing the Proxy’s ***ProxyScreen*** gateway for requesting a Web Screen to be displayed.

An ajax call is made to the tester for requesting the web screen from the Proxy.

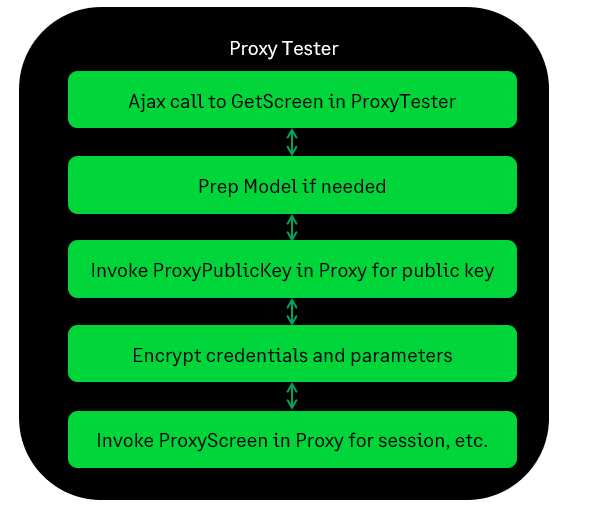
The model is prepped as necessary (i.e. upper casing values, etc.)

The Proxy’s ***ProxyPublicKey*** method is invoked to get a public key used for encryption.

The credentials and any parameters are encrypted using the public key.

The Proxy’s ***ProxyGetScreen*** method is invoked to requesting a Web Screen to be displayed.

The session id and Context Token are returned as these are used in the display of the web screen. A Redirect(URL) is performed prior to assigning the constructed url (in the tester) to the src attribute of the iFrame which displays the web screen.



* 1. Elliptical Encryption

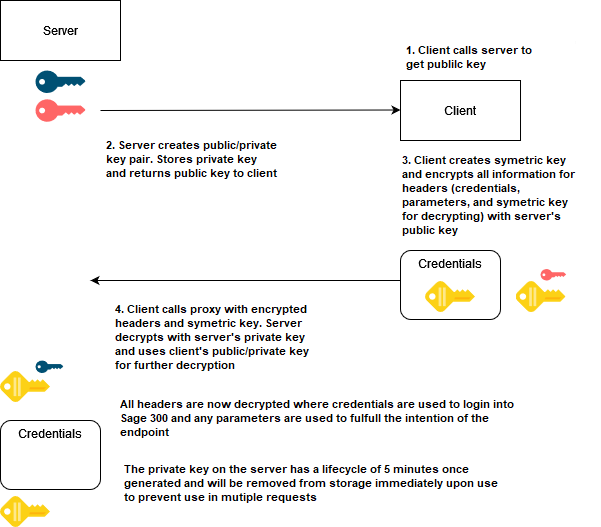
Elliptic-Curve Cryptography (ECC) is an algorithm for asymmetric encryption.

* + 1. ECDiffieHellmanCng
* Smaller keys and signatures
* Faster key creation
* Faster encryption/decryption
* Faster signing/verifying

Graphical user interface, application, Teams

Description automatically generated

* + 1. Sample Workflow



1. Running the Utility

The tester utility is intended to be run by partners in Visual Studio 2022 as it is a utility intended to not only demonstrate interacting with the Proxy but to also provide code snippets and examples to assist partners with development within their applications.

The tester is in the Web SDK in the ***src/utilities/Sage300ProxyTester*** folder.

* Load the **Sage.CA.SBS.ERP.Sage300.ProxyTester.sln** file in Visual Studio 2022.
* Select ***Tools*** 🡪 ***NuGet Package Manager*** 🡪 ***Manage NuGet Packages for Solution***
* ***Restore*** NuGet Packages
* See the ***Proxy Tester Prerequisites*** section in this document for required changes before compiling
* ***Build*** the Solution
* ***Run*** the Solution
* You are now ready to use the tester!

1. Proxy Tester Methods Explained

This section will go into some detail on the files and methods that are of note.

