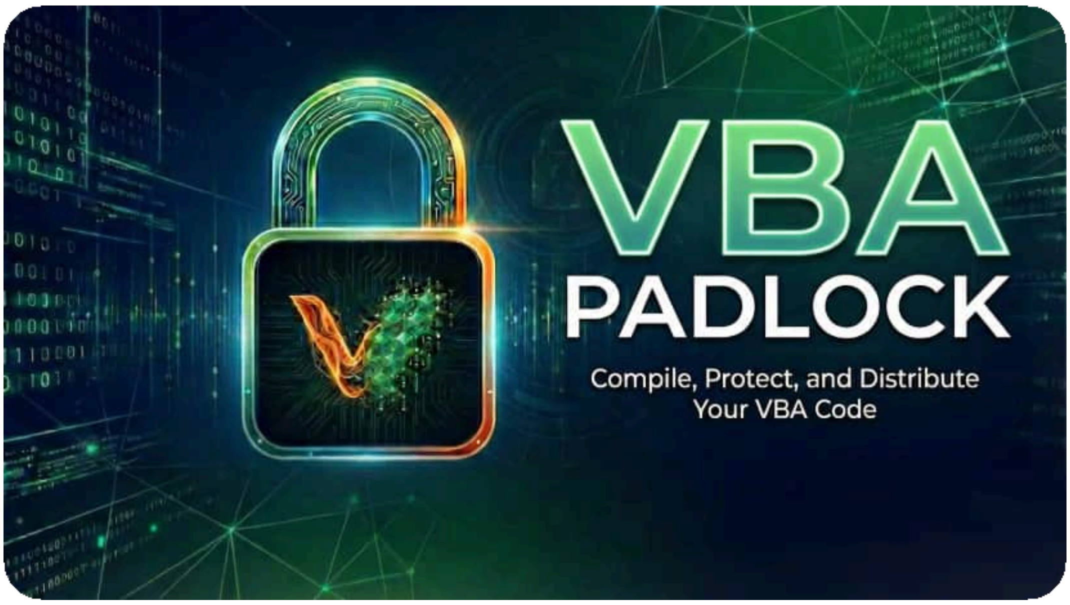


VBA Padlock

User Manual

Version 2026

G.D.G. Software

www.vbapadlock.com

Table of Contents

1. Professional VBA Protection

Transform your vulnerable Office macros into secure, compiled, and hardware-locked DLLs.

VBA Padlock is the most advanced **VBA Compiler and Licensing System** for Microsoft Office. It bridges the gap between simple Excel macros and professional software distribution by converting your code into compiled machine bytecode.

1.1 Why VBA Padlock?

Unbreakable Security

Your code is removed from the Office file and compiled into a secure DLL. No more passwords to crack.

Monetize Your Work

Built-in Licensing, Trial periods, and Online Activation to turn your macros into a SaaS business.

Native Performance

Runs on our proprietary high-performance virtual machine, compatible with both 32-bit and 64-bit Office.

Professional Distribution

Create installers, manage versions, and distribute your Office solutions as a complete software package.

1.2 Jump Right In

Quick Start

Protect your first Excel, Word, or Access project in less than 5 minutes.

Licensing Mode

Learn how to add serial keys, trial periods, and hardware locking.

Hardware Locking

Understand how to bind your software to a specific computer.

Online Activation

Automate your sales with a PHP-based activation server.

1.3 Industry Compatibility

VBA Padlock works wherever VBA works. We support:

- **Microsoft Excel** (.xlsm, .xlam, .xlsb)
- **Microsoft Word** (.docm, .dotm)
- **Microsoft PowerPoint** (.pptm, .ppam)
- **Microsoft Access** (.accdb)
- **32-bit and 64-bit** architectures from Office 2016 to Office 2024 and **Microsoft 365 desktop apps**.

Windows ARM Support

Native Office ARM is not supported yet. However, on **Windows ARM** devices (like **Surface Pro 11, Surface Laptop 7, or Snapdragon X Elite** powered laptops), we recommend installing **Office 64-bit**. It runs seamlessly via Microsoft's **Prism** emulation, allowing VBA Padlock's 64-bit DLLs to function perfectly.

2. Installation Guide

VBA Padlock is a lightweight, high-performance Windows application. It does not require any external runtimes like .NET, Java, or Python, making it one of the most stable compilers for the Office ecosystem.

2.1 System Requirements

Ensure your development machine meets these minimum specifications:

Operating System

Windows 7 SP1 or later (64-bit recommended)

Microsoft Office

Office 2016, 2019, 2021, 2024, or Microsoft 365 desktop apps (32-bit or 64-bit)

Disk Space

Minimum 100 MB free space for installation.

VBA Access

“Trust access to the VBA project object model” must be enabled in Office.

2.2 Installation Steps

1. **Download the Installer**: Fetch the latest version of the VBA Padlock setup from our Download Page.

2. **Run the Setup**: Double-click the `.exe` file.

   > **Note**
   >
   > If Windows SmartScreen appears, click **More info** → **Run anyway**. All our installers are digitally signed by **G.D.G. Software** for your security.

3. **Choose Location**: Follow the wizard. The default path is `C:\Program Files\G.D.G. Software\VBA Padlock`, but you can install it anywhere (including a USB drive for portability).

4 **First Launch**: Open VBA Padlock from your Start Menu. You'll be greeted by the **Welcome Screen**, ready to protect your first project.

2.3 Trial vs. Full Version

We provide a fully functional trial so you can test all features before committing.

- **No Time Limit**: Explore all features at your own pace.
- **Nag Screen**: Compiled DLLs will show a "Trial" notification.
- **Expiration**: Compiled DLLs expire after 5-7 days and are not for distribution.

- **Permanent DLLs**: Create unlimited, permanent software solutions.
- **White-Label**: No "VBA Padlock" mentions in your protected work.
- **Priority Support**: Access to direct assistance from our engineers.
- **Unlock now**: Visit the Store →

2.4 Verifying Success

To ensure everything is correctly configured, try to load any macro-enabled file (`.xlsm`, `.docm`, etc.) into the software. If you see the project tree and code editor load without errors, you are ready to compile!

2.5 Recommended Next Steps

Quick Start Guide

Learn the core workflow: Write code, compile, and inject. Get started →

VBA Padlock Studio

A tour of the Studio, the editors, and the settings tabs. Explore the Studio →

3. Quick Start Guide

This guide will take you from a standard Office file to a secure, compiled DLL solution in 7 simple steps.

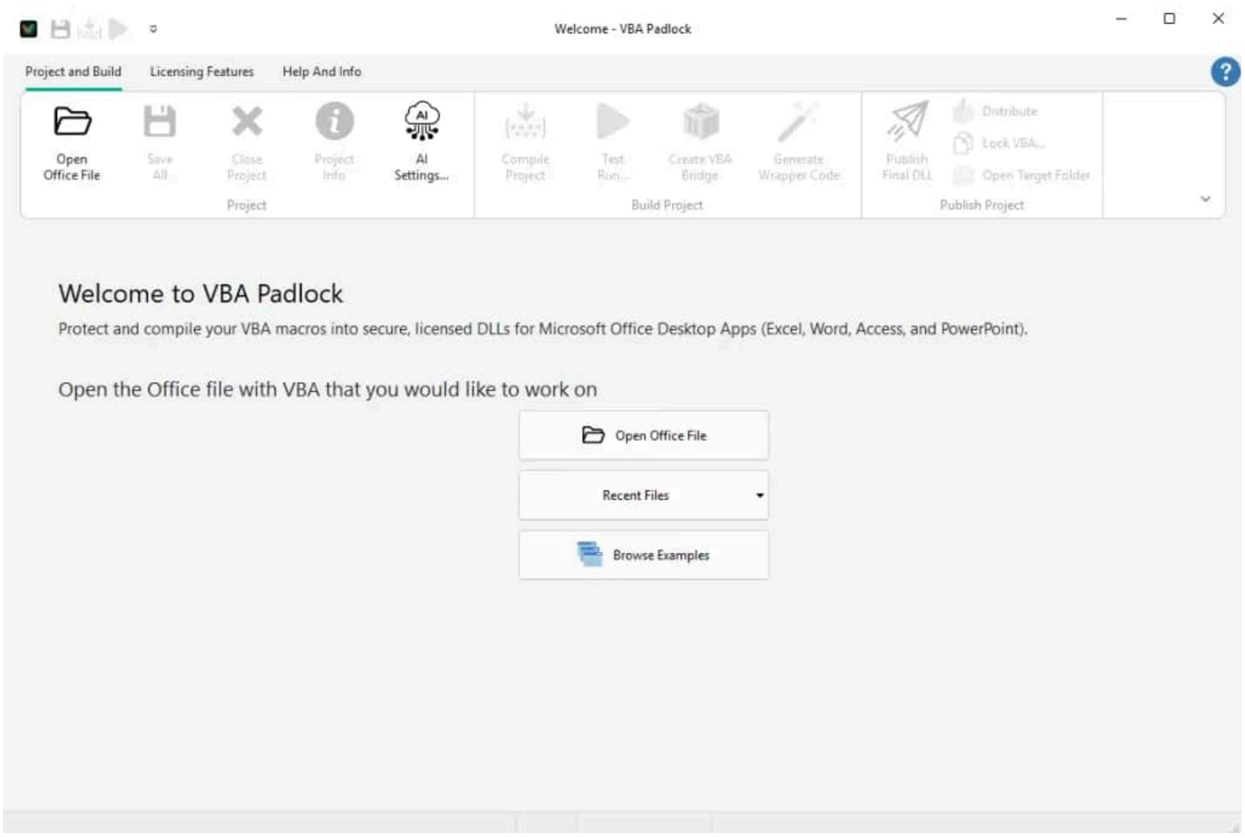
3.1 1. The Core Workflow

Everything in VBA Padlock follows this logical sequence:

1. **Open**: Point the software to your `.xlsm` or `.accdb` file.
2. **Code**: Write or paste the logic you want to protect into the internal editor.
3. **Compile**: Press `F5` to turn your code into an encrypted DLL.
4. **Inject**: Automatically add the “VBA Bridge” to your Office file.
5. **Call**: Invoke your secure functions using `VBAPL_Execute`.

3.2 2. Step-by-Step Walkthrough

1 **Launch & Open**: Open VBA Padlock and click **Open Office File**. Select your macro-enabled file.


Welcome - VBA Padlock

Welcome to VBA Padlock

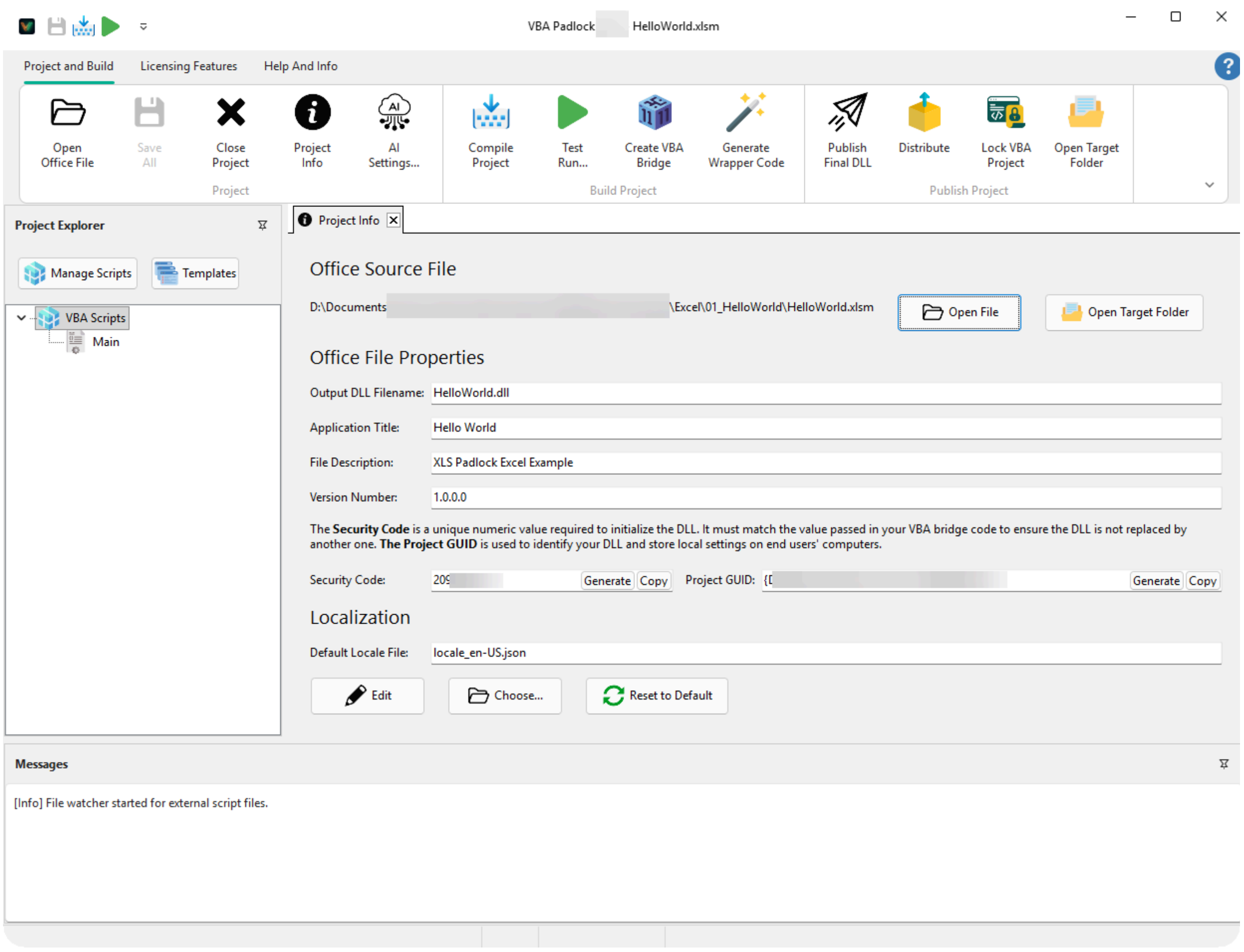
Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

Tip

You can simply drag and drop your Office file directly into the interface!

