Sage 300 Web Screens SDK

Proxy Tester Utility

September 2025

The MIT License (MIT)

Copyright © 2025 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](#_Toc207893849)

[2. What is the Proxy Tester 6](#_Toc207893850)

[3. Proxy Tester Prerequisites 8](#_Toc207893851)

[3.1 Required Properties to be changed 8](#_Toc207893852)

[3.1.1 Target Server 8](#_Toc207893853)

[3.1.2 Proxy Tester Server 8](#_Toc207893854)

[3.2 Optional Properties to be changed 9](#_Toc207893855)

[3.2.1 User 9](#_Toc207893856)

[3.2.2 Password 9](#_Toc207893857)

[3.2.3 Company 9](#_Toc207893858)

[3.2.4 ModuleId 9](#_Toc207893859)

[3.2.5 Controller 9](#_Toc207893860)

[3.2.6 Action 9](#_Toc207893861)

[3.2.7 OptionalParameters 9](#_Toc207893862)

[3.2.8 Source 10](#_Toc207893863)

[3.2.9 Id 10](#_Toc207893864)

[3.2.10 PDFFileDate 10](#_Toc207893865)

[3.2.11 PDFFileName 10](#_Toc207893866)

[4. Diagrams 11](#_Toc207893867)

[4.1 The Proxy 11](#_Toc207893868)

[4.1.1 ProxyPublicKey method 11](#_Toc207893869)

[4.1.2 ProxyMenu method 12](#_Toc207893870)

[4.1.3 ProxyScreen method 13](#_Toc207893871)

[4.1.4 ProxyPDFNames method 14](#_Toc207893872)

[4.1.5 ProxyPDF method 15](#_Toc207893873)

[4.1.6 ProxyDeletePDF method 16](#_Toc207893874)

[4.2 The Proxy Tester 17](#_Toc207893875)

[4.2.1 Testing a Menu 17](#_Toc207893876)

[4.2.2 Testing a screen 18](#_Toc207893877)

[4.2.3 Testing a list of PDF file names (HRMS) 19](#_Toc207893878)

[4.2.4 Testing a deletion of PDF files (HRMS) 20](#_Toc207893879)

[4.2.5 Testing a retrieval of a PDF file (HRMS) 21](#_Toc207893880)

[4.3 Elliptical Encryption 22](#_Toc207893881)

[4.3.1 ECDiffieHellmanCng 22](#_Toc207893882)

[4.3.2 Sample Workflow 23](#_Toc207893883)

[5. Running the Utility 24](#_Toc207893884)

[6. Proxy Tester Methods Explained 25](#_Toc207893885)

[6.1 Home Controller 25](#_Toc207893886)

[6.1.1 Get Menu 25](#_Toc207893887)

[6.1.2 Get Screen 26](#_Toc207893888)

[6.1.3 Get PDF Names 27](#_Toc207893889)

[6.1.4 Delete PDFs 28](#_Toc207893890)

[6.1.5 Get PDF 29](#_Toc207893891)

[6.1.6 Proxy Public Key 30](#_Toc207893892)

[6.1.7 Proxy Menu 31](#_Toc207893893)

[6.1.8 Proxy Screen 32](#_Toc207893894)

[6.1.9 Proxy PDF Names 33](#_Toc207893895)

[6.1.10 Proxy PDF 34](#_Toc207893896)

[6.1.11 Proxy Delete PDF 35](#_Toc207893897)

[6.1.12 Encrypt Items 36](#_Toc207893898)

[6.1.13 Prep Model 36](#_Toc207893899)

[6.1.14 Ping Server 37](#_Toc207893900)

[6.1.15 Build URL 38](#_Toc207893901)

[6.2 Proxy Tester Behavior JavaScript 39](#_Toc207893902)

[6.2.1 Init Buttons 39](#_Toc207893903)

[6.2.2 Assign Source 41](#_Toc207893904)

[6.2.3 Display Alert 41](#_Toc207893905)

[6.2.4 Assign PDF Names 41](#_Toc207893906)

[6.2.5 Assign PDF 42](#_Toc207893907)

[6.2.6 Delete PDF 42](#_Toc207893908)

[6.2.7 Ajax Post 42](#_Toc207893909)

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 the consumer is allowed to:

* Display a Sage 300 Web Screen in their web application.
* Receive a JSON payload of a module’s menu items (HRMS Integration)
* Receive and Delete Employee PDFs (HRMS Integration)

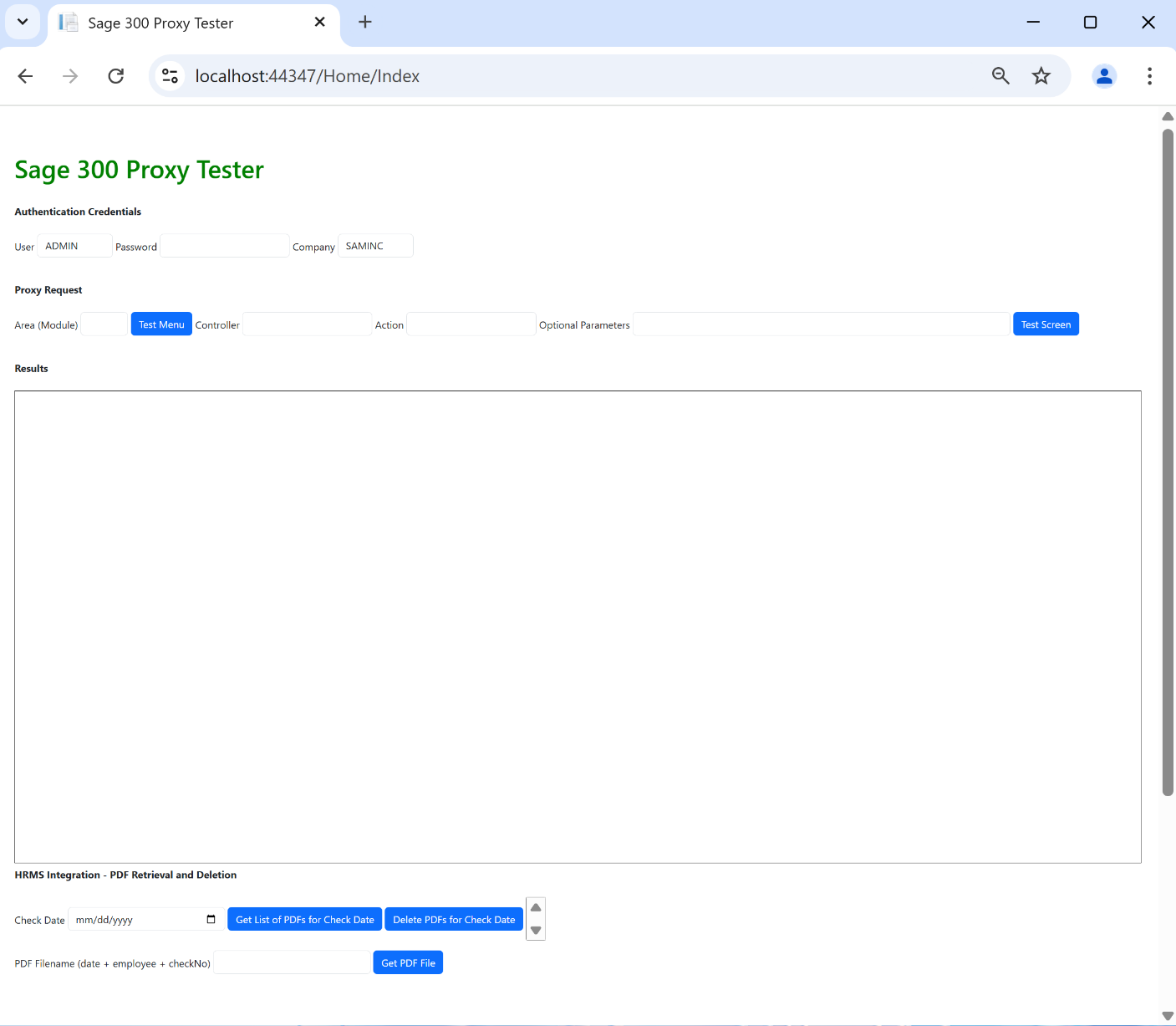
The Proxy is a process that facilitates the encryption and decryption of credentials, logs into the application, and allows for the above actions.

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.

The Sage CRM Integration will be modified to use this Proxy for displaying Sage 300 Web Screens in Sage CRM’s web application.

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).

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

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

Additionally, for our HRMS Integration, the proxy provides the ability to retrieve and delete PDF files for split checks.