* 1. Home Controller

This is where the button clicks on the tester screen will end up preparing to communicate with the Proxy and, also contains worker methods

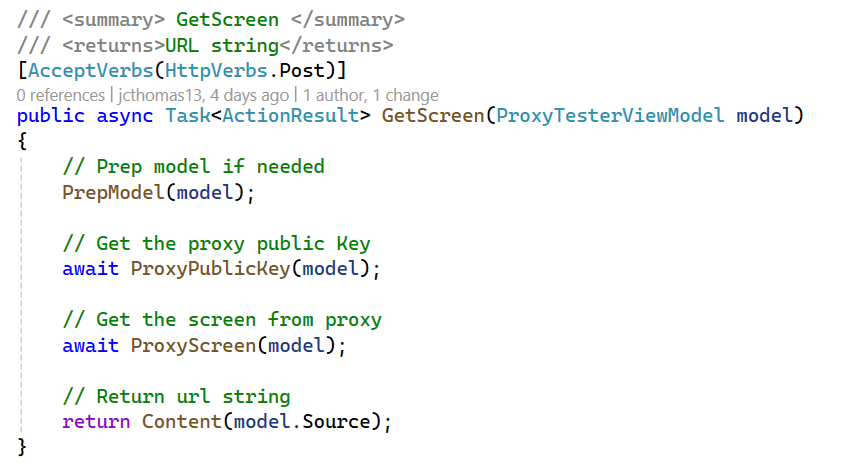
* + 1. Get Menu

The ***GetMenu*** method is invoked with the ***Test Menu*** button is pressed with the model properties from the tester screen



* + 1. Get Screen

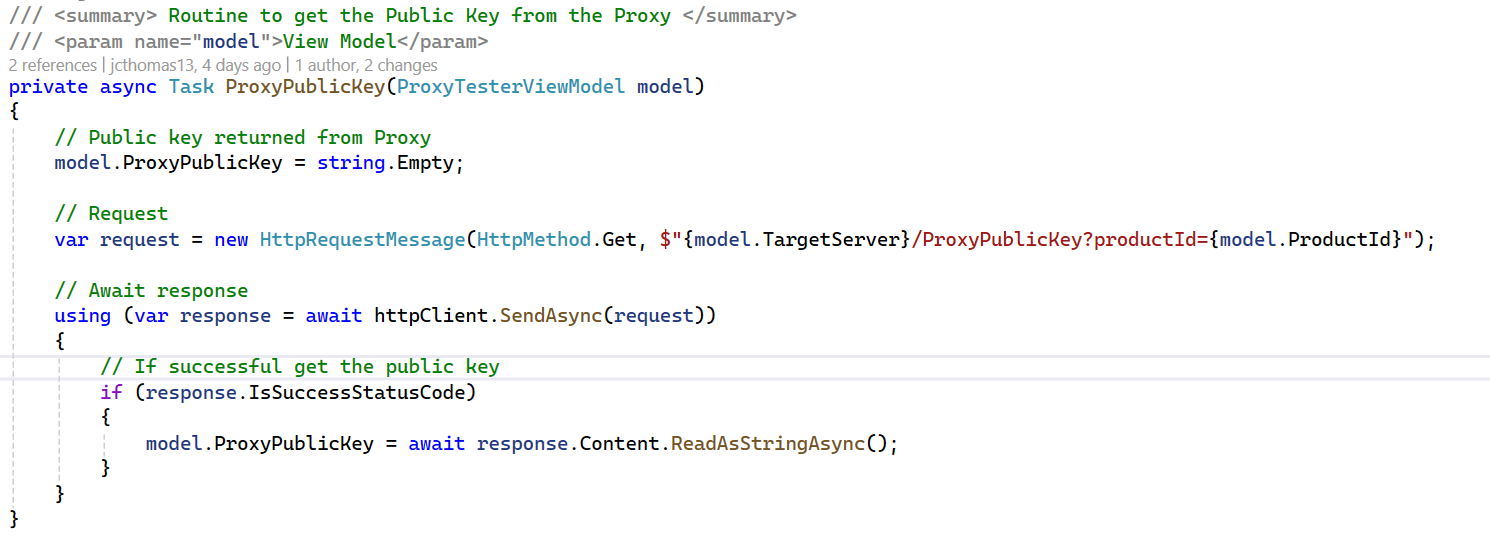
The ***GetScreen*** method is invoked with the ***Test Screen*** button is pressed with the model properties from the tester screen



* + 1. Proxy Public Key

The ***ProxyPublicKey*** method is the actual routine that invokes the Proxy ***ProxyPublicKey*** endpoint.