2 **Project Setup**: Enter your project name and version in the **Project Info** tab. This helps you keep track of your builds.



Office Source File

D:\Documents\Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description: XLS Padlock Excel Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 20 Project GUID: {

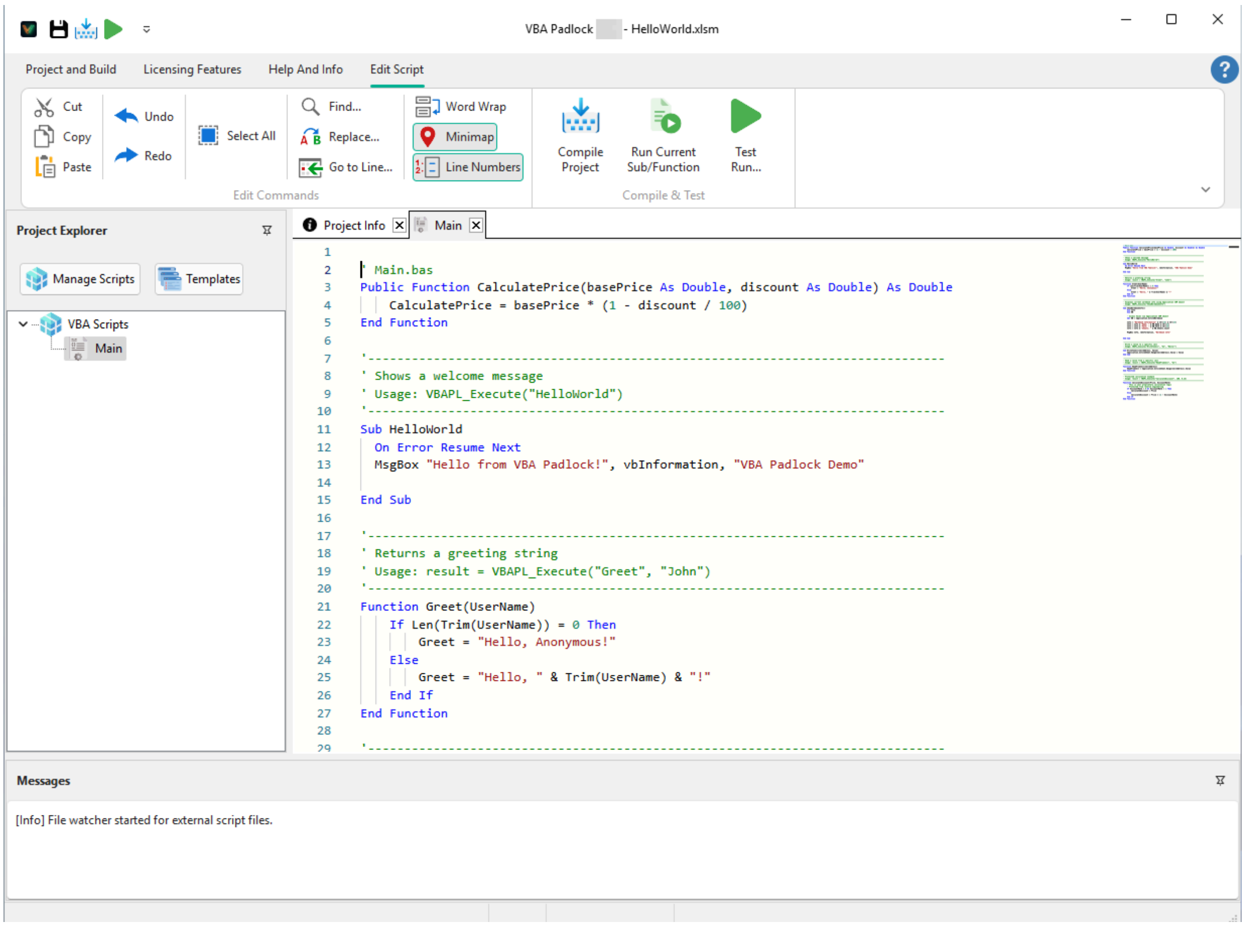
Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

3 **Secure Your Logic**: Go to the **Code Editor**. This is where your intellectual property lives. Write your functions here using standard VBA syntax.

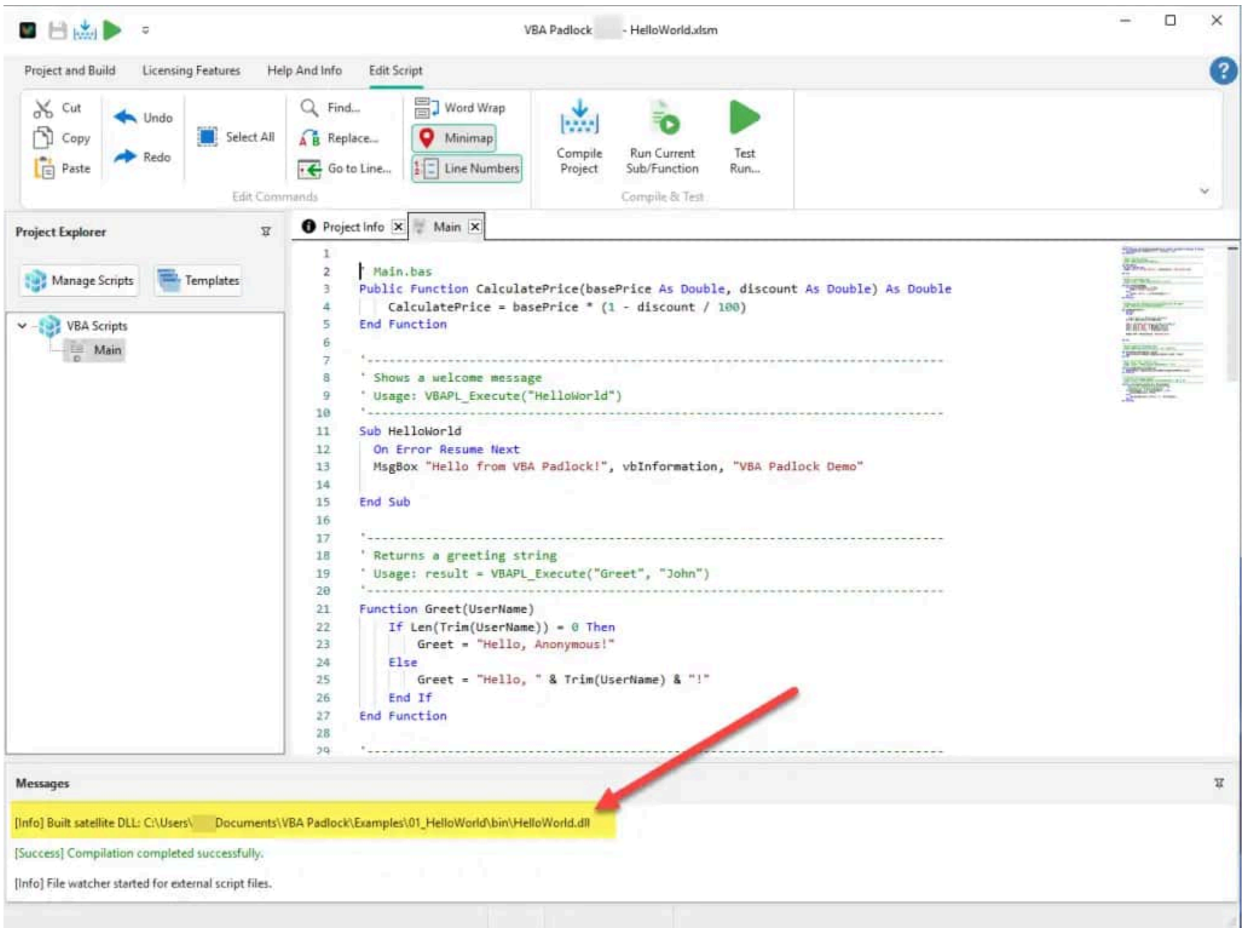
```vb
' Main.bas
' This logic is compiled into the DLL and is no longer present in your Office file.
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function
```



```vb

' Main.bas
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function

'--------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'--------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'--------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'--------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'--------------------------------------------------------------------------------
```

[Info] File watcher started for external script files.

4 **Compile (`F5`)**: Click the **Compile** button. VBA Padlock transforms your code into bytecode.