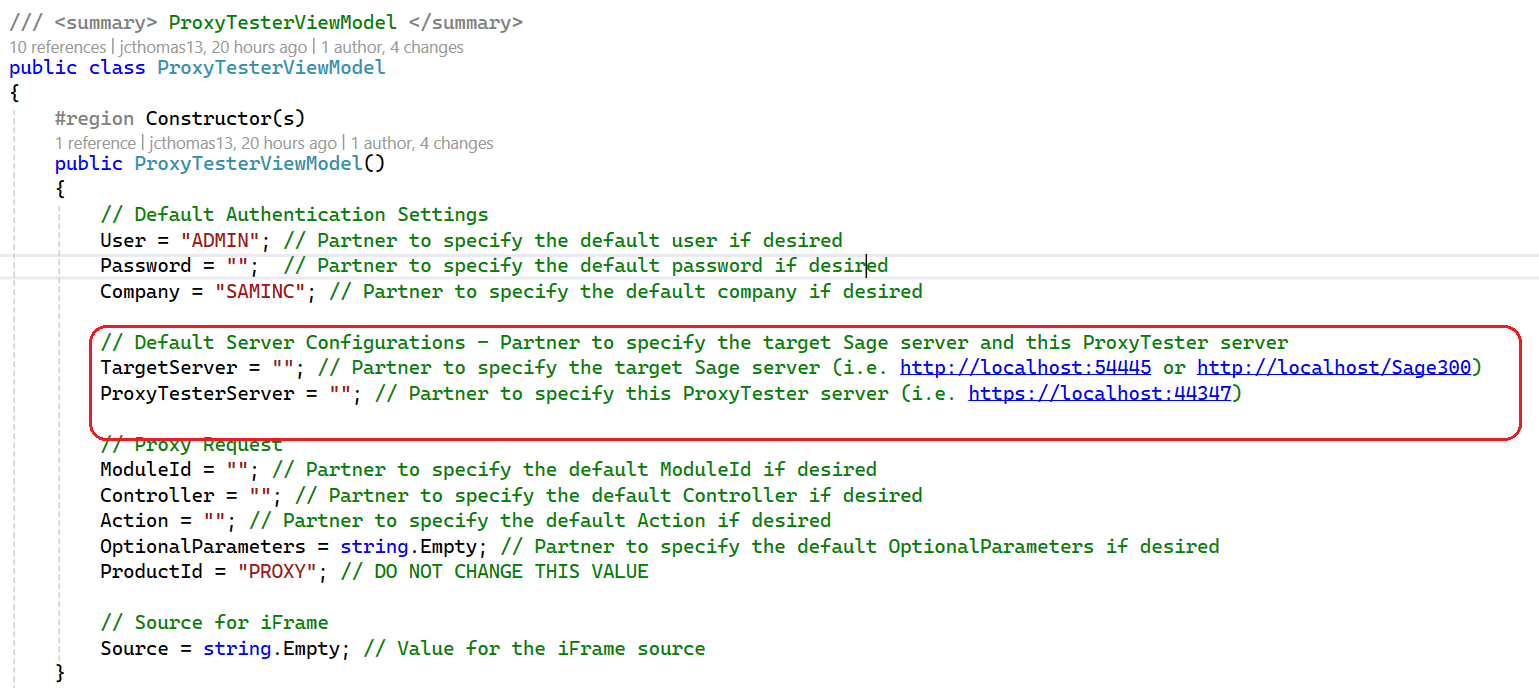
1. Proxy Tester Prerequisites

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 application. 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. Id

The Id representing a unique id for the proxy session request. Defaulted to a new Guid.

Do not change this value as it assigned a new guid upon startup of the session

Id = Guid.NewGuid().ToString();

* + 1. PDFFileDate

The check date representing the check date for PDF files. Defaulted to “”. (HRMS)

PDFFileDate = string.Empty;

* + 1. PDFFileName

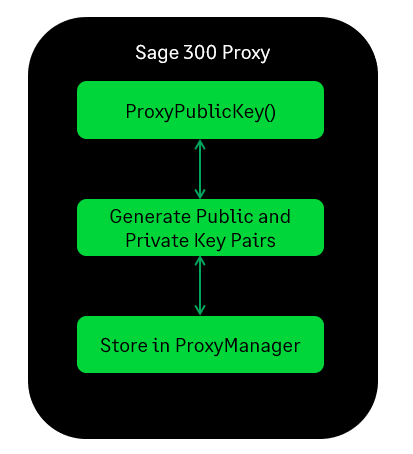
The file name representing a PDF file (check date + employee + checkNo). Defaulted to “”. (HRMS)

PDFFileName = 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 proxy request (screen, menu, etc.) as headers are encrypted every time with a new public key that is only valid for a single use.



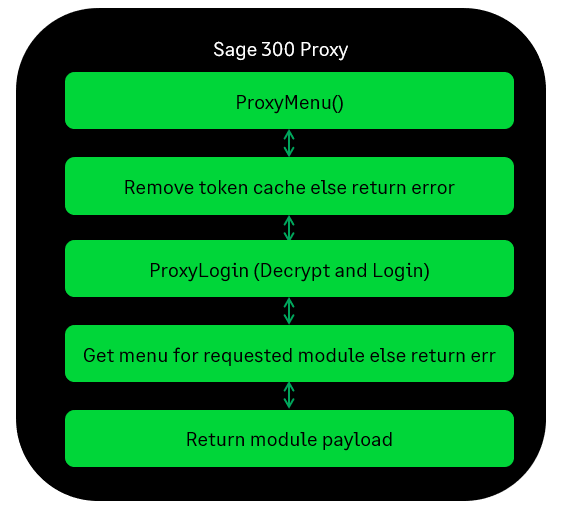
* + 1. ProxyMenu method

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, also has a limited lifecycle of 5 minutes once created.



* + 1. ProxyScreen method

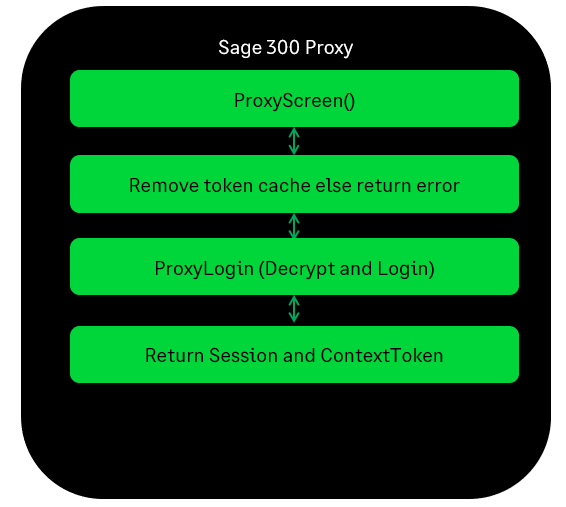
Returns a session id and ContextToken to be used for the requested screen.

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, also has a limited lifecycle of 5 minutes once created.

The consumer is responsible for building the URL and setting the **src** attribute in the consumers iFrame.



* + 1. ProxyPDFNames method

Returns a list PDF file names for a requested check date.

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, also has a limited lifecycle of 5 minutes once created.

If the user does not have access to the Employee Checks process, the request fails.

A screenshot of a computer program

AI-generated content may be incorrect.

* + 1. ProxyPDF method

Returns a PDF file for a requested check date, employee, and check number.

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, also has a limited lifecycle of 5 minutes once created.

If the user does not have access to the Employee Checks process, the request fails.

A screenshot of a computer program

AI-generated content may be incorrect.

* + 1. ProxyDeletePDF method

Deletes all PDF files for a requested check date.

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, also has a limited lifecycle of 5 minutes once created.

If the user does not have access to the Employee Checks process, the request fails.

A screenshot of a computer program

AI-generated content may be incorrect.

* 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.

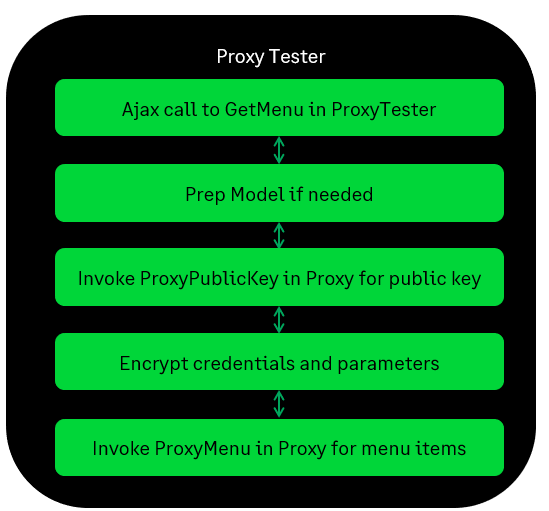
The model is prepped as necessary (i.e. upper casing values, etc.)