Upon successful return, it stores this public key from the server that will be used to encrypt the headers before either the ***ProxyMenu*** or ***ProxyScreen*** request.

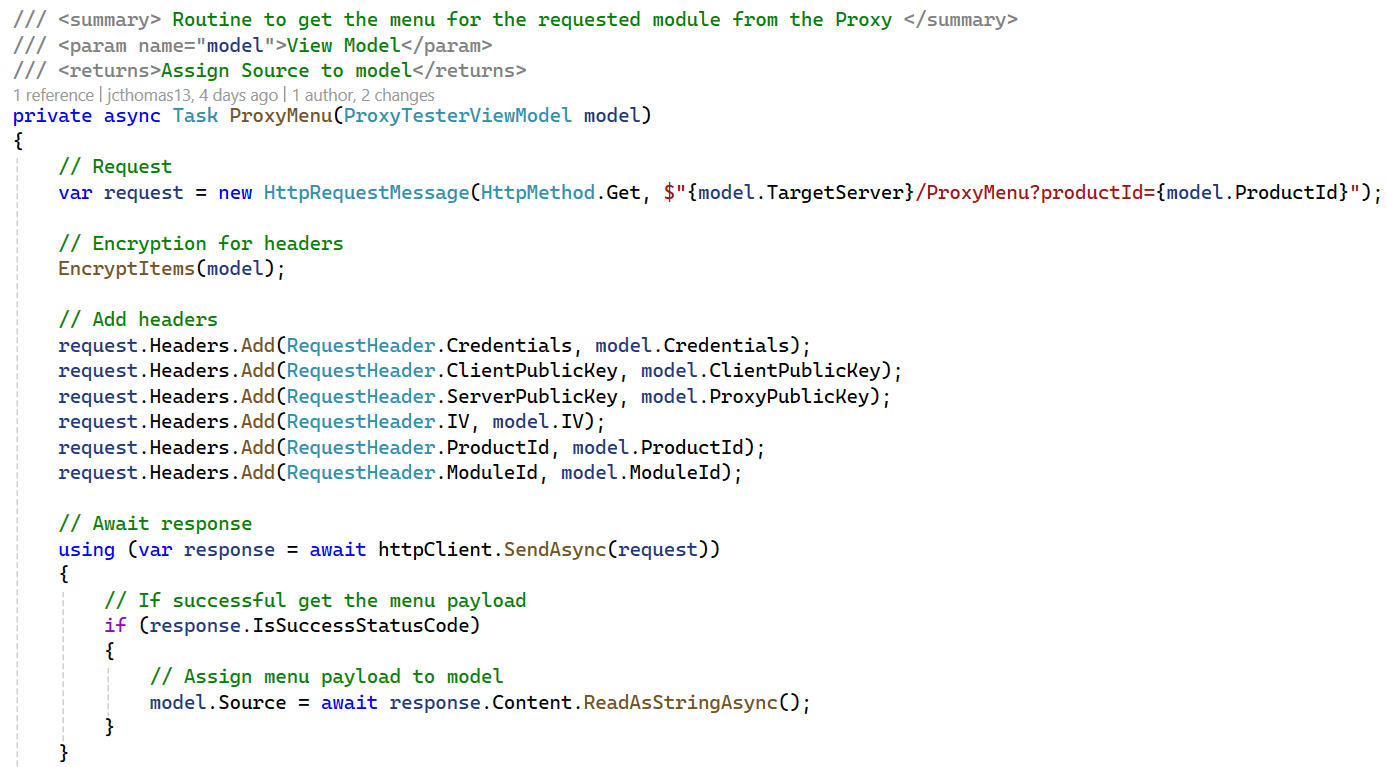


* + 1. Proxy Menu

The ***ProxyMenu*** method is the actual routine that invokes the Proxy ***ProxyMenu*** endpoint.