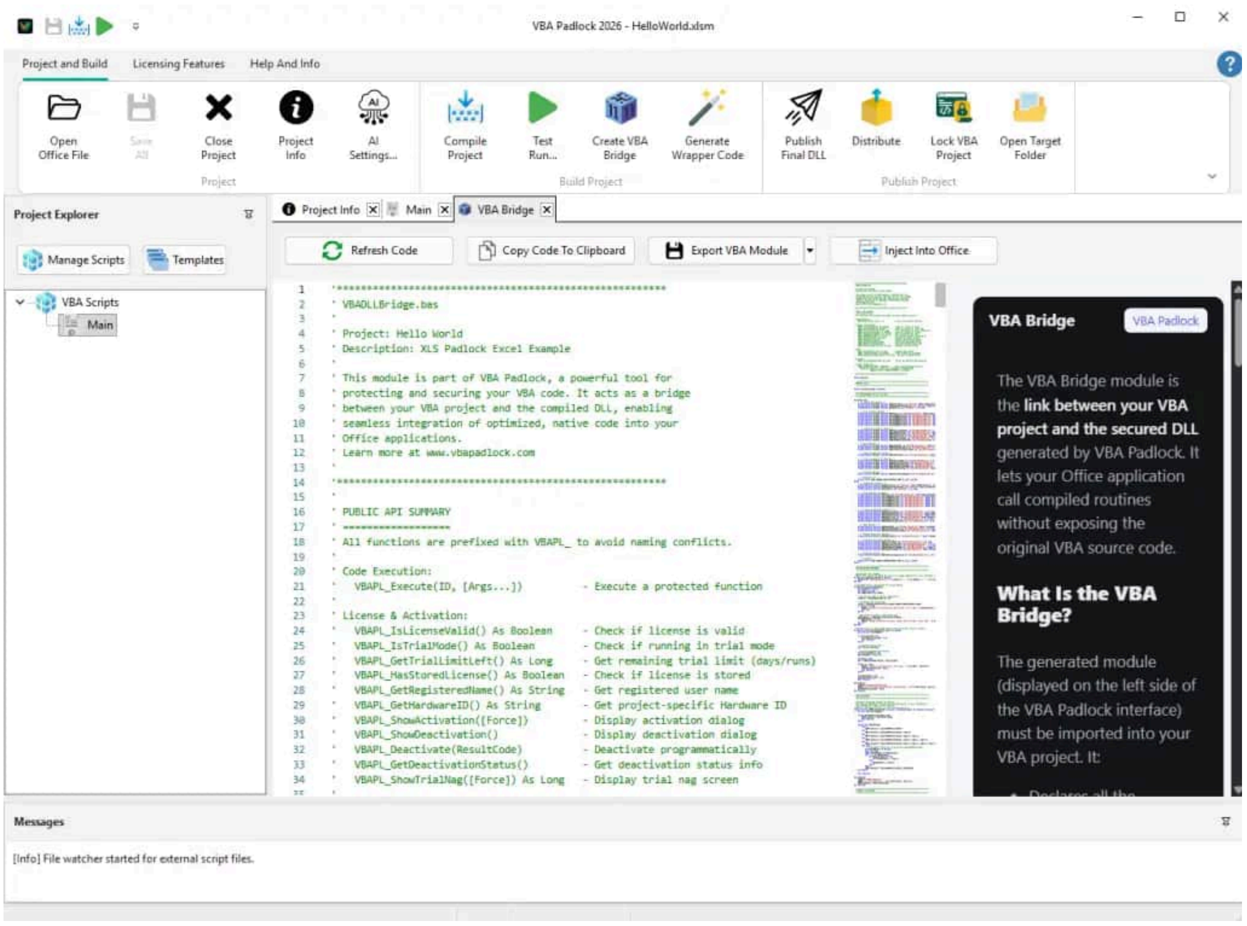
```vb
' Main.bas
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function

'----------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'----------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'----------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'----------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'----------------------------------------------------------------------------
```

```
[Info] Built satellite DLL: C:\Users\[redacted]Documents\VBA Padlock\Examples\01_HelloWorld\bin\HelloWorld.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

Hot Reloading

You don’t need to restart Excel! Every time you recompile, the DLL is updated in memory. Just switch back to your workbook and run your macro again to see the changes instantly.

5 **Inject the VBA Bridge**: Click **VBA Bridge → Inject Into Office**. This adds a specialized module to your workbook that communicates with the DLL.



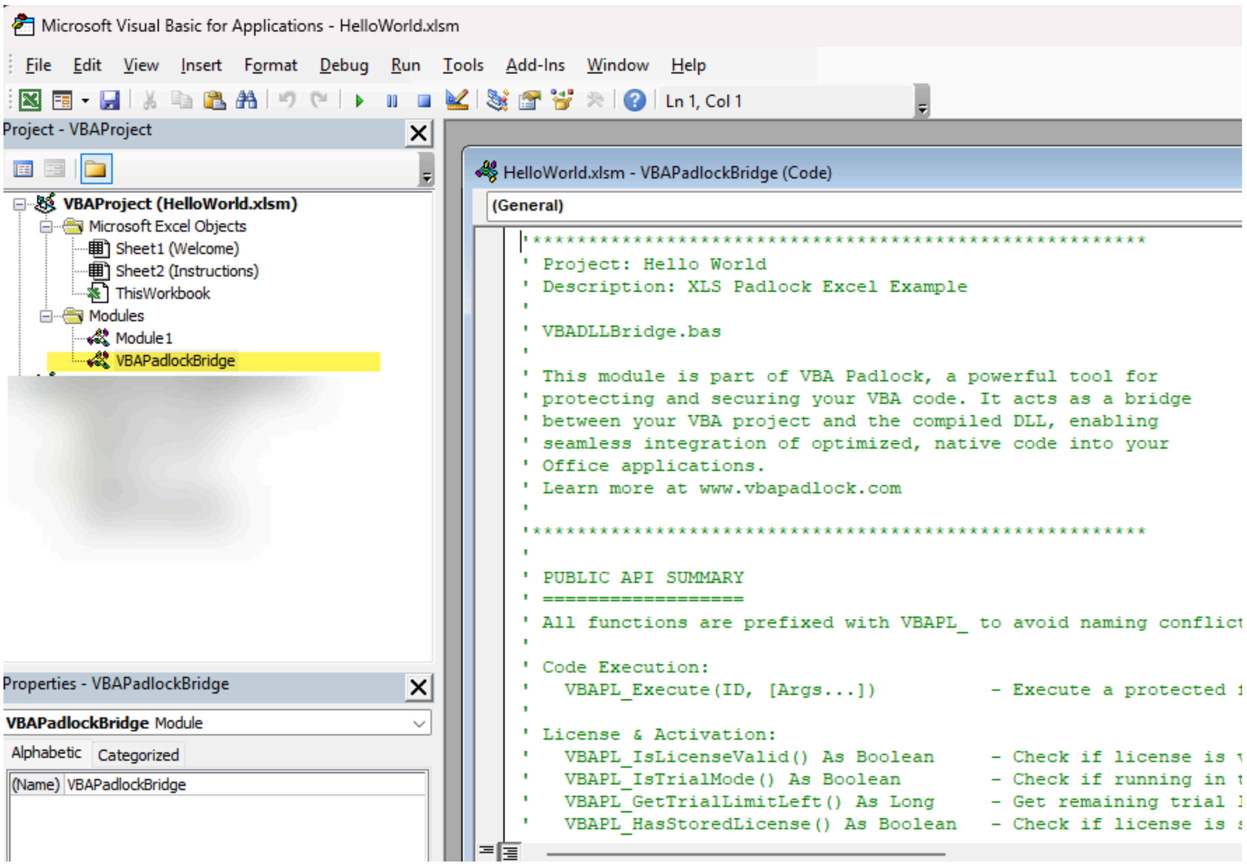
```
'**********************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'**********************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean        - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is stored
'   VBAPL_GetRegisteredName() As String   - Get registered user name
'   VBAPL_GetHardwareID() As String       - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])         - Display activation dialog
'   VBAPL_ShowDeactivation()              - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)          - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()         - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long   - Display trial nag screen
'
```

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

[Info] File watcher started for external script files.



```
'**********************************************************
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' VBADLLBridge.bas
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'**********************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflict
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected f
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is v
'   VBAPL_IsTrialMode() As Boolean        - Check if running in t
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial l
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is s
```

6 **Call Your Code**: In your Excel VBA editor (ALT+F11), call your compiled function using the bridge.

```
Sub RunProtectedCode()
    Dim result As Double
    ' Call your compiled function: "FunctionName", Arguments...
    result = VBAPL_Execute("CalculatePrice", 1000, 15)
    MsgBox "Final Price: " & result
End Sub
```

7 **Success!**: Your code is now running from the DLL. If you look at your Excel VBA project, you'll only see the bridge — your `CalculatePrice` logic is gone.

3.3 3. What to Distribute

After compilation, your project folder will contain a `bin/` directory. This is the heart of your application.

```
  MyProject.xlsm
▼ bin/
      MyProject.dll
      MyProjectrun32.dll
      MyProjectrun64.dll
▶ MyProject.vbapadlock/
```

3.4 4. Next Steps

Excel Tutorial

A deep dive into protecting an Excel Workbook with advanced features. Read the guide →

Licensing Setup

Learn how to add trial periods and hardware locking to your build. Configure licensing →

VBA Bridge API

View the full list of `VBAPL_*` functions for advanced integration. View API →

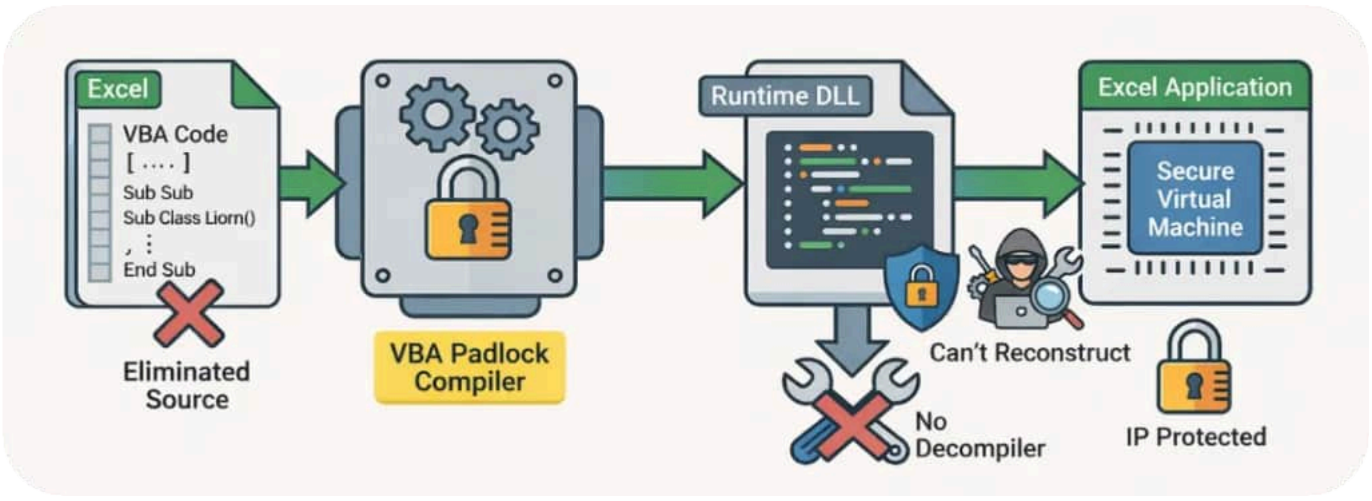
Compatibility

Check which VBA features and libraries are supported in the compiler. Check compatibility →

4. VBA Compilation

VBA Padlock’s core feature is its ability to **compile VBA source code into native, protected DLLs**.

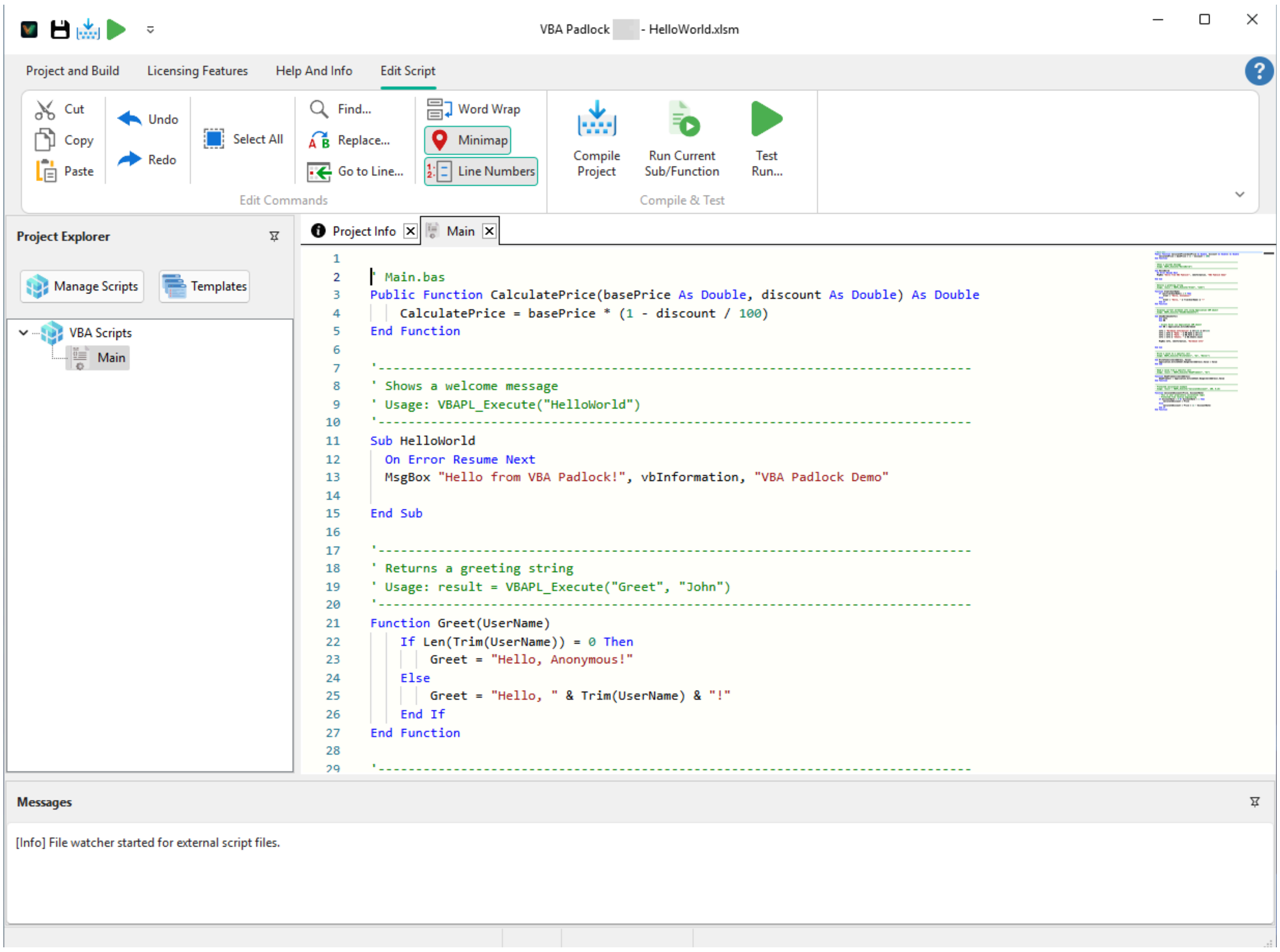
Instead of leaving your intellectual property exposed in the insecure VBA editor, your logic is transformed into encrypted bytecode and moved into external library files. The Office document itself only contains a lightweight **VBA Bridge** module that acts as a secure connector.



4.1 How it Works

The compilation process is designed to be seamless while providing maximum security.

Write your sensitive VBA logic directly within the **VBA Padlock Studio** editor. You can use standard VBA syntax and access the full Office Object Model.

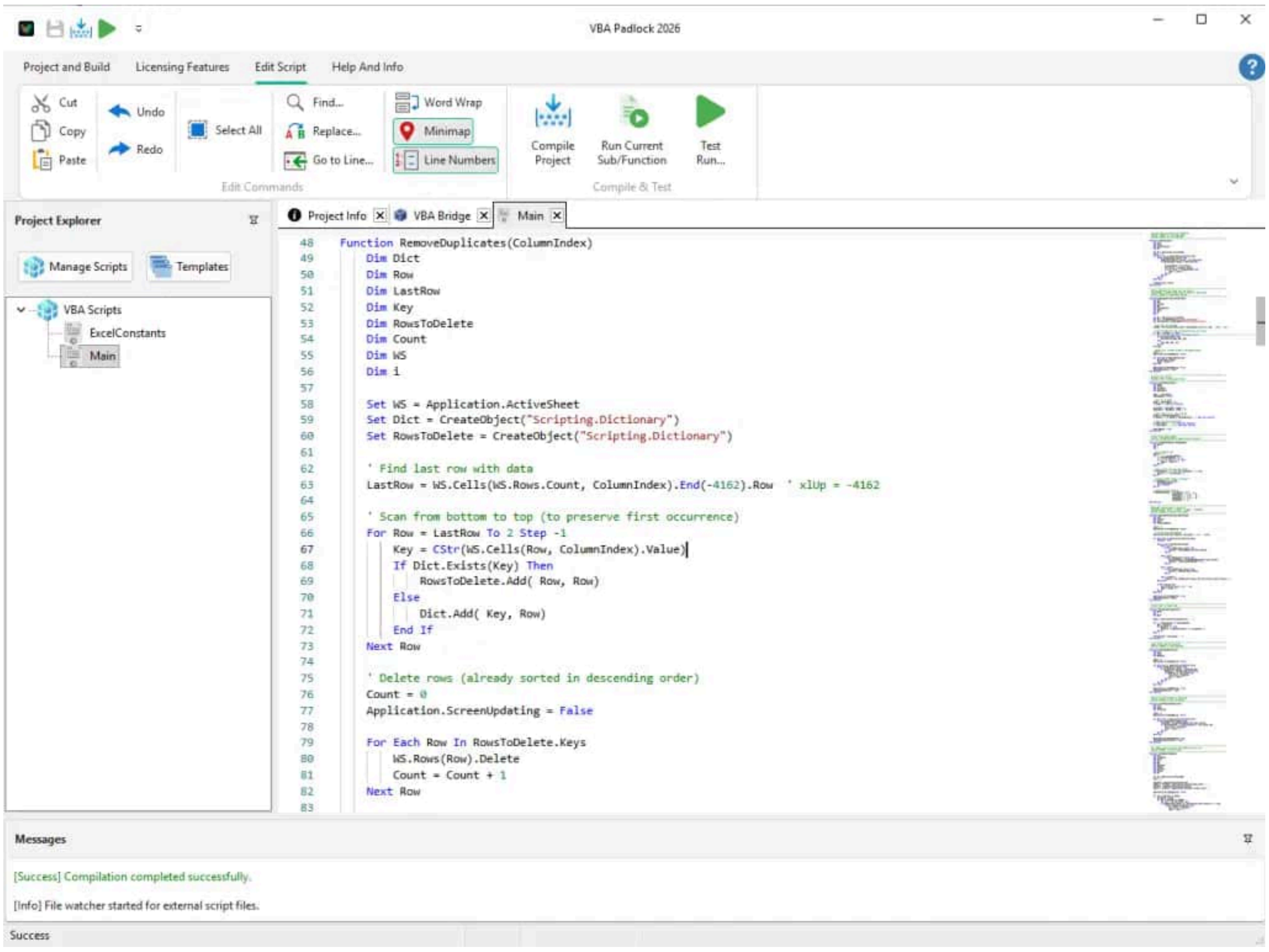