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.

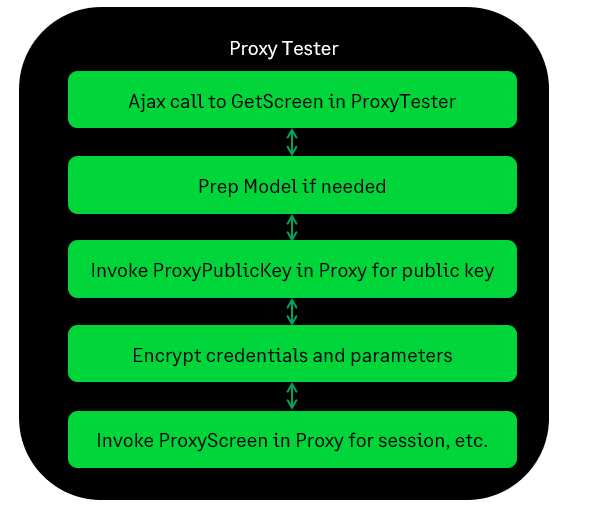
The model is prepped as necessary (i.e. upper casing values, etc.)

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. Testing a list of PDF file names (HRMS)

The tester has a ***Get List of PDFs for Check Date*** button for testing the Proxy’s ***ProxyPDFNames*** gateway for requesting a list of PDF file names to be retrieved.

An ajax call is made to the tester for requesting the list of files 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 ***ProxyPDFNames*** method is invoked to request the list of PDF files for a specific check date to be retrieved.

A screenshot of a computer

AI-generated content may be incorrect.

* + 1. Testing a deletion of PDF files (HRMS)

The tester has a ***Delete PDFs for Check Date*** button for testing the Proxy’s ***ProxyDeletePDFs*** gateway for deleting PDF files.

An ajax call is made to the tester for deleting PDF files 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 ***ProxyDeletePDFs*** method is invoked to delete the PDF files for a specific check date.

A black and green screen with green text

AI-generated content may be incorrect.

* + 1. Testing a retrieval of a PDF file (HRMS)

The tester has a ***Get PDF File*** button for testing the Proxy’s ***ProxyPDF*** gateway for retrieving a PDF file.

An ajax call is made to the tester for deleting PDF files 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 ***ProxyPDF*** method is invoked to retrieve a PDF file for a specific check date.

A screenshot of a computer

AI-generated content may be incorrect.

* 1. Elliptical Encryption

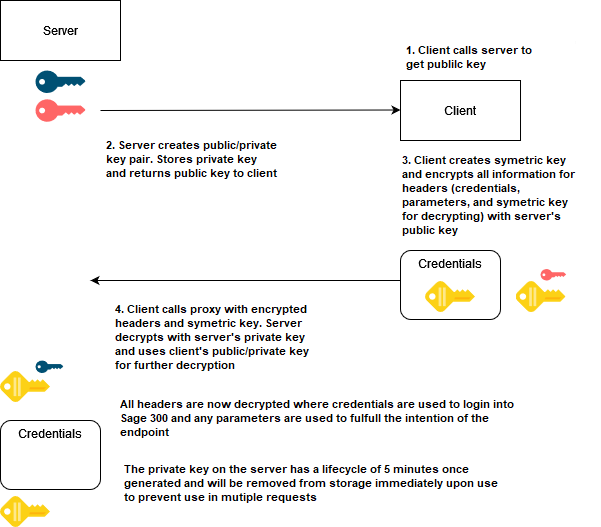
Elliptic-Curve Cryptography (ECC) is an algorithm for asymmetric encryption.