Before invocation, it must encrypt the headers and then add the headers to the request object.

Upon successful return, it stores the menu items payload and when the ajax handler is invoked, it will assign this to the iFrame’s ***src*** attribute.

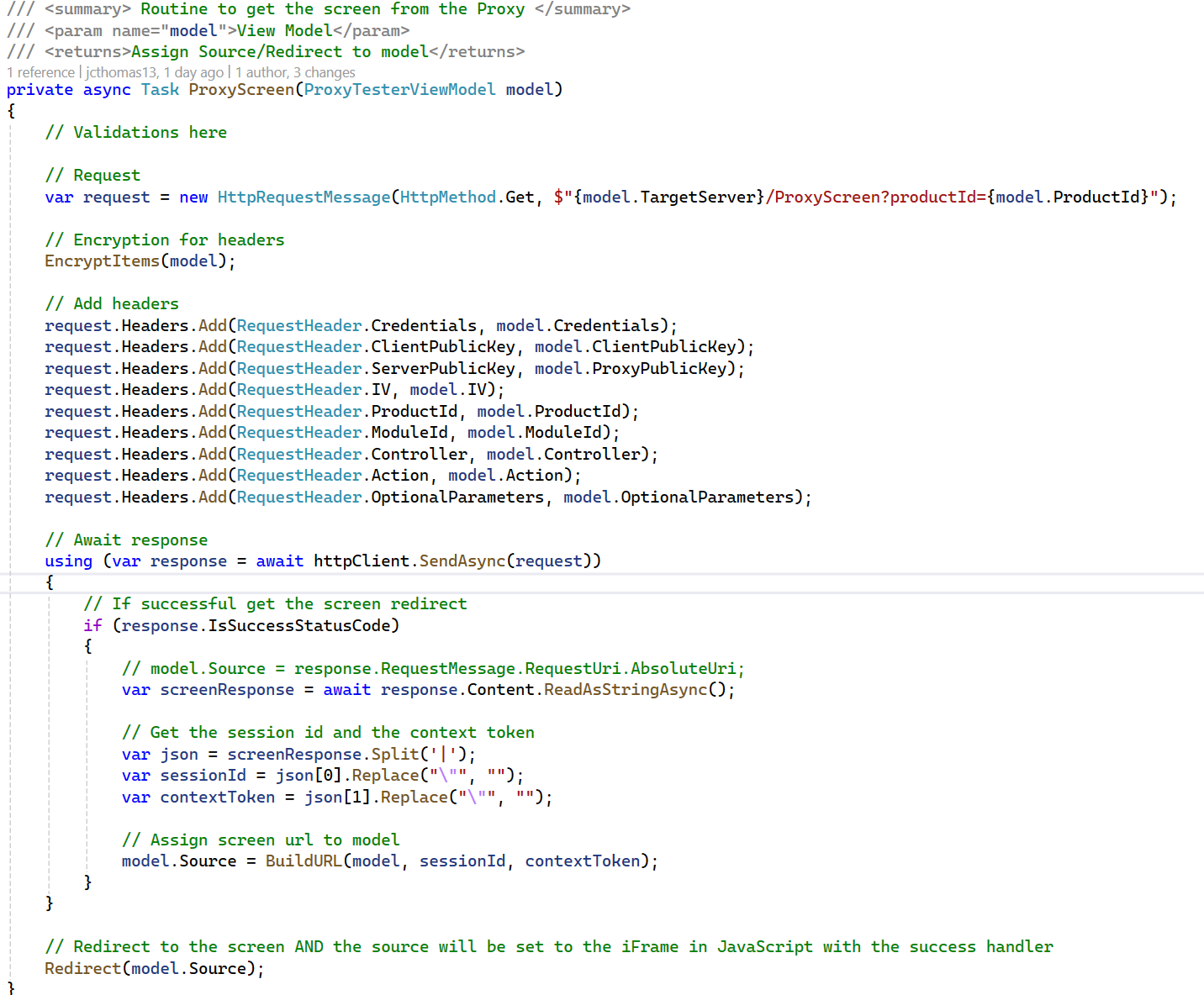


* + 1. Proxy Screen

The ***ProxyScreen*** method is the actual routine that invokes the Proxy ***ProxyScreen*** endpoint.