```vba

' Main.bas
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function

'-------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'-------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'-------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'-------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'-------------------------------------------------------------------------------
```

```
[Info] File watcher started for external script files.
```

Developer Tip: *Because your scripts are stored as plain text files, you can use* ***AI Coding Agents*** *(like Claude Code, Gemini, OpenAI Codex or Cursor) to help you refactor code, fix compatibility issues, or even write entirely new protected modules.*

1. Press `F5` in the Studio.
2. VBA Padlock translates your source code into a unique, encrypted bytecode.
3. A project-specific **Satellite DLL** is generated, containing this secure code.

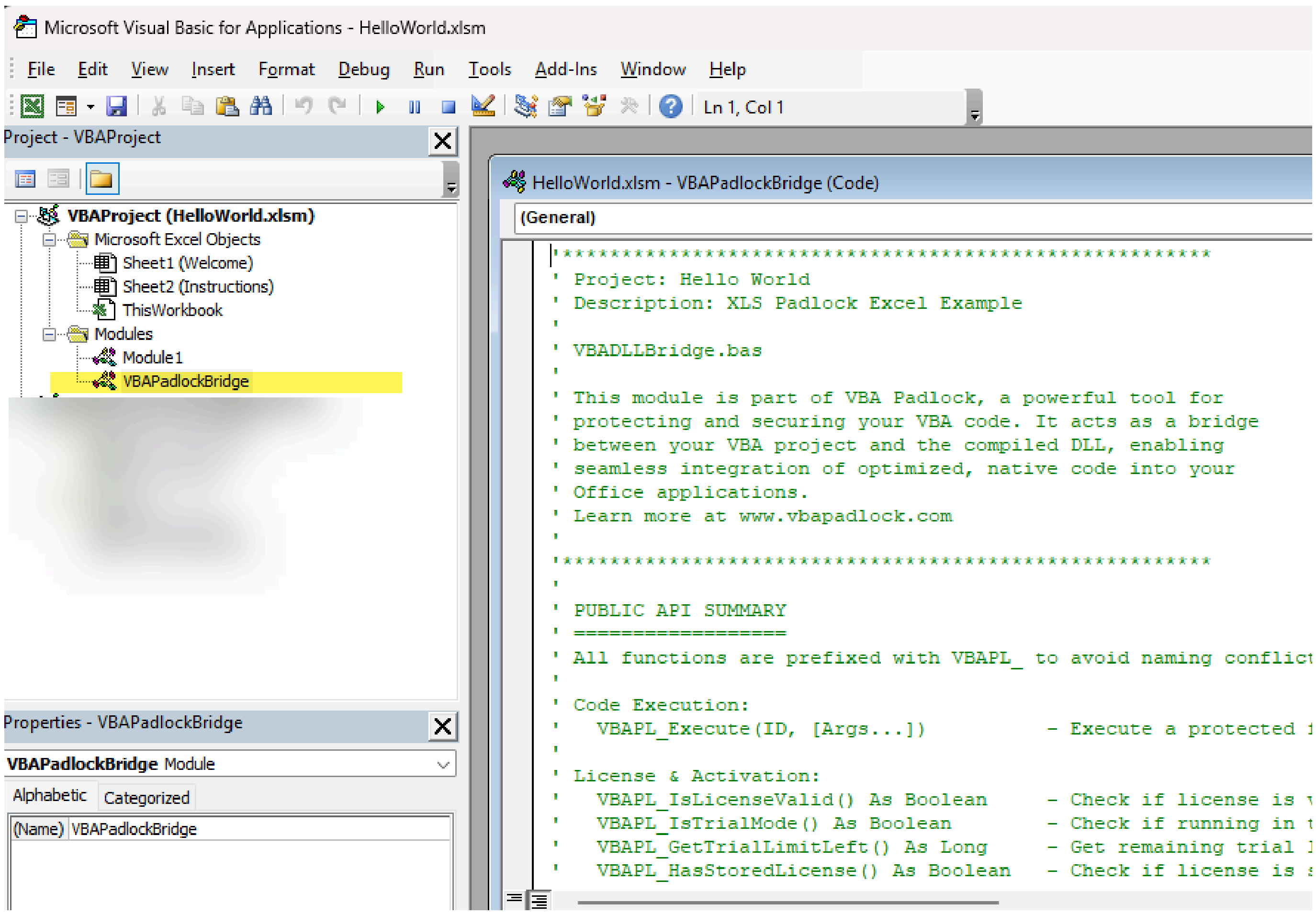


```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Dim Row
    Dim LastRow
    Dim Key
    Dim RowsToDelete
    Dim Count
    Dim WS
    Dim i

    Set WS = Application.ActiveSheet
    Set Dict = CreateObject("Scripting.Dictionary")
    Set RowsToDelete = CreateObject("Scripting.Dictionary")

    ' Find last row with data
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp = -4162

    ' Scan from bottom to top (to preserve first occurrence)
    For Row = LastRow To 2 Step -1
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Dict.Exists(Key) Then
            RowsToDelete.Add( Row, Row)
        Else
            Dict.Add( Key, Row)
        End If
    Next Row

    ' Delete rows (already sorted in descending order)
    Count = 0
    Application.ScreenUpdating = False

    For Each Row In RowsToDelete.Keys
        WS.Rows(Row).Delete
        Count = Count + 1
    Next Row
```

```
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

Success

VBA Padlock automatically injects a **VBA Bridge** module into your Excel, Word, or PowerPoint file. This module contains simple “wrapper” functions that you call from your UI (buttons, events).


Microsoft Visual Basic for Applications - HelloWorld.xlsm
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - VBAProject
VBAProject (HelloWorld.xlsm)
Microsoft Excel Objects
Sheet1 (Welcome)
Sheet2 (Instructions)
ThisWorkbook
Modules
Module1
VBAPadlockBridge
Properties - VBAPadlockBridge
VBAPadlockBridge Module
Alphabetic
Categorized
HelloWorld.xlsm - VBAPadlockBridge (Code)
(General)

(Name) VBAPadlockBridge

```
'*******************************************************
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' VBADLLBridge.bas
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*******************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflict
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected f
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean      - Check if license is v
'   VBAPL_IsTrialMode() As Boolean         - Check if running in t
'   VBAPL_GetTrialLimitLeft() As Long      - Get remaining trial l
'   VBAPL_HasStoredLicense() As Boolean    - Check if license is s
```

4.2 Output Structure

When you compile, VBA Padlock creates a `bin/` directory containing the files necessary for your application to run.

Distribution Package

```
MyWorkbook.xlsm    ← Protected Office file (VBA Project is locked)
▼ bin/    ← Distribution folder (required)
    MyWorkbookrun32.dll    ← 32-bit Secure Runtime
    MyWorkbookrun64.dll    ← 64-bit Secure Runtime
    MyWorkbook.dll    ← Your compiled logic (Satellite DLL)
```

Component	Responsibility	Protection
Runtime DLLs	Interface between Office and your code.	Authenticode Signed by G.D.G. Software.
Satellite DLL	Stores your encrypted VBA bytecode.	Cryptographic Integrity Verification.
Office File	UI and public entry points.	VBA Project Password Protected.

4.3 Capabilities & Integration

4.3.1 Accessing the Office Environment

Compiled scripts are not “disconnected” from your workbook. They have full, high-speed access to the host application.

- **Full Object Model**: Use `Application.ActiveWorkbook` / `ActiveSheet` (Excel), `Application.ActiveDocument` (Word), `Application.ActivePresentation` (PowerPoint), or `Application.CurrentDb` (Access).
- **COM Support**: Use `CreateObject` for FileSystemObject, ADODB, or custom libraries.
- **Built-in API**: Access over 60+ specialized functions for licensing and security.

4.3.2 Code Example: Calling the DLL

Calling a protected function is simple. Use the `VBAPL_Execute` function provided by the VBA Bridge.

```vb
' --- OFFICE VBA SIDE (Module1.bas) ---
Public Sub OnClick_Calculate()
    Dim result As Variant
    ' Call "CalculateTax" inside the compiled DLL
    result = VBAPL_Execute("CalculateTax", Range("A1").Value)
    MsgBox "Tax Result: " & result
End Sub
```

```vb
' --- COMPILED SIDE (VBA Padlock Studio) ---
Public Function CalculateTax(Amount)
    ' This code is invisible to the user and survives reverse-engineering
    CalculateTax = Amount * 0.20
End Function
```

Tip

The VBA Bridge automatically handles the architecture (32-bit vs 64-bit) for you. You don’t need to write `Declare PtrSafe` or `LongPtr` logic in your Office VBA.

4.4 Integration with Microsoft Office

A common misconception is that compiled code is "isolated" from your document. On the contrary, **your protected scripts have full, native access to the host application.**

> **The Application Object**
>
> In VBA Padlock Studio, the `Application` object is your gateway to everything. You don't need to define it; it is automatically provided to your script at runtime.

4.4.3 Support per Application

Host Application	Gateway Object	Typical Usage
Microsoft Excel	`Application.ActiveWorkbook`	Read/Write cells, format ranges, protect sheets.
Microsoft Word	`Application.ActiveDocument`	Generate reports, replace bookmarks, manage sections.
Microsoft Access	`Application.CurrentDb`	Execute SQL queries, open recordsets, manage data.
Microsoft PowerPoint	`Application.ActivePresentation`	Create slides, automate animations, export to PDF.

> ***Pro Tip:*** *Moving your heavy data processing into the compiled DLL is not only more secure, it is often faster because it bypasses the overhead of the standard VBA interpreter for complex logic.*

4.5 Security & Reliability

- **Native Performance**: Bytecode is executed by a custom-built virtual machine optimized for VBA logic.
- **Anti-Tamper**: The Satellite DLL is bound to the Runtime DLLs. Modify one, and the application will refuse to load.
- **AV Friendly**: Our runtime DLLs are digitally signed, ensuring high trust with antivirus software and Windows SmartScreen.