* + 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

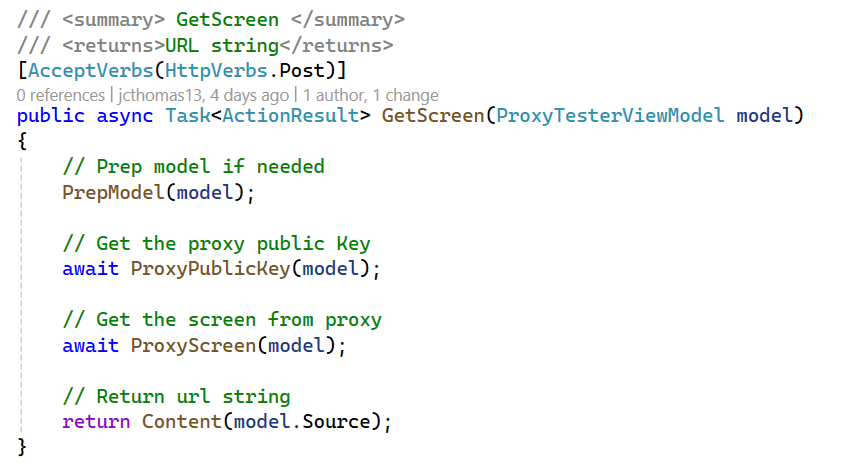
* + 1. Get Menu

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



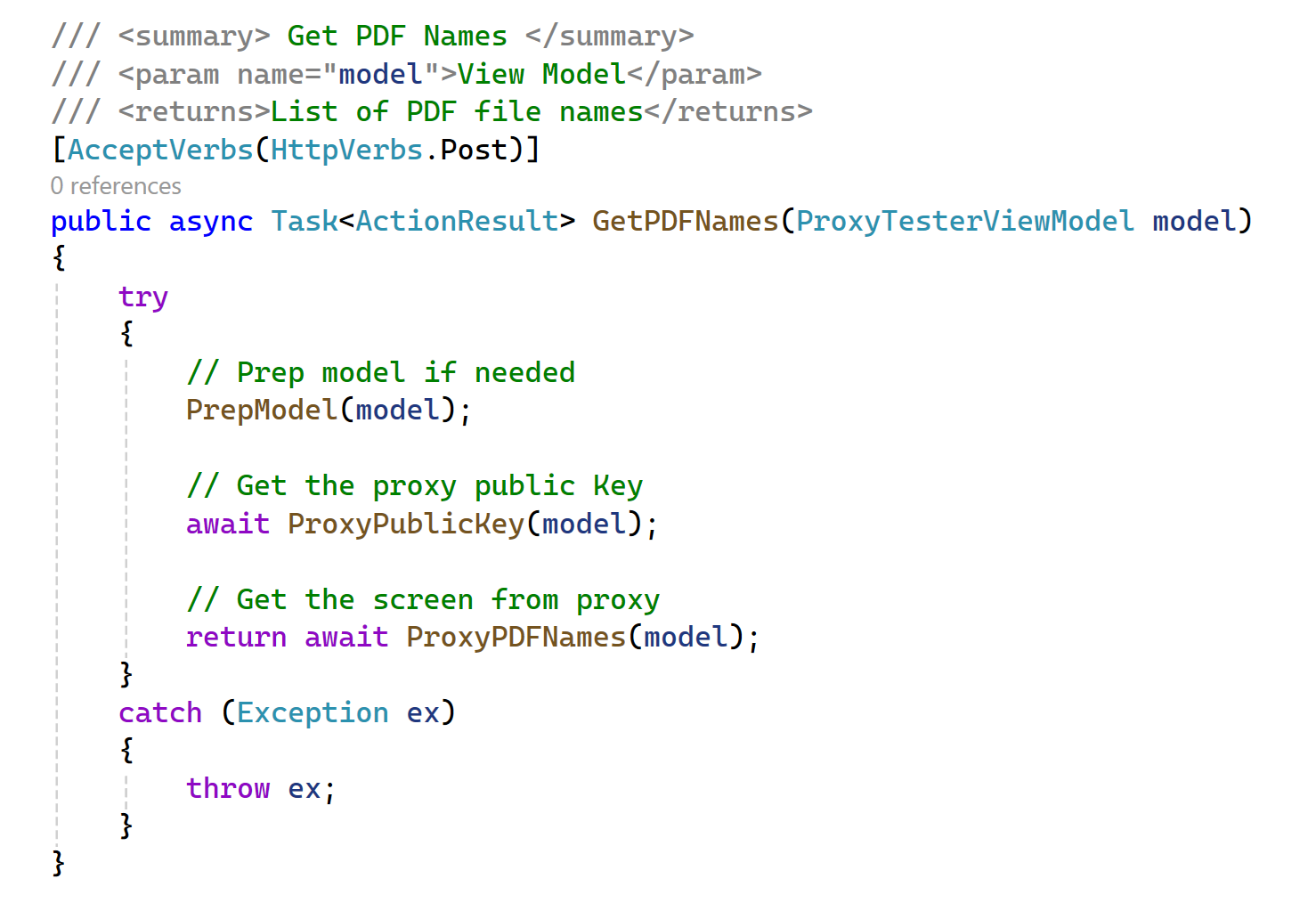
* + 1. Get Screen

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



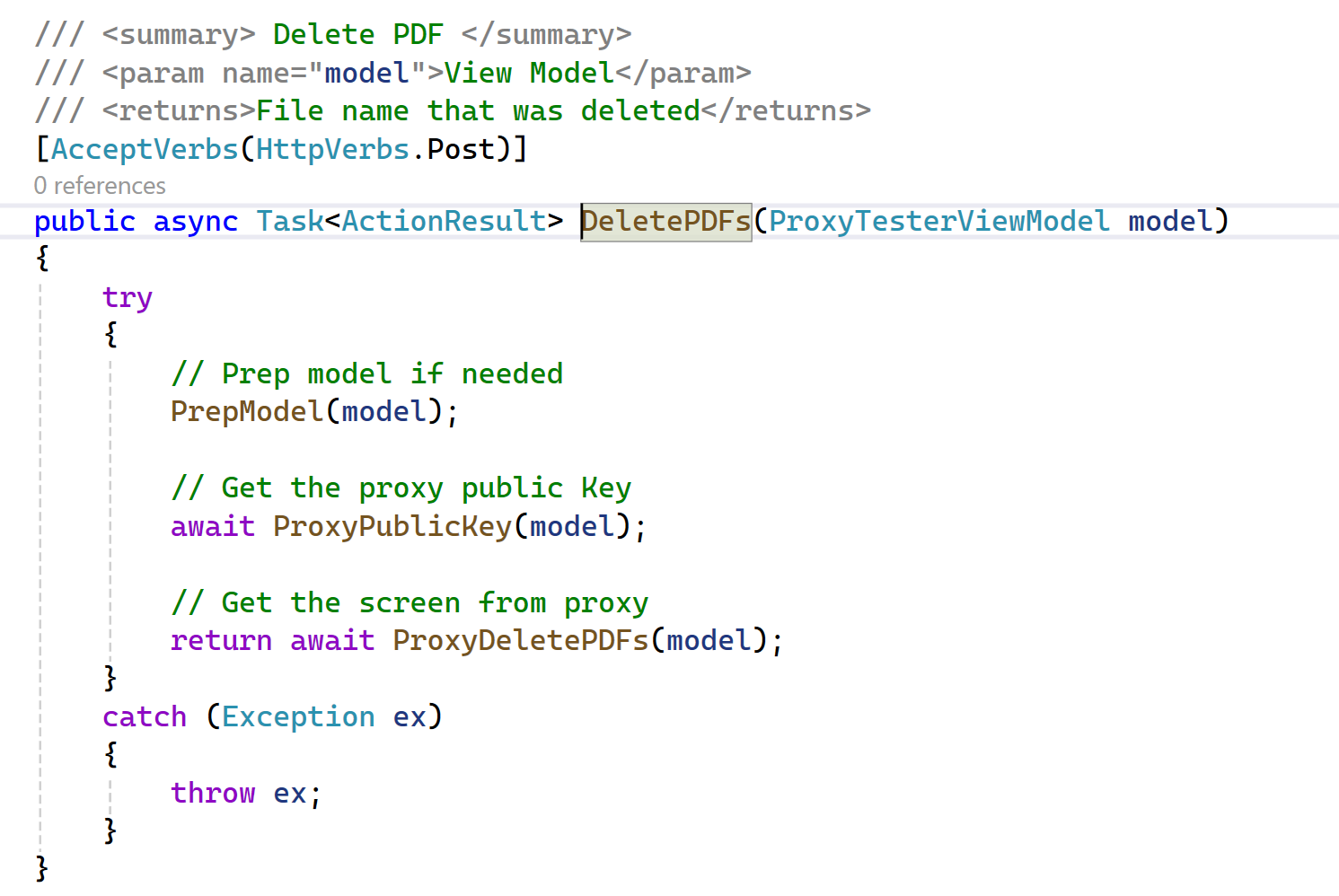
* + 1. Get PDF Names

The ***GetPDFNames*** method is invoked with the ***Get List of PDFs for Check Date*** button is pressed with the model properties from the tester screen



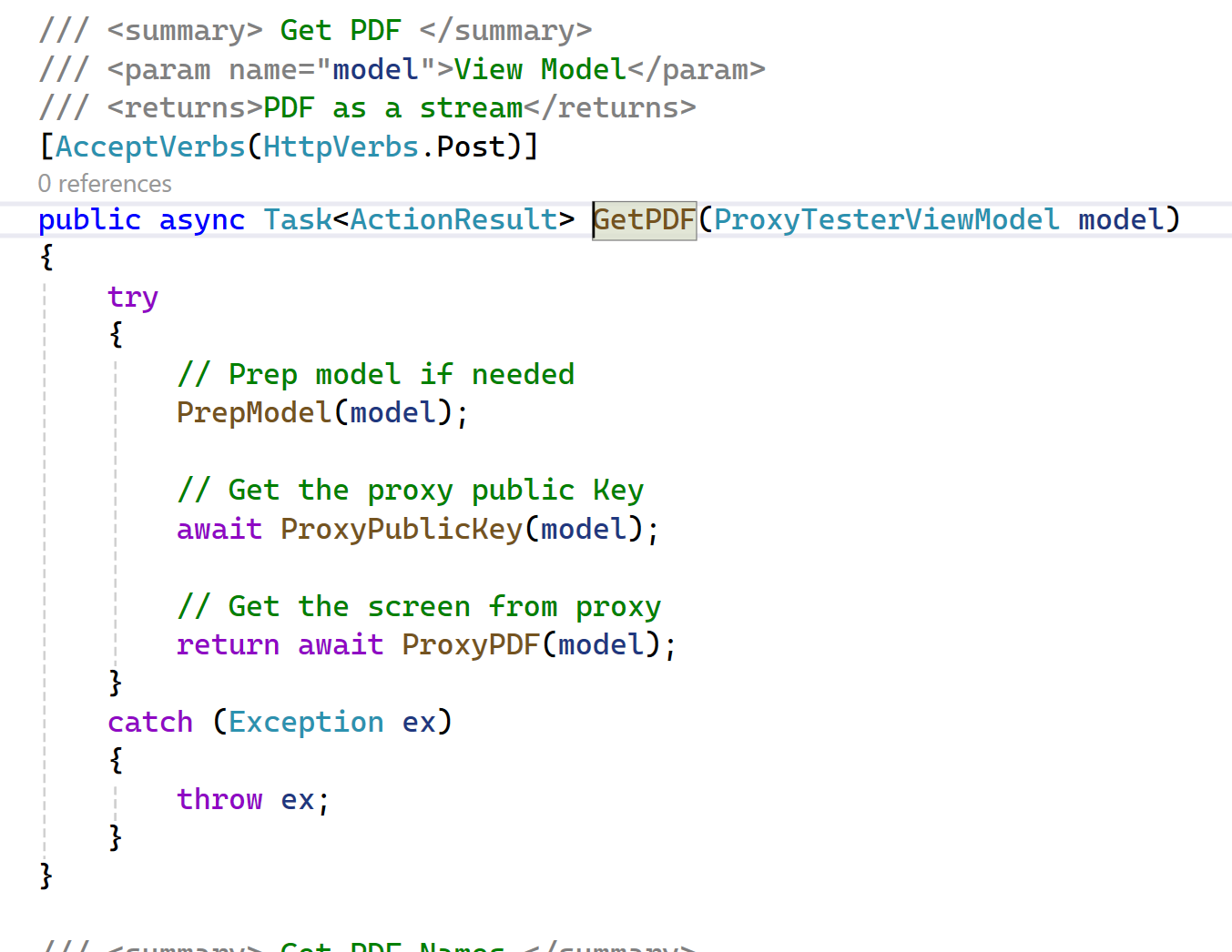
* + 1. Delete PDFs

The ***DeletePDFs*** method is invoked with the ***Delete PDFs for Check Date*** button is pressed with the model properties from the tester screen



* + 1. Get PDF

The ***GetPDF*** method is invoked with the ***Get PDF File*** 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.