Before invocation, it must encrypt the headers and then add the headers to the request object.

Upon successful return, it splits the returned session id and token strings which are used to the build the Sage 300 screen url. It stores this url for when the ajax handler is invoked, it will assign this to the iFrame’s ***src*** attribute.



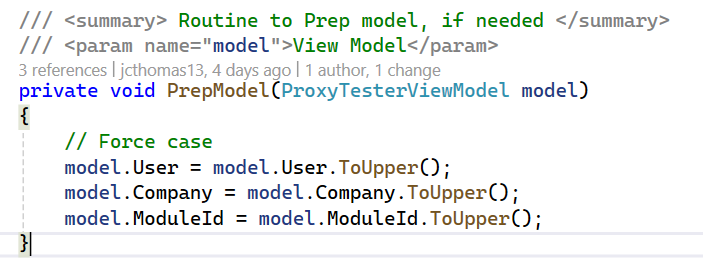
* + 1. Encrypt Items

The ***EncryptItems*** method is used to encrypt the items that will be added to the request header object. It uses the public key received from the Proxy to perform this encryption so that the server is the only one that can decrypt it.



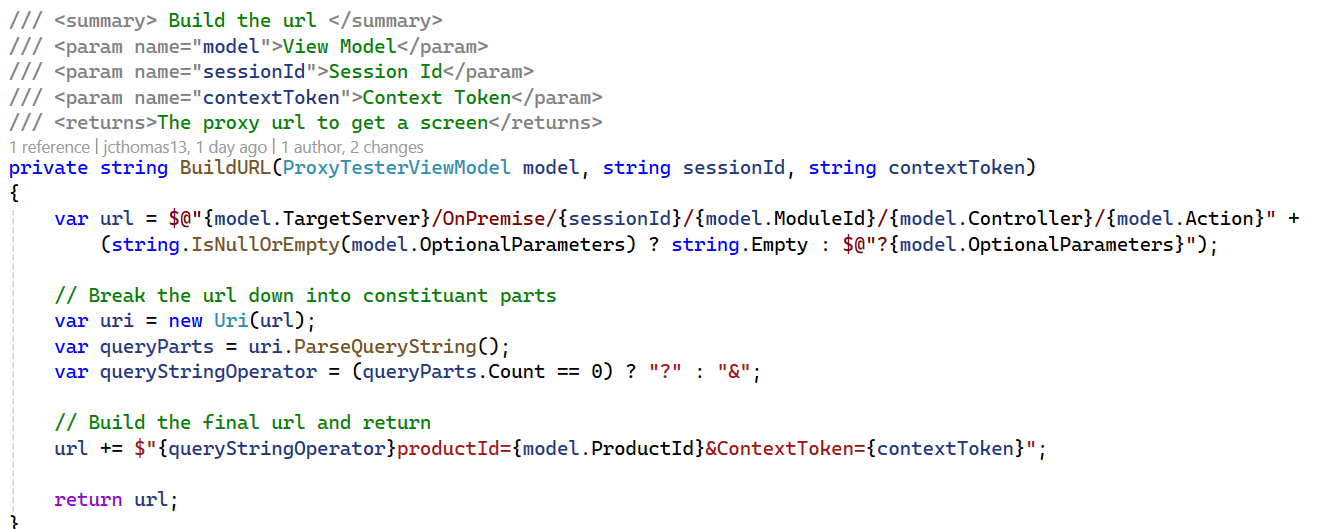
* + 1. Prep Model

The ***PrepModel*** method is used to perform any validations or preparations required.



* + 1. Build URL

The ***BuildURL*** method is used to build the URL for the requested web screen. It uses inputs from the tester screen and values returned from the Proxy to build the URL.