4.6 Next Steps

The VBA Bridge

Learn how to customize the communication between Office and your DLL. View API Reference →

VBA Compatibility

Review supported VBA syntax and limitations. Check Compatibility →

Protect a Workbook

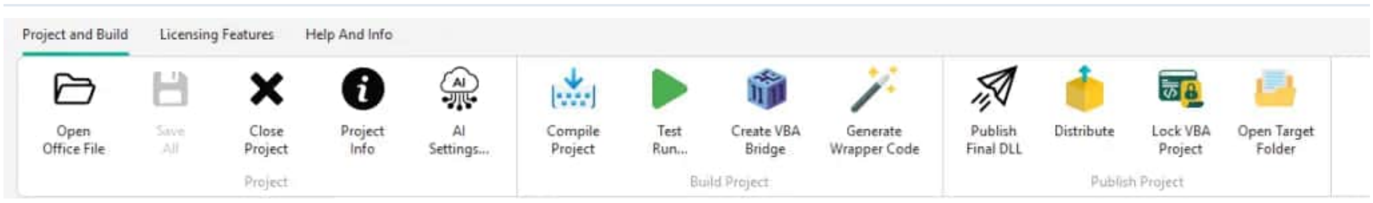
Follow our step-by-step tutorial for Excel. Start Tutorial →

5. VBA Padlock Studio

VBA Padlock includes a dedicated development environment — **VBA Padlock Studio** — for managing your projects, writing protected scripts, configuring licensing, and compiling your VBA code into protected DLLs.

5.1 The Ribbon Interface

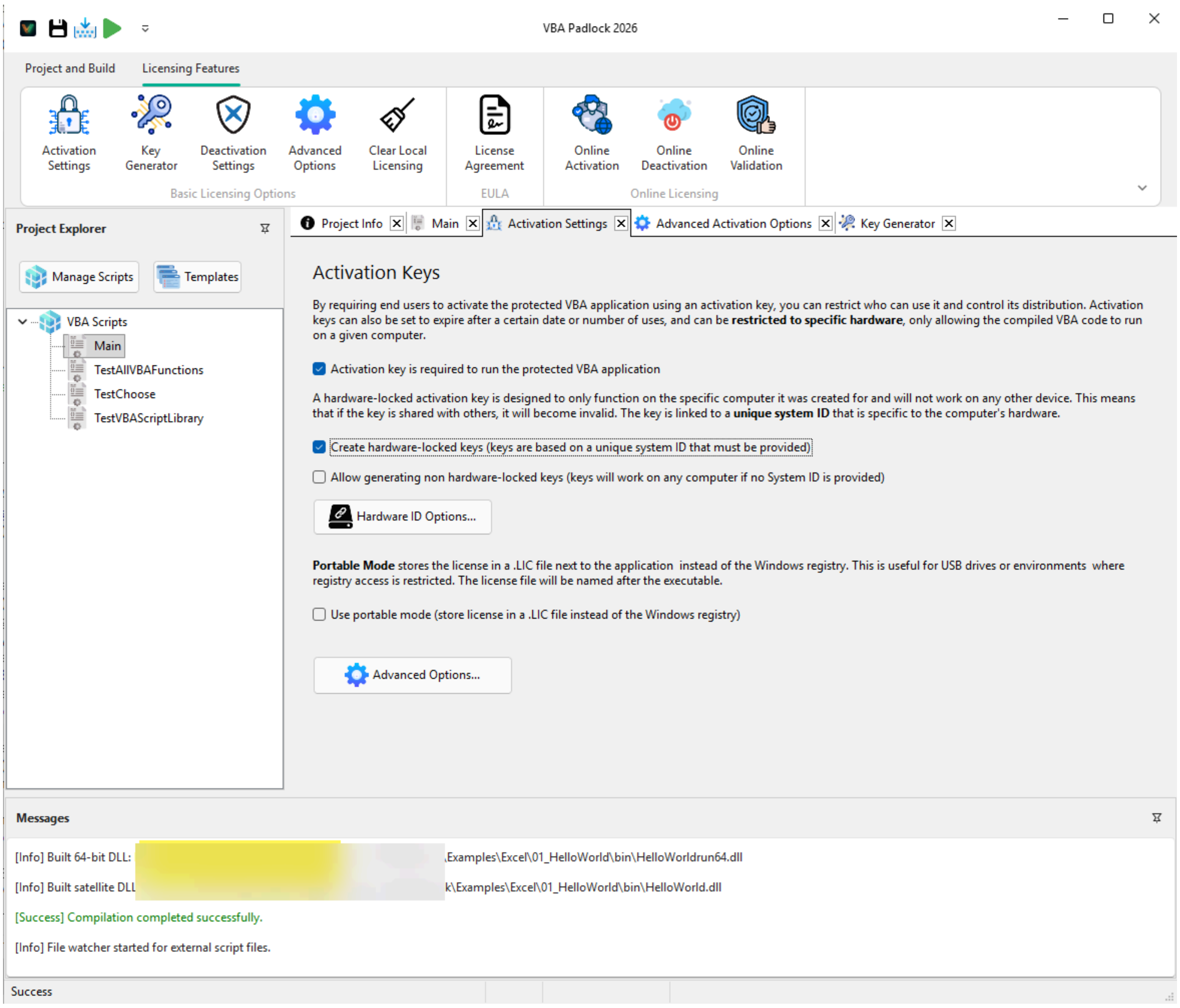
The Studio uses a modern **ribbon interface** designed to follow the natural workflow of protecting your application.



The primary tab for development where you manage your source code and build your protected project.

5.1.1 Key Operations

- **Open Office File**: Load your Excel, Word, or PowerPoint file to start.
- **Project Info**: Define your DLL metadata (name, version) and unique **Security Code**.
- **Script Templates**: Access a library of pre-built code patterns.
- **Compile** (`F5`): Transform your VBA script into an encrypted, compiled DLL.
- **Test Runner** (`Ctrl+F6`): Test your compiled functions instantly without opening Excel.
- **VBA Bridge & Wrapper**: Tools to automate the connection between Office and your DLL.
- **Publish & Distribution**: Finalize your build and prepare files for delivery.
- **Advanced Tools**: Password-protect your VBA project or create a full installer.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

- [x] Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

- [x] Create hardware-locked keys (keys are based on a unique system ID that must be provided)
- [ ] Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

- [ ] Use portable mode (store license in a .LIC file instead of the Windows registry)

```
[Info] Built 64-bit DLL: [redacted]Examples\Excel\01_HelloWorld\bin\HelloWorldrun64.dll
[Info] Built satellite DLL[redacted]k\Examples\Excel\01_HelloWorld\bin\HelloWorld.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

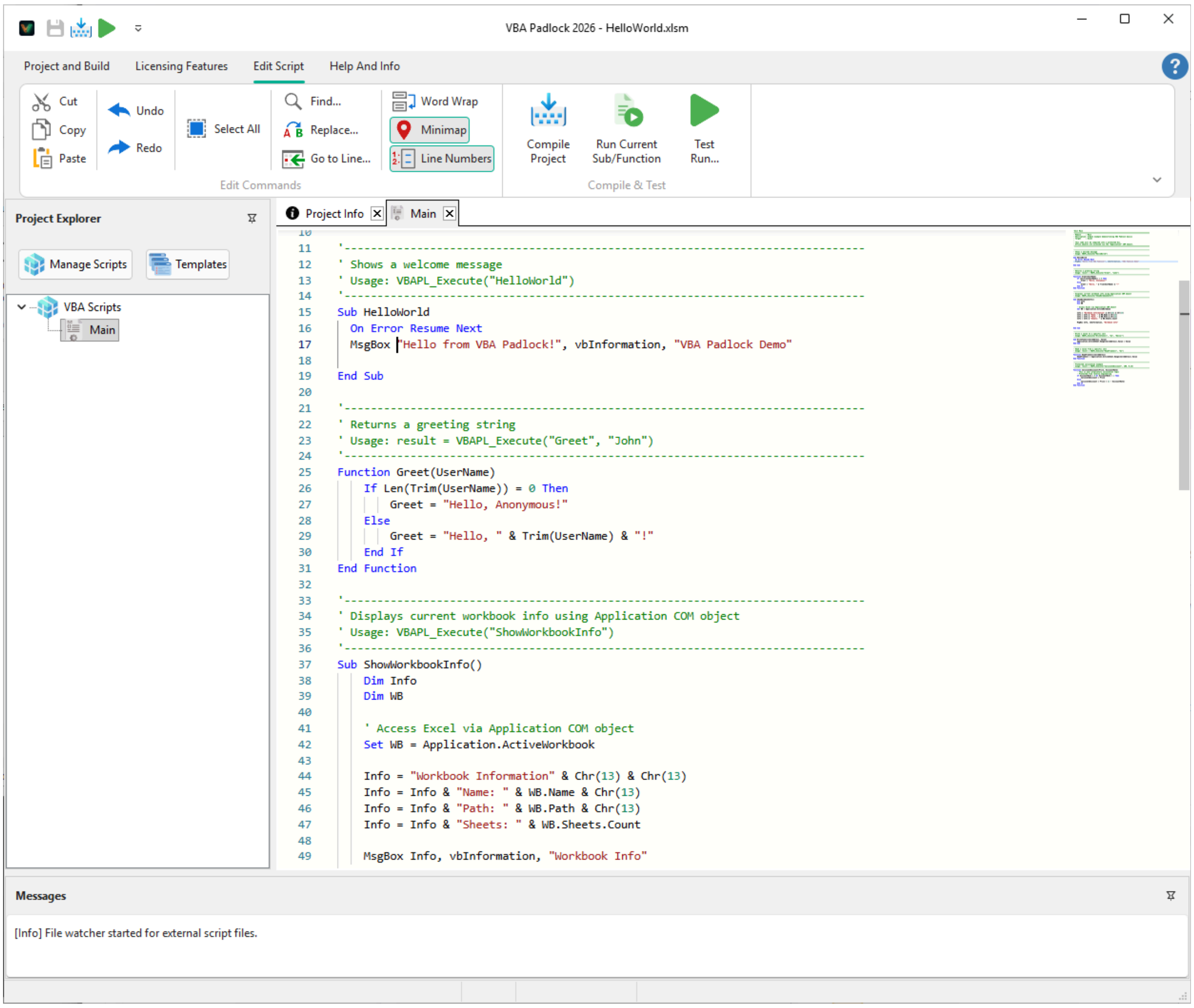
Configure how users access and activate your software.

5.1.2 Protection Options

- **Activation Settings**: Enable trial periods, hardware locking, or portable mode.
- **Hardware ID**: Choose which components (CPU, HDD, Motherboard) lock the license.
- **Online Services**: Set up automated online activation, deactivation, and validation.
- **Key Generator**: Create license keys manually for your customers.
- **EULA**: Insert a custom legal agreement that users must accept before activation.

5.2 Powerful Code Editor

VBA Padlock Studio features a built-in editor specifically optimized for writing compiled VBA logic.



```
'--------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'--------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'--------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'--------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'--------------------------------------------------------------------------------
' Displays current workbook info using Application COM object
' Usage: VBAPL_Execute("ShowWorkbookInfo")
'--------------------------------------------------------------------------------
Sub ShowWorkbookInfo()
    Dim Info
    Dim WB

    ' Access Excel via Application COM object
    Set WB = Application.ActiveWorkbook

    Info = "Workbook Information" & Chr(13) & Chr(13)
    Info = Info & "Name: " & WB.Name & Chr(13)
    Info = Info & "Path: " & WB.Path & Chr(13)
    Info = Info & "Sheets: " & WB.Sheets.Count

    MsgBox Info, vbInformation, "Workbook Info"
```

```
[Info] File watcher started for external script files.
```

- **VBA-Compatible Syntax**: Full highlighting for keywords, strings, and comments.
- **Modular Design**: Organize your code into multiple `.bas` modules (handled as tabs).
- **Secure by Default**: Your code never leaves the encrypted environment until it's compiled into the DLL.
- **Rich Library**: Access over 60+ built-in functions directly from your compiled scripts.

> **Tip**
>
> Don't write everything from scratch! Use **Script Templates** to quickly insert proven patterns for hardware ID checks, license validation, and error handling.

5.3 Project Structure

When you create a project, VBA Padlock Studio creates a hidden environment alongside your Office file to store your source code and settings.

Workspace Layout

```
MyWorkbook.xlsm                ← Your Office file
MyWorkbook.vbapadlock/          ← Secure Project Folder
├── project.xml                 ← Project settings (XML)
├── Main.bas                    ← Your protected VBA logic
├── Helpers.bas                 ← Additional script modules
└── locale_en-US.json           ← Custom UI localized text
```

Note

The `.vbapadlock` folder contains plain text files. This makes it easy to use **Version Control (Git)** to track changes, and also allows **AI Coding Agents** (like Claude Code, Gemini, or Cursor) to read and modify your VBA logic directly to fix errors or improve compatibility.

5.4 Productivity & Automation

5.4.3 Keyboard Shortcuts

Shortcut	Action
`F5`	**Compile Project**
`Ctrl+F6`	**Launch Test Runner**
`Ctrl + S`	Save Current Project
`Ctrl + N`	Create New Module
`Ctrl + O`	Open Office File

5.4.4 Command-Line Interface (CLI)

For CI/CD pipelines and automated builds, you can run the Studio headlessly:

```
VBAPadlock.exe "C:\Path\To\MyFile.xlsm" --compile --silent --log:build.log
```

Switch	Description
`--compile`	Build the project immediately.
`--silent`	Run without a GUI (headless mode).
`--publish`	Build for production (disables HotLoading debugging).
`--log`	Output build results to a specific file.

5.5 Next Steps

UI Reference

Explore every dialog and setting in detail. View reference →

Quick Start

Protect your first workbook in under 5 minutes. Follow the guide →

Compatibility

Check which VBA features are supported in compiled code. Learn more →

6. Security Overview

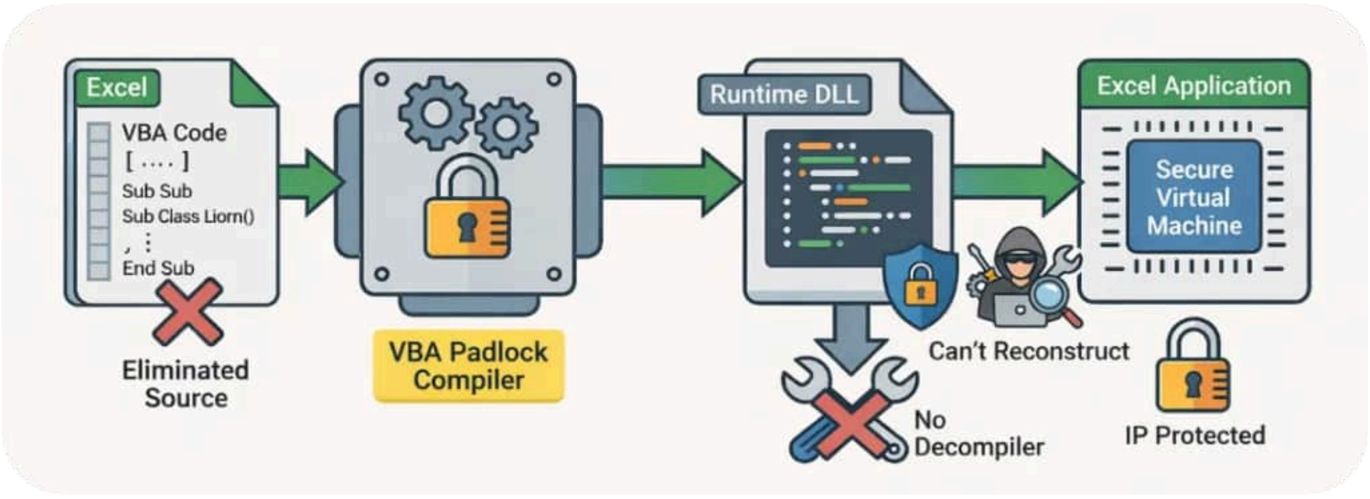
Stop worrying about passwords that can be cracked in seconds. VBA Padlock provides a **professional-grade security architecture** that protects your source code by physically removing it from your Office document and executing it within a secure, compiled environment.

6.1 The Gold Standard: True Compilation

Unlike "VBA Obfuscators" that merely scramble your text, VBA Padlock performs a **one-way compilation** of your source logic.

Source Code Destruction

During the build process, your original VBA source code is **completely eliminated**. It is converted into a proprietary bytecode format that exists only within the satellite DLL. Even the most advanced reverse-engineering tools cannot reconstruct your original code.



6.2 Security Layers

Our defense-in-depth strategy ensures that your application is protected from every angle.

6.2.1 Secure Execution

Your compiled logic is executed by a **proprietary Virtual Machine** (VM) embedded in the runtime.

- **No Readable Text**: There are no string literals or logic flows visible to the naked eye.
- **Sandboxed**: The VM is optimized for secure macro execution without exposing host vulnerabilities.

6.2.2 Verified Identity

VBA Padlock runtimes are **Authenticode-signed by G.D.G. Software**.

- **Trust**: Reduces "untrusted publisher" warnings in Windows SmartScreen.
- **Integrity**: Every satellite DLL has an internal cryptographic signature. If even 1 bit of your DLL is changed by a virus or hacker, the runtime will refuse to load it.

6.2.3 Security Code

Your Office file is "wedded" to its DLL via a unique **Security Code**.

- **Anti-Hijacking**: Prevents someone from taking your DLL and using it with their own Excel file.
- **Initialization Proof**: The DLL only activates if the caller provides the correct project-specific GUID.

6.3 Threat Mitigation

Threat	Our Protection
Logic Theft / IP Theft	Source code is destroyed; only bytecode remains.
Password Crackers	Standard Excel VBA passwords are bypassed; we don't use them for the main logic.
Unauthorized Sharing	Hardware Locking binds the product to one PC.
Binary Tampering	Continuous internal integrity checks detect resource modifications.

6.4 Why VBA Padlock is different

Feature	Standard Excel	Obfuscators	VBA Padlock
Source Code Removed?	❌ No	❌ No	✅ **Yes**
Runtime Encryption?	❌ No	❌ No	✅ **Yes**
Hardware Binding?	❌ No	❌ No	✅ **Yes**
Digital Signatures?	❌ No	❌ No	✅ **Yes**

6.5 Next Steps

VBA Compilation

Learn how your scripts are transformed into secure DLLs. Learn more →

Hardware Locking

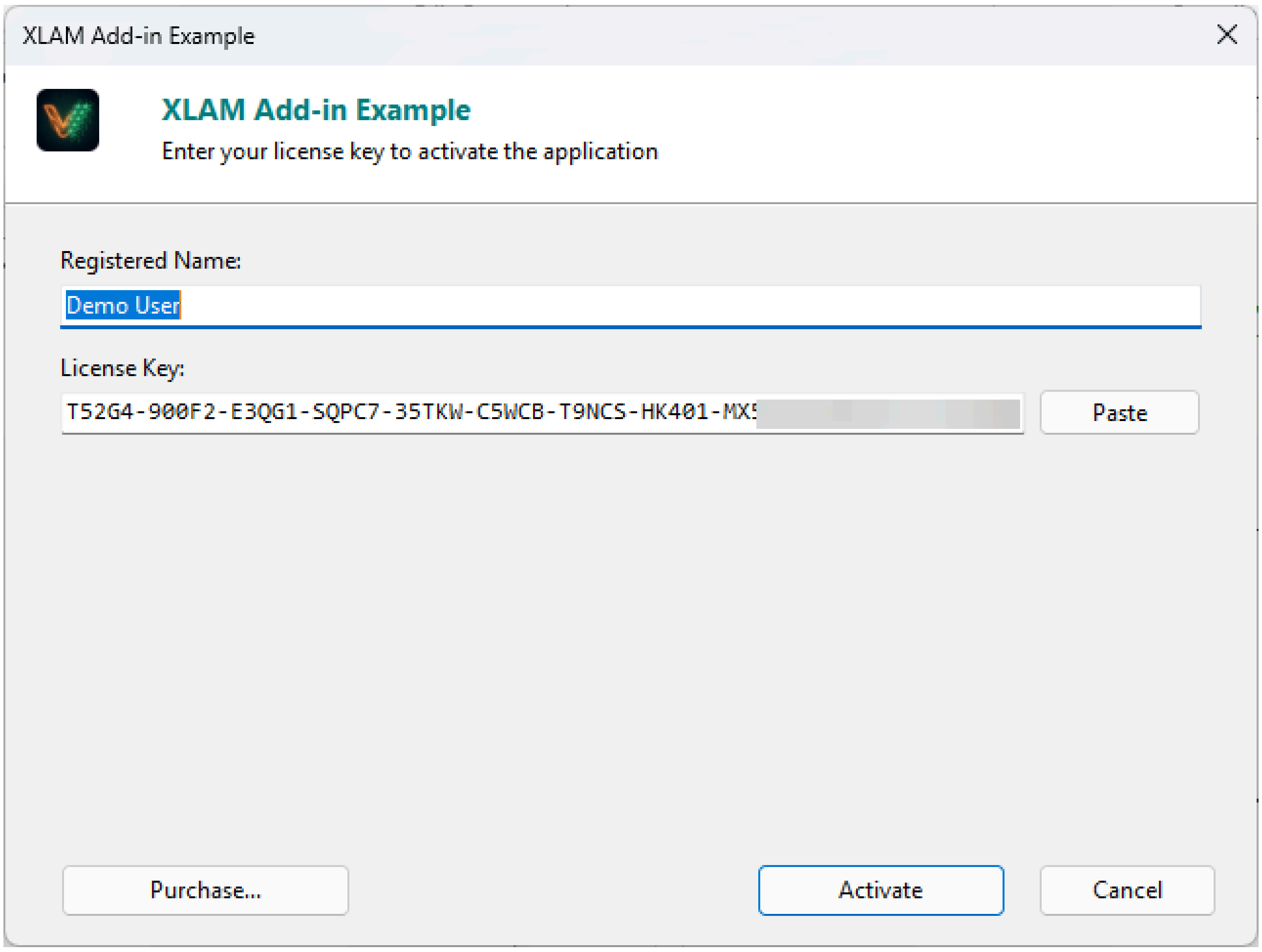
Explore how we bind your code to specific machines. Explore HWID →

Lock VBA Project

Hide the VBA Bridge code that initiates your protection. View UI reference →

7. Licensing System

VBA Padlock provides a modern, robust **Licensing System** that transforms your Excel workbooks or Access databases into commercial software. From simple product keys to complex hardware-locked subscriptions, you have total control over how users access your specialized logic.



XLAM Add-in Example

Enter your license key to activate the application

Registered Name: Demo User

License Key: T52G4-900F2-E3QG1-SQPC7-35TKW-C5WCB-T9NCS-HK401-MX[redacted]

7.1 Choosing Your License Model

Different business models require different types of protection. Choose the one that fits your distribution strategy:

7.1.1 Infinite Access

A permanent license with no restrictions. Once correctly activated, the application works indefinitely on the user's machine.

- No expiration dates.
- Optionally hardware-locked.
- Perfect for "standard" one-time purchase software.

7.1.2 Recurring Revenue

A license that expires on a specific date. The user must enter a new renewal key periodically to continue using the software.

- Expiration date is embedded securely in the key.
- Automated renewal via Online Validation.
- Ideal for SaaS-style Office applications.

7.1.3 Lead Generation

Allow users to test your application for a limited time before buying.

- **Time-Limited**: e.g., 30 days of evaluation.
- **Run-Limited**: e.g., 5 launches allowed.
- Upgradeable to a Full License without re-installing.
- Learn more about Trial Mode →

7.2 Key Formats & Security

VBA Padlock supports two distinct key architectures. We recommend **Long Keys** for maximum cryptographic security.

Short Keys (HMAC)

35 Characters

- **Pros**: Easy to type, fits in standard emails.
- **Best for**: Simple hardware locking and expiration dates.
- **Format**: `XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX`

Long Keys (ECC Ed25519)

160+ Characters

- **Pros**: Maximum security, uses modern elliptic curve signatures.
- **Best for**: Online Activation, run-limited trials, and tamper-resistance.
- **Format**: Digital signature block (copy-paste only).

7.3 Implementation & Workflow

Setting up licensing is a two-step process in VBA Padlock Studio.

1. **Enable Activation**: In Activation Settings, check the **"Activation Key is required"** option.
2. **Generate Master Key**: Go to Advanced Activation Settings to generate your unique encryption keys. These ensure that only *you* can generate valid keys for your project.

7.4 Programming Integration

Verify the license status directly within your code to protect specific macros or features.

VBA Bridge Example