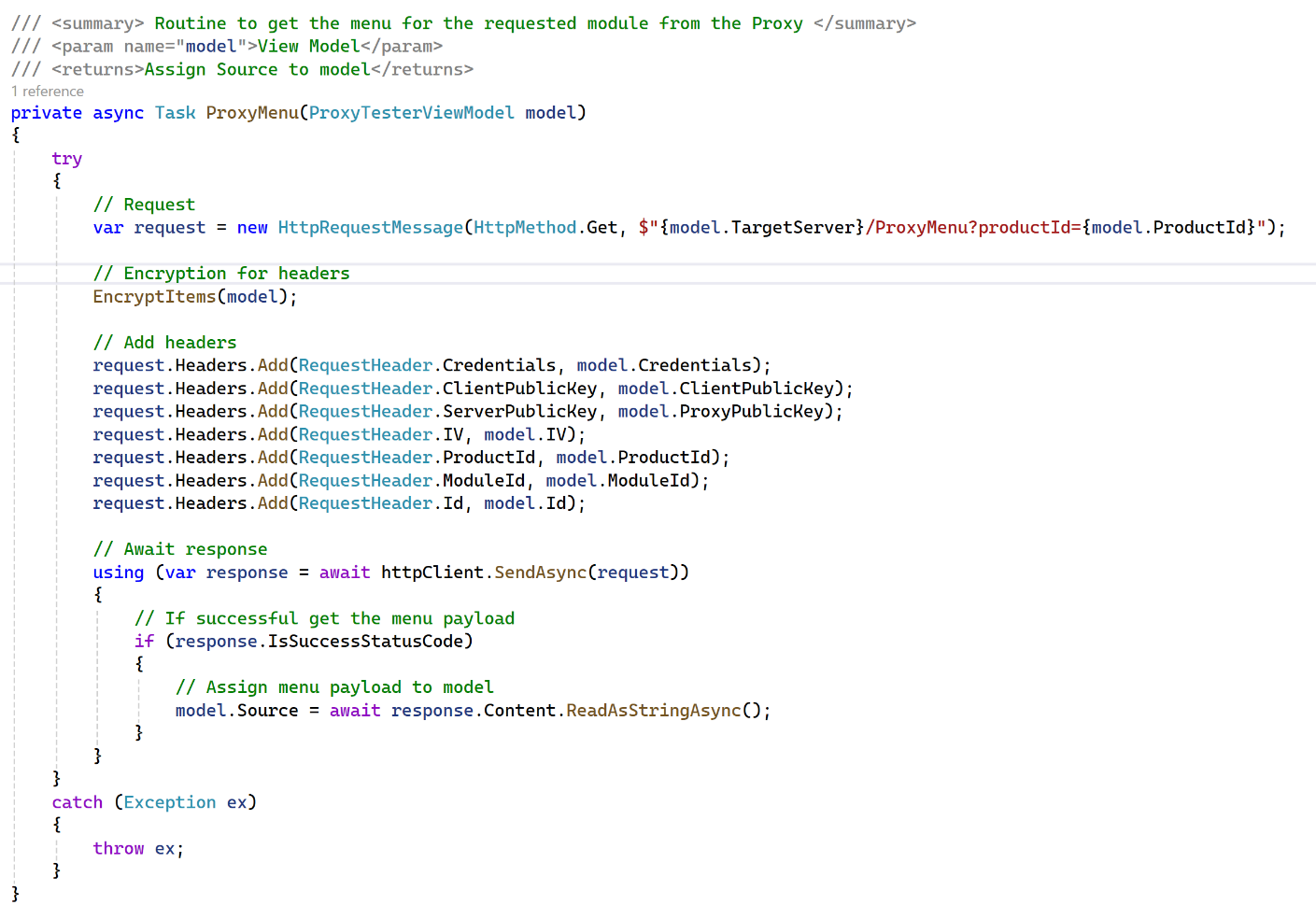


* + 1. Proxy Menu

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 display the payload in an alert dialog.

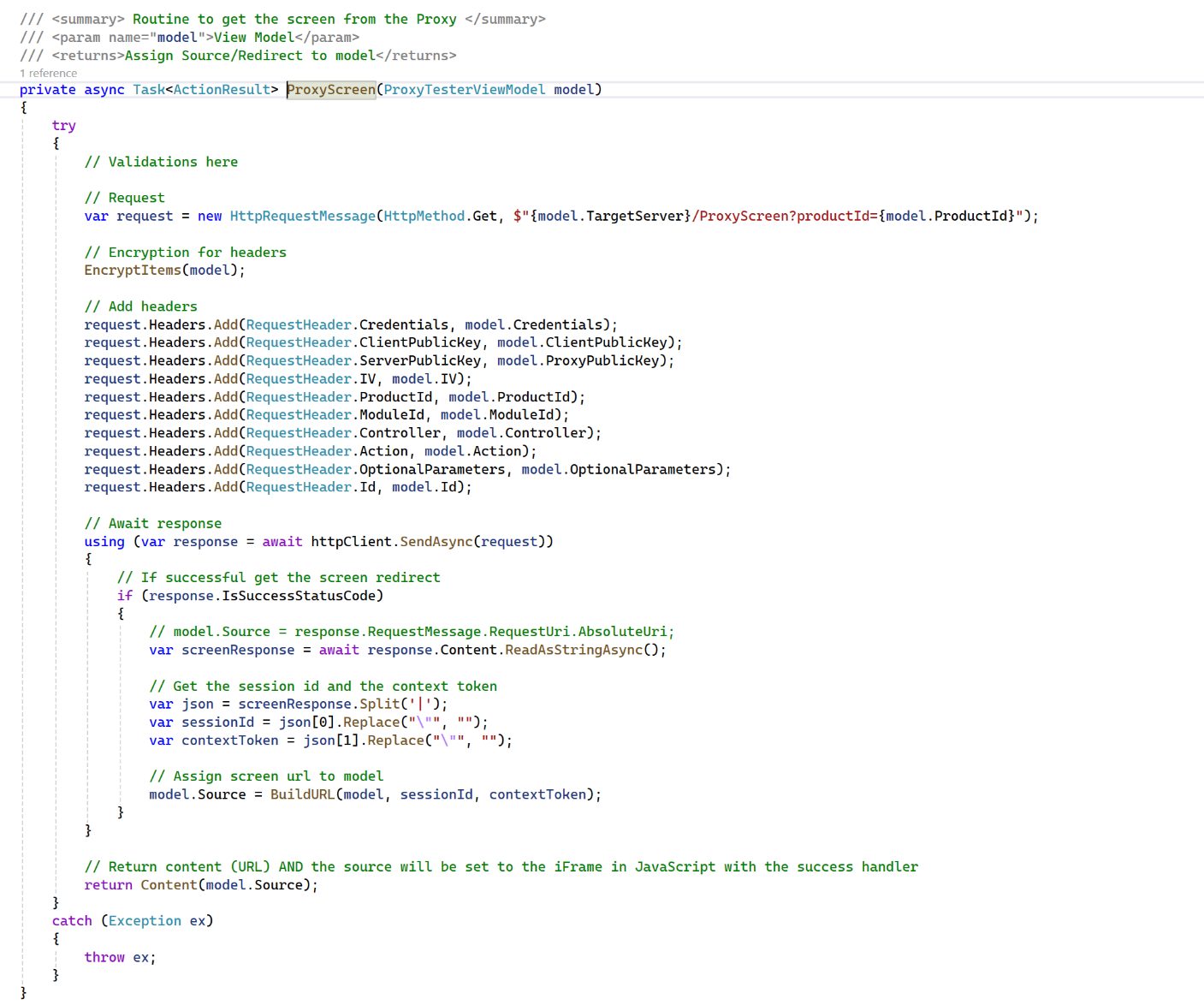


* + 1. Proxy Screen

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 iFrames ***src*** attribute.