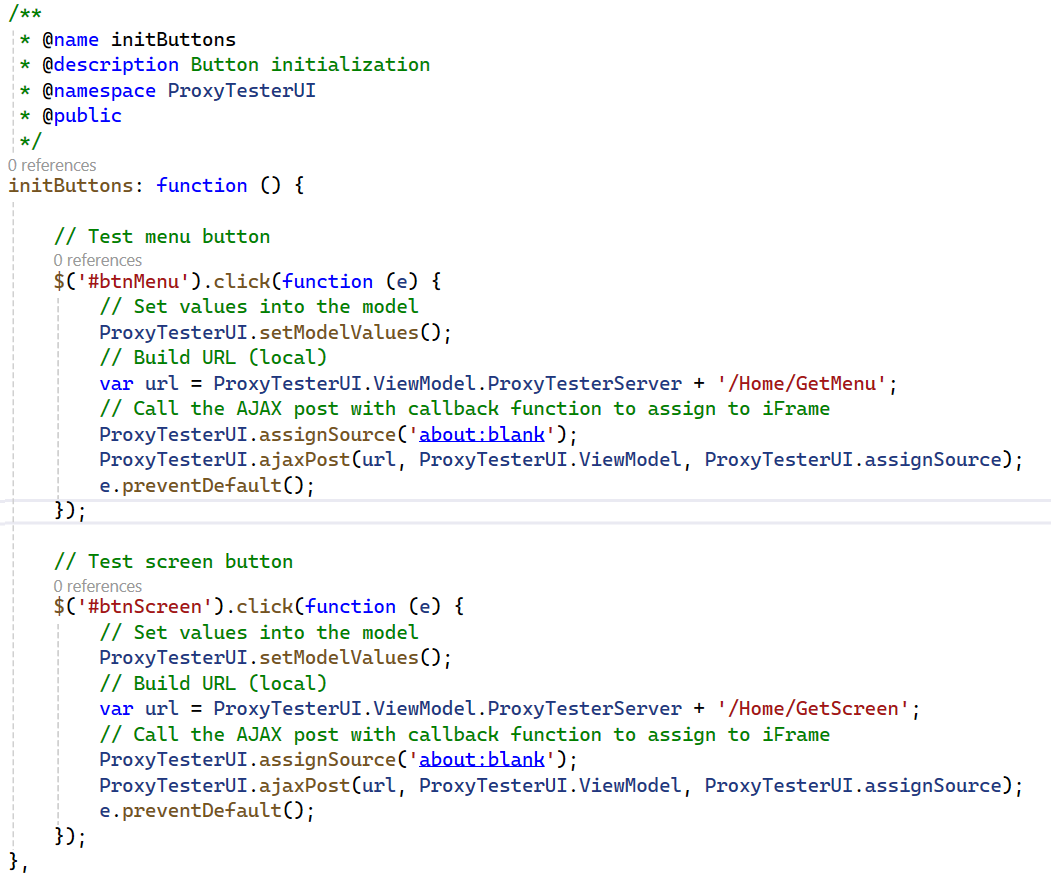
* 1. Proxy Tester Behavior JavaScript

This is the JavaScript file behind the tester screen.

* + 1. Init Buttons – Test Menu and Test Screen Clicks

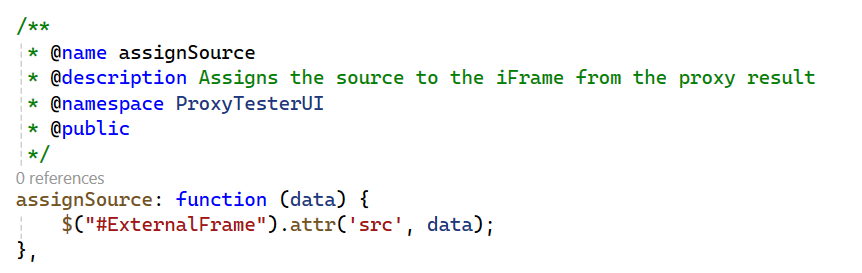
Upon clicking the selected button, an ajax call is made to the Home Controller for communicating with the Proxy.

A success handler will assign the source to the ***src*** attribute of the iFrame.



* + 1. Assign Source

Call back event to assign source from Proxy or constructed URL to be assigned to the ***src*** attribute of the iFrame.



* + 1. Ajax Post

Method to invoke the Home Controller with the requested action.