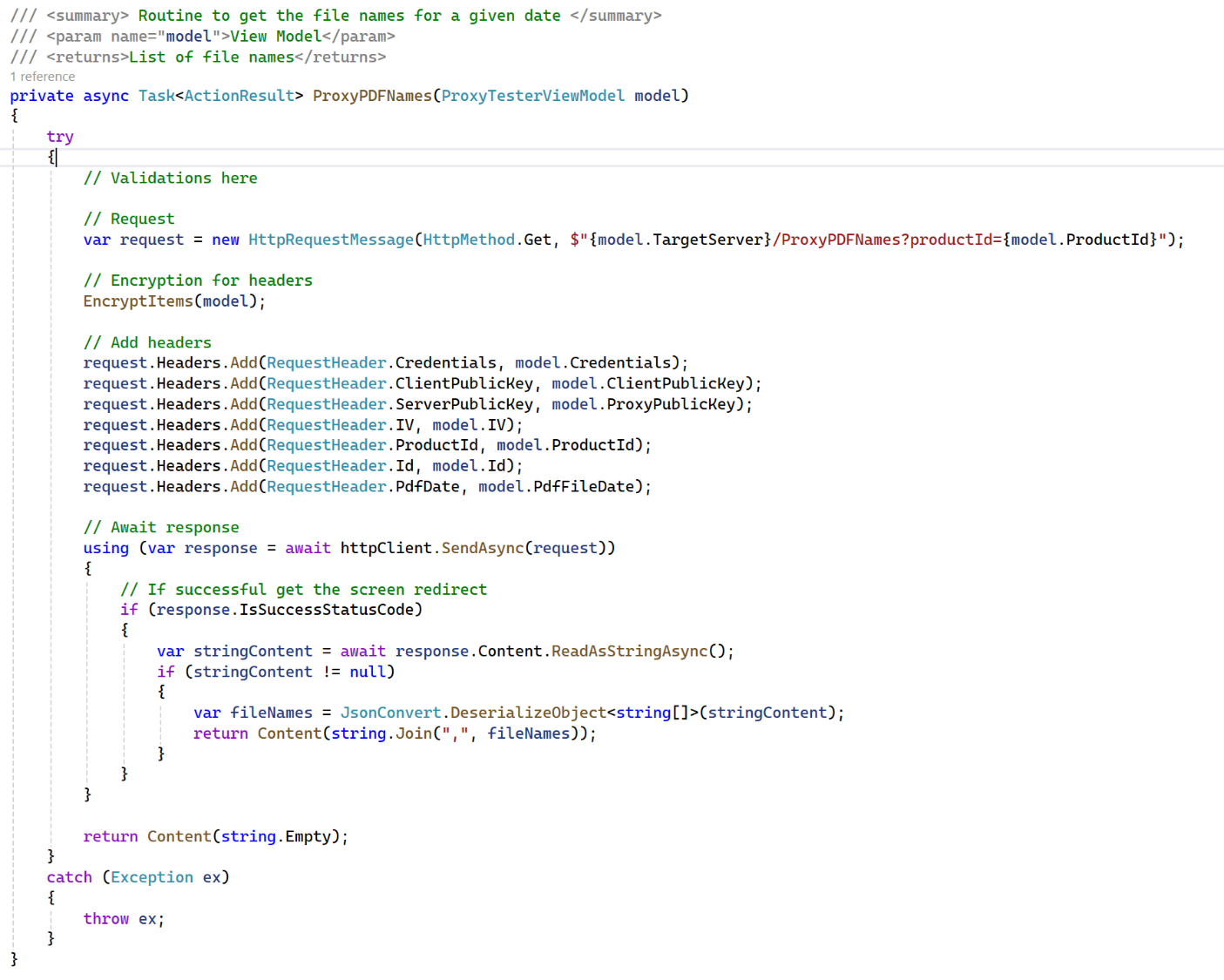


* + 1. Proxy PDF Names

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

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

Upon successful return, it stores the PDF file names in a drop-down list for display.



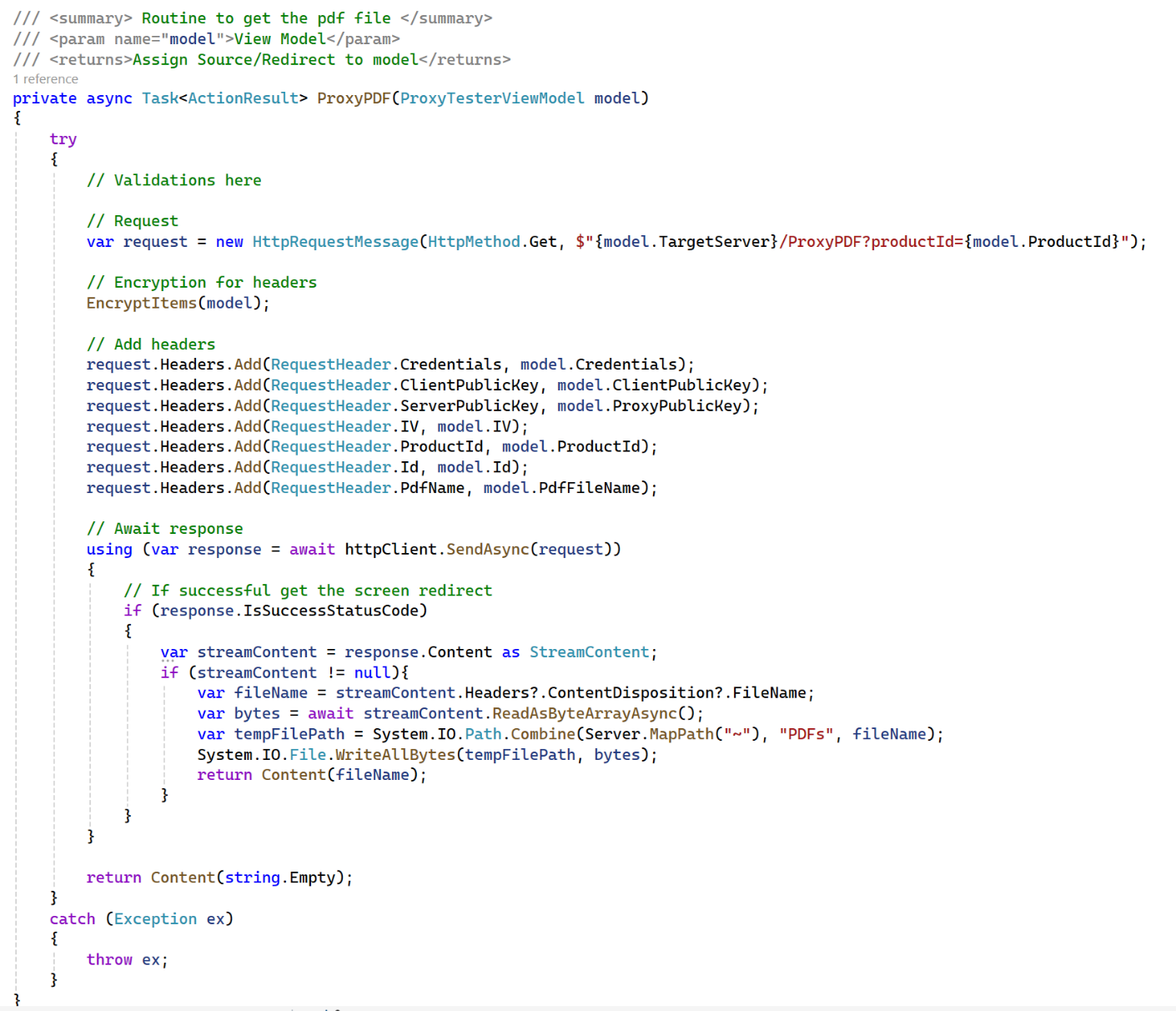
* + 1. Proxy PDF

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

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

Upon successful return, it displays the requested PDF for viewing and confirmation that the requested PDF was retrieved.

It should be noted that the return from the Proxy is a stream of the PDF file whereas the this tester shows the contents for mere confirmation.

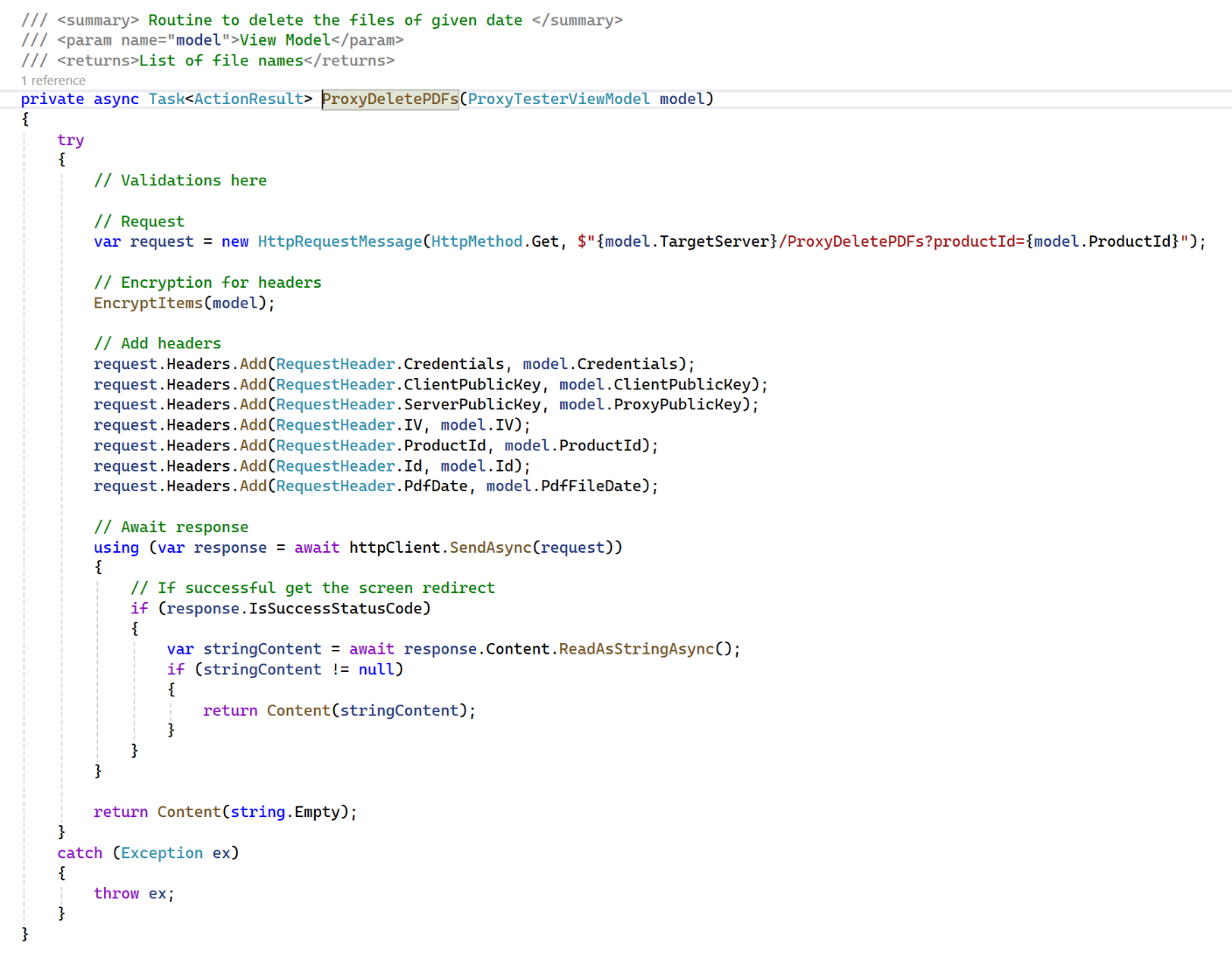


* + 1. Proxy Delete PDF

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

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

Upon successful return, it displays the requested PDF file name for deletion in an alert dialog to confirm that the file was deleted.



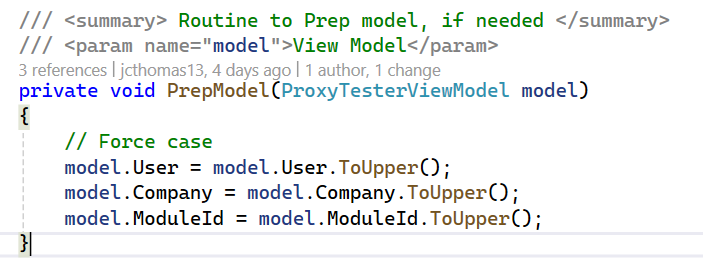
* + 1. Encrypt Items

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

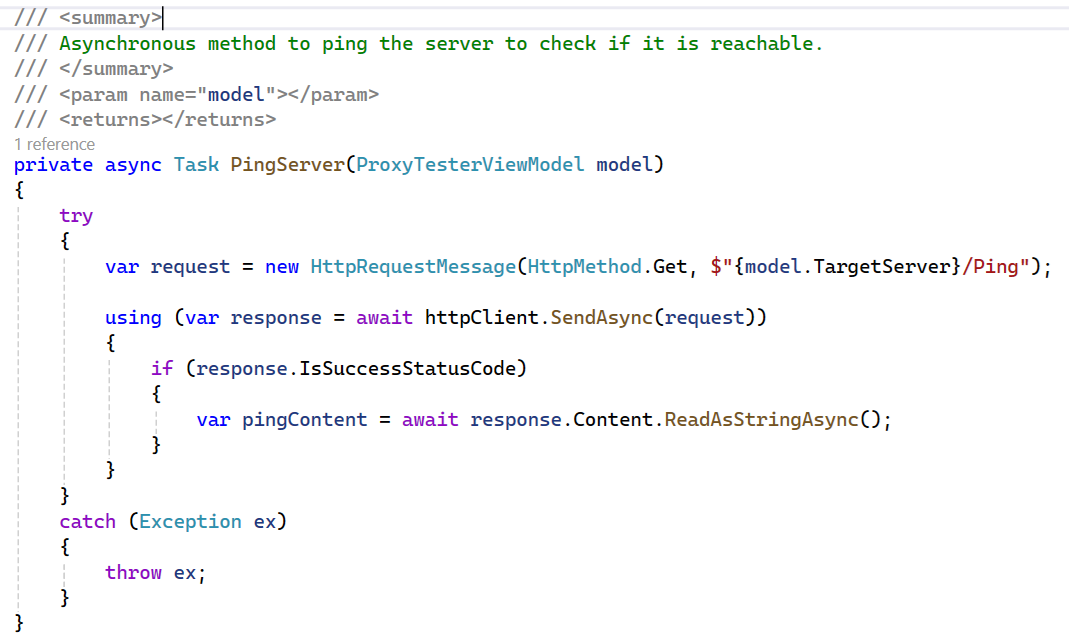
The ***PrepModel*** method is used to perform any validations or preparations required.



* + 1. Ping Server

The ***PingServer*** method is used to perform a “ping” to the web server to wake it up if it is asleep or has not started.

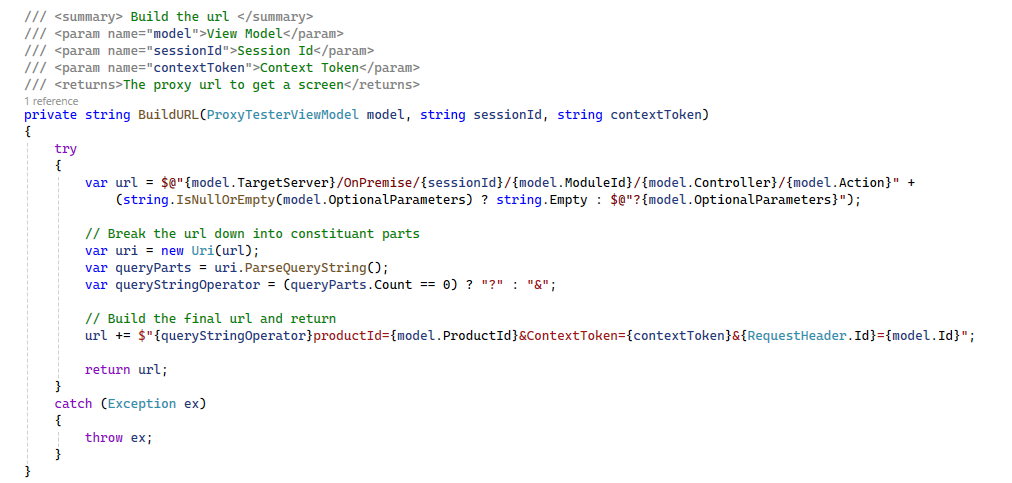
This is simply a performance enhancement since if the server is asleep or has not started for the day, there could be a 30-45 second delay in making the first request.



* + 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.

The Id has been added to the URL as a security prevention. If this URL were to be run outside of this process, the Id would not be valid even though this same URL works via the proxy.



* 1. Proxy Tester Behavior JavaScript

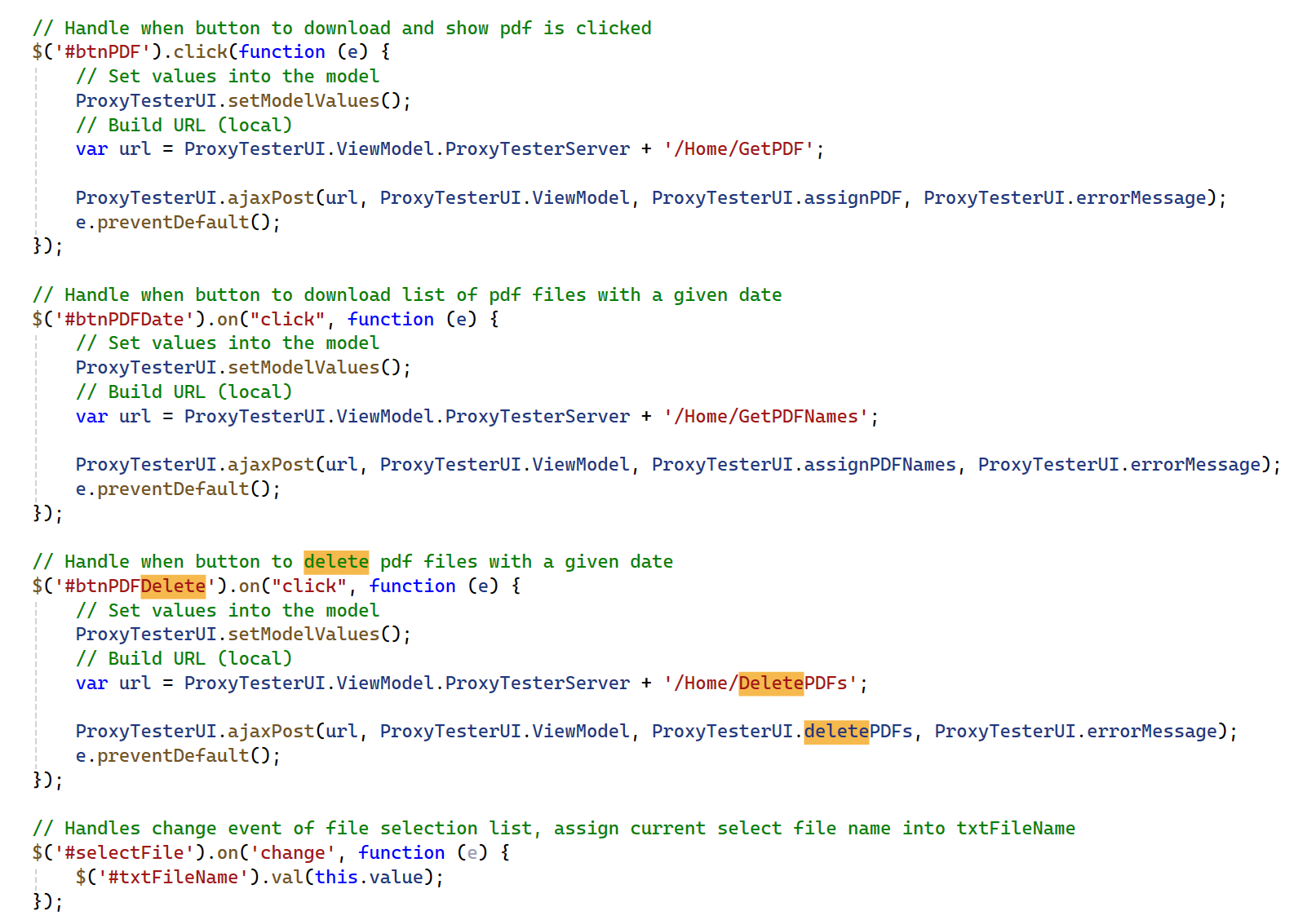
This is the JavaScript file behind the tester screen.

* + 1. Init Buttons

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

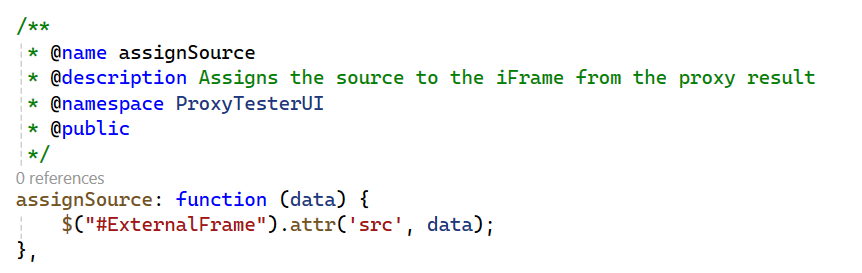
Success handlers will perform various actions.





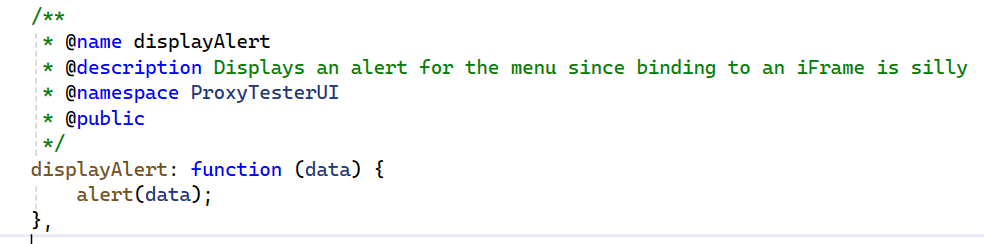
* + 1. Assign Source

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



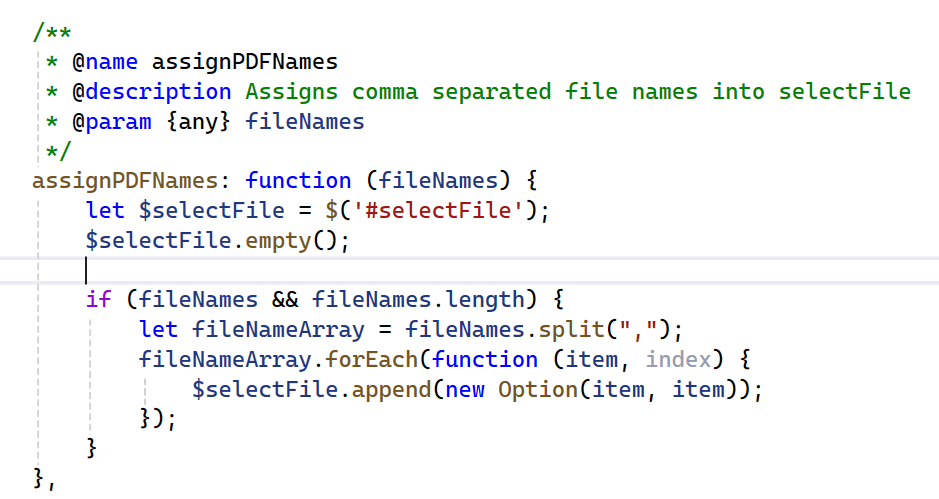
* + 1. Display Alert

Call back event to display payload from Proxy’s menu call in an alert dialog.



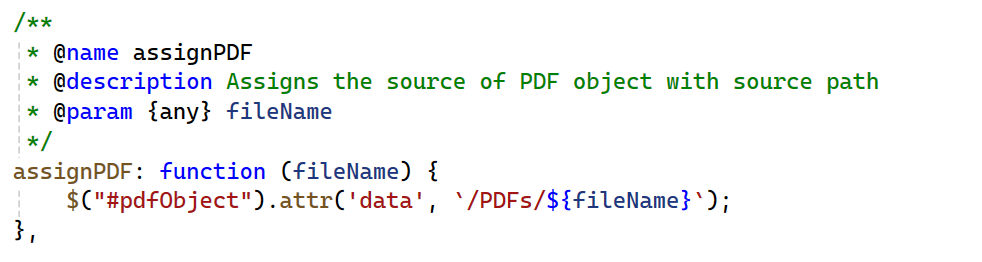
* + 1. Assign PDF Names

Call back event to assign list of PDF file names from Proxy to be assigned to the drop-box.



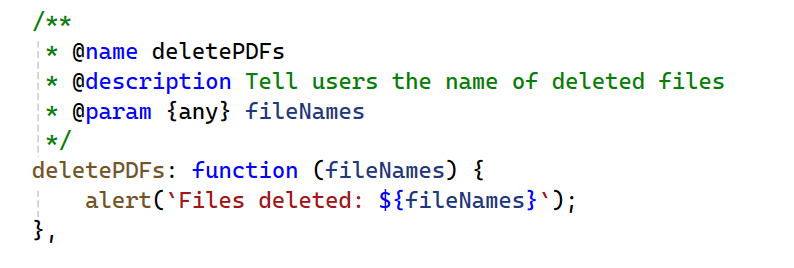
* + 1. Assign PDF

Call back event to display the PDF file retrieved from the Proxy.



* + 1. Delete PDF

Call back event to display the deleted PDF file in an alert dialog.



* + 1. Ajax Post

Method to invoke the Home Controller with the requested action.