```vba
' Check if the application is fully licensed
If VBAPL_IsLicenseValid() Then
    ' Run proprietary logic
ElseIf VBAPL_IsTrialMode() Then
    MsgBox "Trial: " & VBAPL_GetTrialLimitLeft() & " days left."
Else
    MsgBox "Please activate your license."
End If
```

7.5 Distribution & Storage

Where should the license be stored once the user activates it?

- **Registry (Default)**: Best for standard Windows installations. Keys are stored in `HKEY_CURRENT_USER`.
- **Portable (.LIC file)**: Use this for USB-based applications or environments where users don't have registry write permissions. Enable this via **Portable Mode**.

7.6 Next Steps

Setup Guide

Follow our step-by-step walkthrough for a full implementation. View setup guide →

Hardware Locking

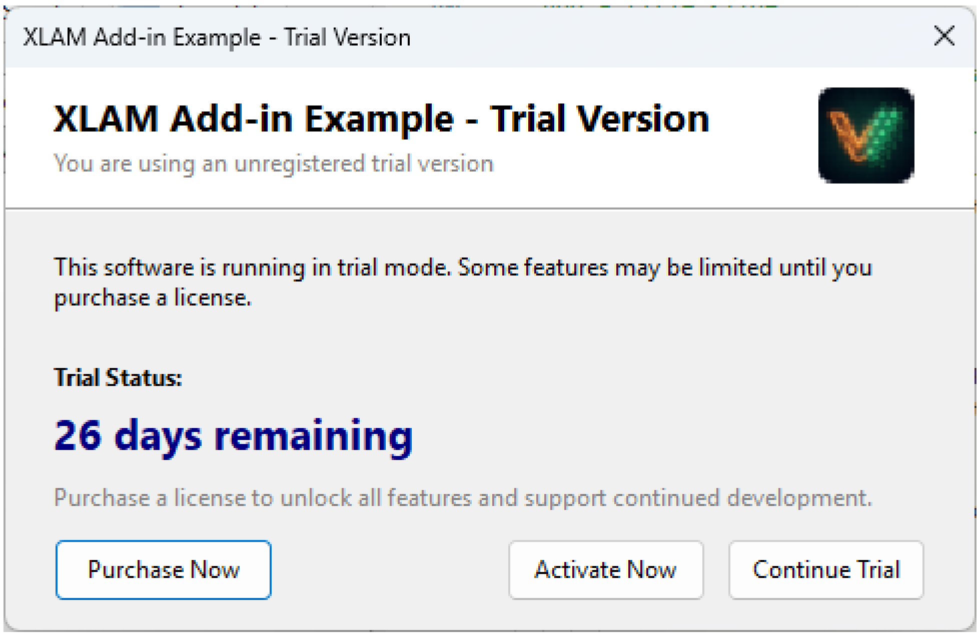
Bind your license keys to specific computers to prevent sharing. Learn more →

Key Generator

Start creating license keys for your users. View UI reference →

8. Trial Mode

Lower the barrier to entry for your potential customers by offering a **Trial Version** of your software. VBA Padlock allows you to distribute your application with built-in evaluation limits that automatically prompt for purchase once the trial expires.


XLAM Add-in Example - Trial Version

XLAM Add-in Example - Trial Version

You are using an unregistered trial version

This software is running in trial mode. Some features may be limited until you purchase a license.

Trial Status:

26 days remaining

Purchase a license to unlock all features and support continued development.

8.1 Trial Strategies

Select the evaluation model that best fits your application’s use case:

8.1.1 30-Day Evaluation

The application runs for a fixed number of days starting from the first launch.

- **Tamper-Proof**: Detects if the user changes their system clock to cheat the trial.
- **Countdown**: Displays “X days remaining” in the trial dialog.

8.1.2 10-Run Trial

The application works for a specific number of launches (e.g., 50 runs).

- **Long Key Only**: Requires the **Long (ECC)** key format.
- **Precision**: Ideal for utilities that are used occasionally rather than daily.

8.1.3 Beta / Seasonal

The application works until a specific calendar date (e.g., the end of the year).

- **Universal**: All users expire at the same time, regardless of when they started.
- **Perfect for**: Beta testing phases or early-access promotions.

8.2 The Nag Screen

The **Nag Screen** is an optional reminder dialog shown at every launch when a **Trial Key** is active. It encourages users to convert by providing direct buttons for purchase and activation. Without a key, even if evaluation is allowed, the Nag Screen will not be displayed automatically.

8.2.4 Configuration Options

In Advanced Activation Options, you can fine-tune the trial behavior:

Mode	Behavior
Always Prompt	Shows the activation dialog on every launch.
Nag Screen	Shows a friendly reminder with a “Continue Trial” button.
Silent Mode	No dialog is shown. You must check the status via VBA code.

Tip

Add a **Purchase URL** in your settings to include a “Purchase Now” button on the nag screen, leading users directly to your online store.

8.3 Integration in Your Code

Use the VBA Bridge to customize the behavior of your application based on the trial status.

Display Trial Info

```
' Office VBA side
If VBAPL_IsTrialMode() Then
    Dim daysLeft As Long
    daysLeft = VBAPL_GetTrialLimitLeft()
    MsgBox daysLeft & " days left in trial."
End If
```

Check in DLL

```
' Inside VBA Padlock Studio
If GetLicenseType() = 2 Then
   ' 2 = Trial Mode
   ' Disable "Premium" features
End If
```

8.4 Recommended Workflow

Most professional developers follow this sequence for distributing trials:

1. **Distribute Trial**: Provide your workbook. Ensure **Allow compiled VBA code to run without activation** is checked for a silent start, or issue a **Trial Key** for a guided evaluation.
2. **Display Nag**: Once a **Trial Key** is entered, the user sees the remaining days and the **Purchase Now** button at every launch.
3. **Customer Buys**: The user purchases an activation key on your website.
4. **Full Activation**: You send a **Full License Key** locked to their machine.
5. **Clean Transition**: Once the full key is entered, the trial limits and nag screens disappear instantly.

8.5 Next Steps

Key Generator

Configure trial limits and generate evaluation keys. View reference →

Localization

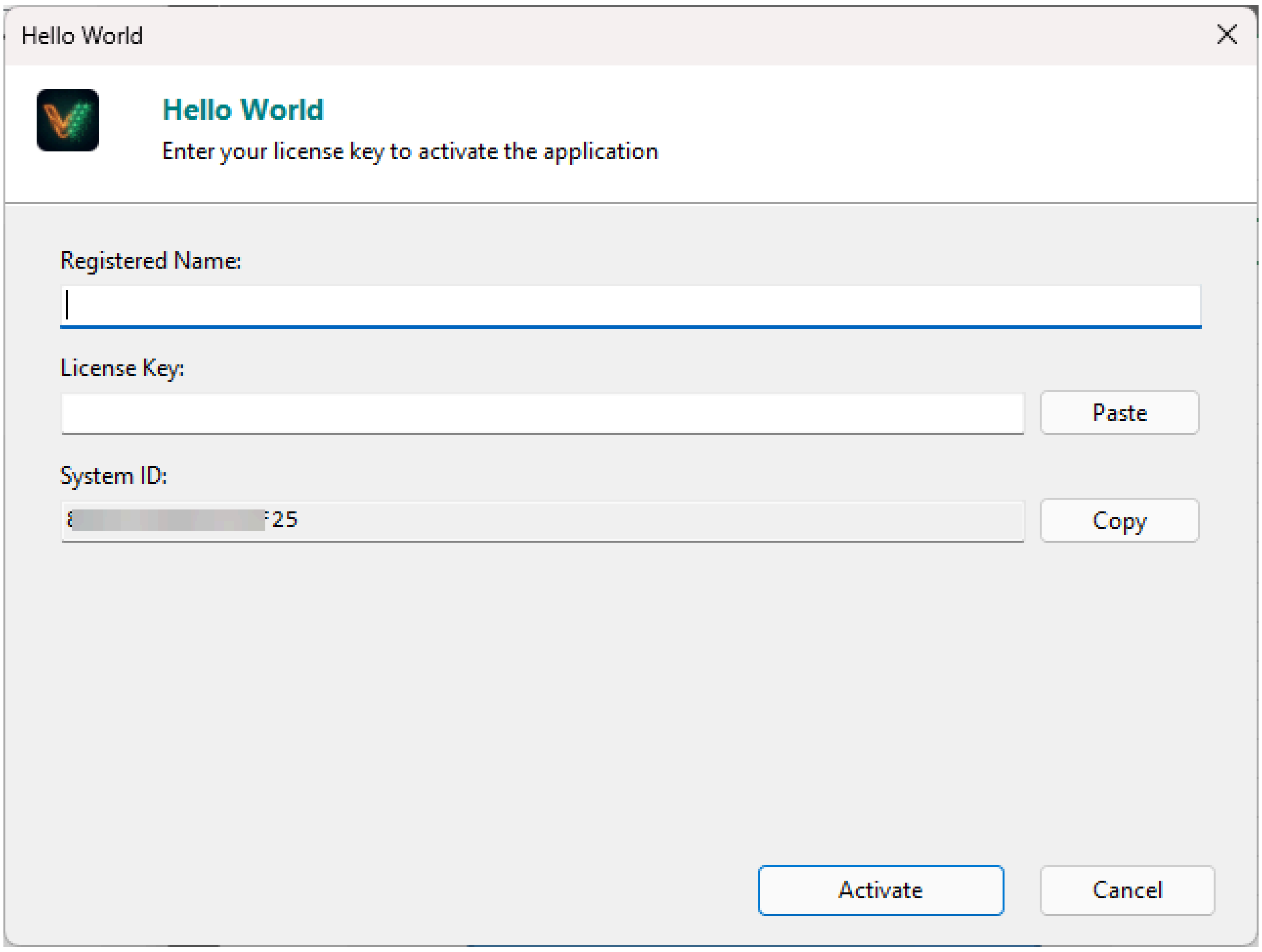
Translate trial countdowns and nag screen labels. Learn how →

Licensing System

Understand the difference between trial and full licenses. View overview →

9. Hardware Locking

Prevent software piracy and unauthorized license sharing by binding your application to a specific machine. **Hardware locking** ensures that a license key generated for one user cannot be used on a different computer.


Hello World

Hello World
Enter your license key to activate the application

Registered Name:

License Key:

System ID: [redacted]25

9.1 How it Works

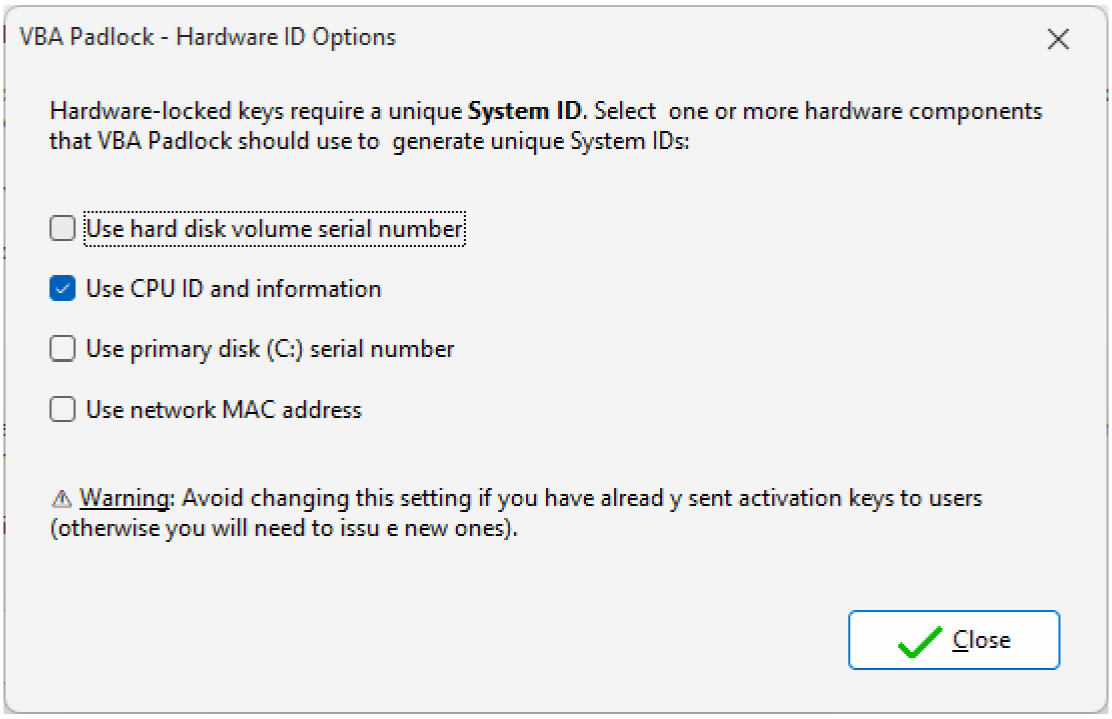
The process creates a unique “fingerprint” of the user’s computer, known as the **Hardware ID** (or System ID).

1. **Fingerprinting**: When your protected file is launched, VBA Padlock reads the machine’s hardware serial numbers.
2. **ID Generation**: A unique, non-reversible Hardware ID is generated for that specific machine.
3. **Key Request**: The user sends you this ID (via the activation dialog or automatically via Online Activation).
4. **Locked Key**: You generate a license key that is cryptographically bound to that specific Hardware ID.

5 **Validation**: On every launch, the software re-checks the hardware. If it doesn't match the key, the application remains locked.

9.2 Hardware Identity Components

You have full control over which hardware parts form the "fingerprint". You can configure these in the **Hardware ID Options** dialog.



VBA Padlock - Hardware ID Options

Hardware-locked keys require a unique **System ID**. Select one or more hardware components that VBA Padlock should use to generate unique System IDs:

- [ ] Use hard disk volume serial number
- [x] Use CPU ID and information
- [ ] Use primary disk (C:) serial number
- [ ] Use network MAC address

⚠ Warning: Avoid changing this setting if you have alread y sent activation keys to users (otherwise you will need to issu e new ones).

Component	Reliability	Stability	Recommended?
CPU Info	High	Very Stable	**Yes** (Default)
HD Serial	High	Stable	**Yes** (Default)
MAC Address	Moderate	Variable	Only for stable networks
Primary Disk	High	Subject to upgrades	Optional

Tip

The default combination of **CPU Info + HD Serial** offers the best balance. It's unique enough for security but stable enough to survive standard Windows updates.

9.3 Retrieving the Hardware ID

There are several ways to get the Hardware ID from your users.

The standard activation dialog displays the Hardware ID prominently. Users can simply copy and paste it into an email or your web store.

You can create a “Get My ID” button in your Excel/Word ribbon:

```
Public Sub ShowHardwareID()
    Dim hwid As String
    hwid = VBAPL_GetHardwareID()
    MsgBox "Your System ID is: " & hwid
End Sub
```

If you use Online Activation, the Hardware ID is sent silently to your server. The system then generates and delivers the locked key automatically.

9.4 Hardware Locking vs. Portable Mode

Sometimes, hardware locking is too restrictive (e.g., users working from multiple computers via USB).

Hardware Locked

- **Maximum Security**: Keys cannot be shared.
- **Usage**: Standard desktop installs.
- **Storage**: Windows Registry.

Portable Mode (.LIC)

- **Moderate Security**: License follows the file.
- **Usage**: USB drives, virtual machines.
- **Storage**: `.LIC` file next to workbook.

9.5 Next Steps

Hardware Options

Configure the components used for the ID fingerprint. View reference →

Key Generator

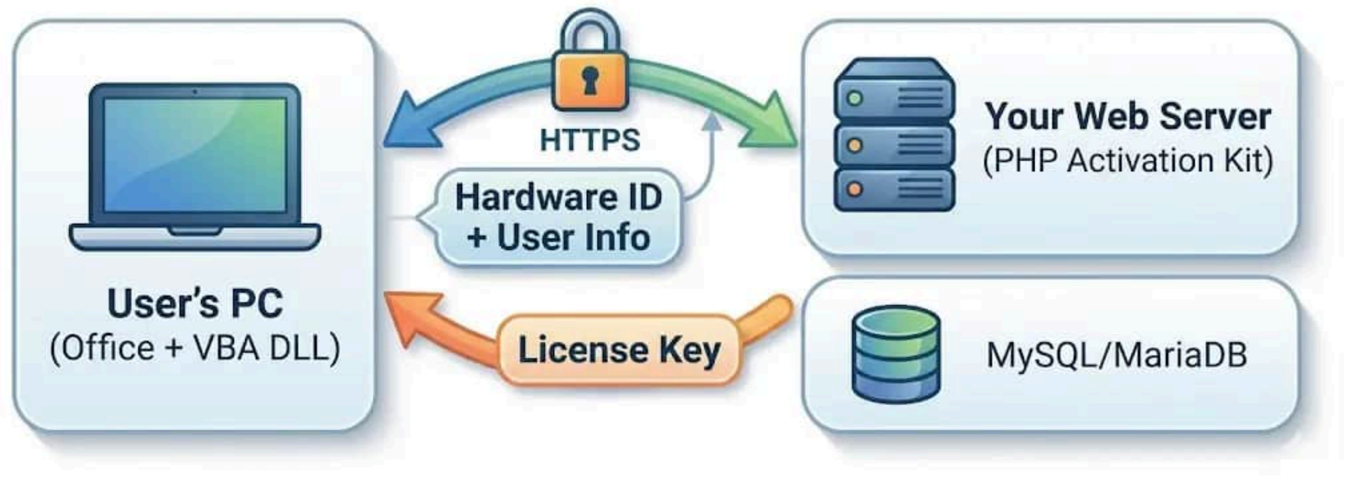
Generate keys locked to specific Hardware IDs. Learn how →

Deactivation

Allow users to transfer their license to a new machine. Learn more →

10. Online Activation

Eliminate the manual work of sending license keys. The **VBA Padlock Activation Kit** is a complete, server-ready PHP application that automates the entire licensing lifecycle — from purchase to activation, validation, and deactivation.



10.1 How It Works

The automated flow provides a seamless experience for your end users while maintaining high security.

1. **Purchase**: The user buys your software (e.g., via the built-in PayPal store).
2. **Activation Code**: Your server generates a unique **Activation Code** and sends it to the user.
3. **App Launch**: The user launches your workbook and enters their code into the activation dialog.
4. **Secure Request**: The app sends the code + Hardware ID to your server via HTTPS.
5. **Key Delivery**: Your server validates the code, decrements the activation count, and returns a verified license key.
6. **Unlocked**: The application activates instantly on the user's machine.

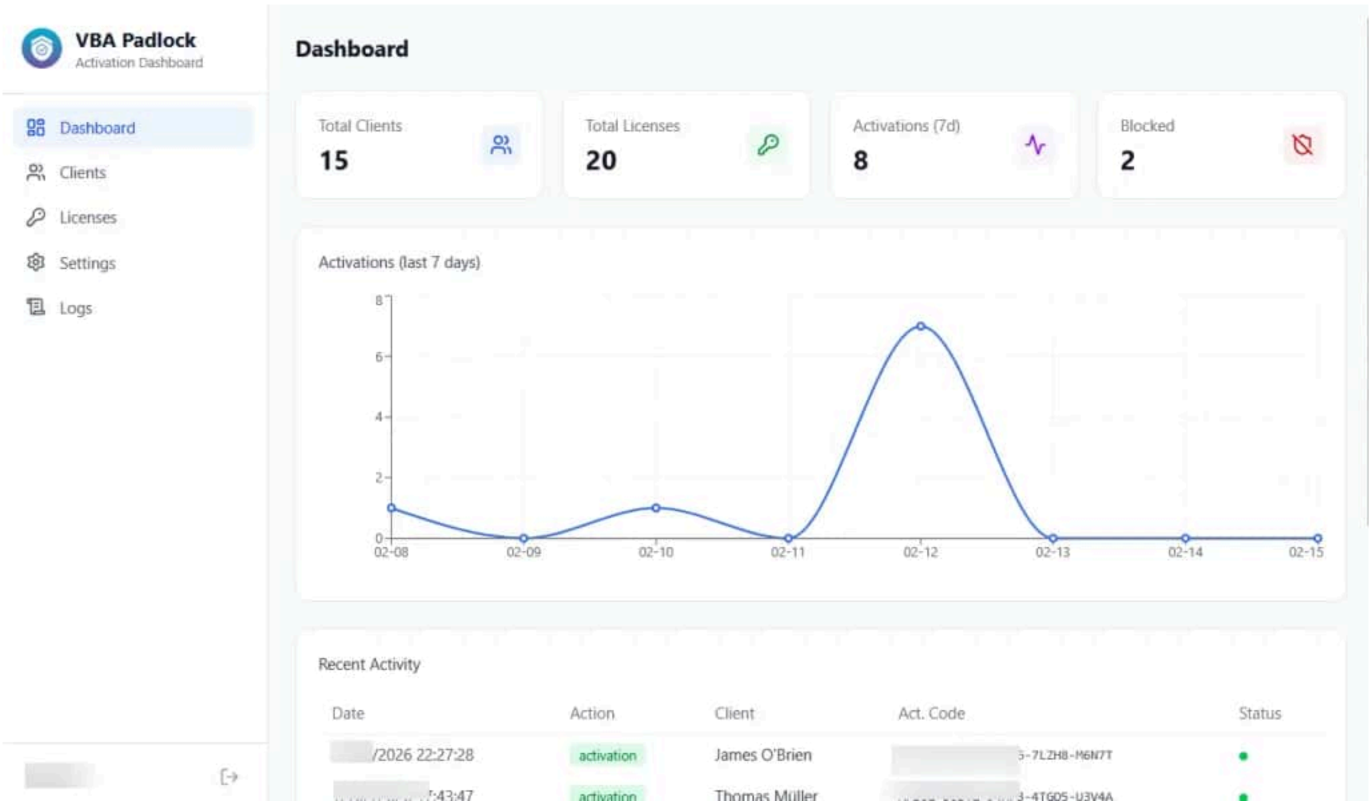
10.2 The Activation Kit “In the Box”

The kit is much more than just an API. It's a full management suite for your software business.

10.2.1 Manage Everything

A modern React-based SPA (Single Page Application) to track your customers.

- **Statistics**: View activation trends (7, 30, 90 days).
- **Client Management**: Create, edit, and search for customers.
- **License Control**: Manually block a license or reset activation counts.
- **Audit Logs**: Every validation and deactivation request is logged.



Dashboard

Total Clients	Total Licenses	Activations (7d)	Blocked
15	20	8	2

Recent Activity

Date	Action	Client	Act. Code	Status
[redacted]/2026 22:27:28	activation	James O'Brien	[redacted]5-7LZH8-M6N7T	●
[redacted]:43:47	activation	Thomas Müller	[redacted]3-4TGD5-U3V4A	●

10.2.2 Sell Instantly

Turn your web server into a store with zero additional coding.

- **Auto-Provisioning**: Licenses are created automatically after successful payment.
- **Email Alerts**: Automatic purchase confirmation for both you and the buyer.
- **IPN Support**: Secure PayPal Instant Payment Notification integration.

10.2.3 Power Your Own Tools

Standard RESTful endpoints if you want to integrate activation into your existing website.

- `POST /getactivation`
- `POST /dovalidation`
- `POST /dodeactivation`
- **Security**: Validates requests using SHA256 checksums and your project's Master Key.

10.3 Technical Requirements

The Activation Kit is lightweight and runs on most standard web hosting environments.

Requirement	Specification
PHP Version	8.1 or higher
Database	MariaDB 10.3+ or MySQL 5.7+
Server	Apache 2.4+ (with `mod_rewrite`)
Security	**Mandatory HTTPS (SSL)**
Extensions	`pdo_mysql`, `sodium`, `curl`, `openssl`

10.4 Core Features

Online Validation

Periodically check if a license is still valid. Allows you to **remote-kill** a license if a payment is charged back.

Self-Service Deactivation

Users can deactivate their machine via the UI to free up a slot for their new computer.

ECC Security

Supports **Long Keys** based on Ed25519 elliptic curve signatures for maximum cryptographic strength.

Offline Fallback

If a user has no internet, they can still perform a manual “Hardware ID” activation as a backup.

10.5 Next Steps

Deployment Guide

Full step-by-step walkthrough on how to install the Activation Kit on your server. View deployment guide →

Online Activation Settings

Configure your VBA Padlock Studio project to point to your new server. View UI reference →

11. License Deactivation & Transfer

Provide your users with the flexibility they expect. When they upgrade their hardware or switch machines, the **Deactivation System** allows them to securely release their license and move it to a new computer without your constant intervention.

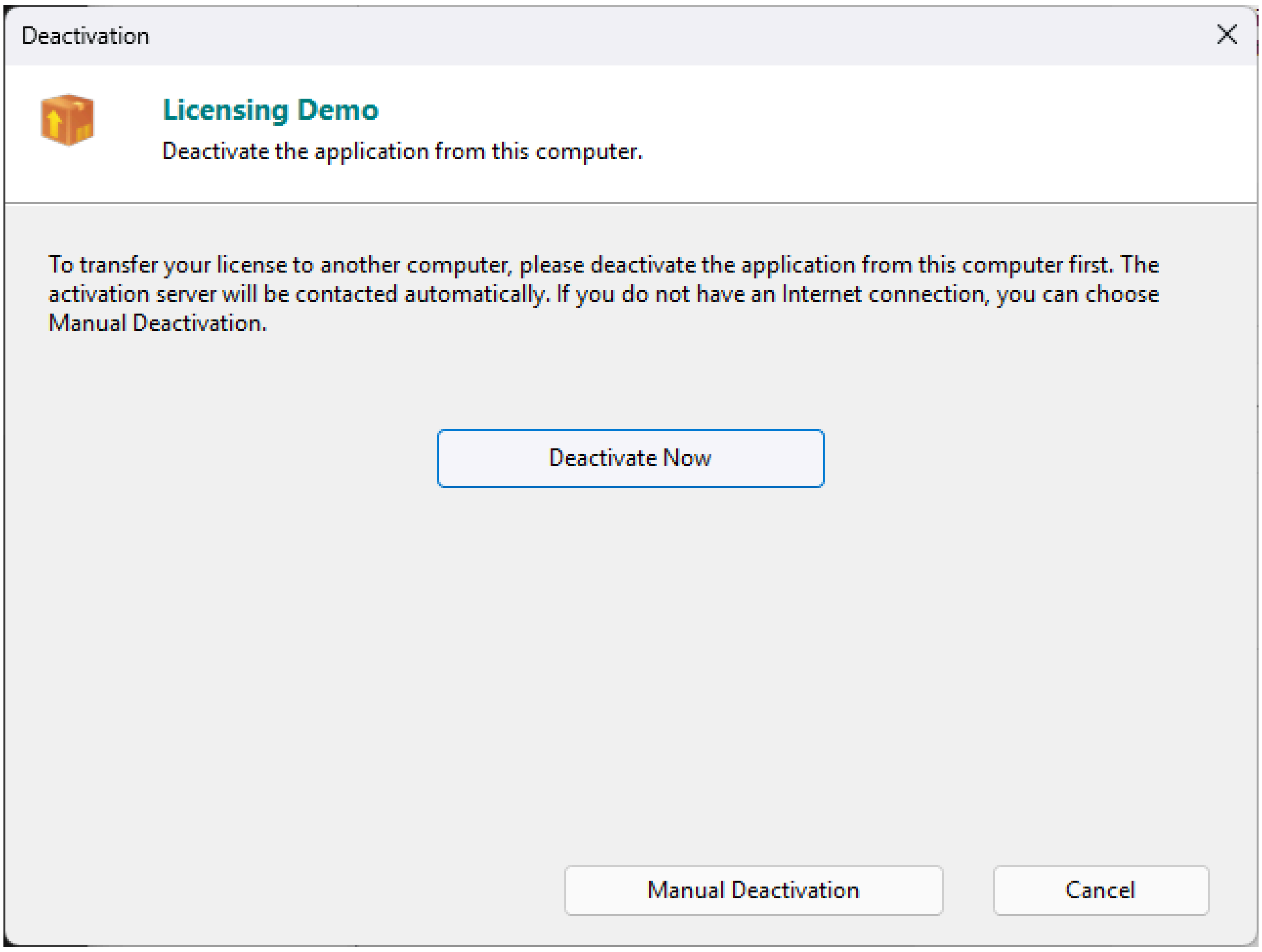
11.1 Deactivation Methods

VBA Padlock supports two ways to handle license transfers.

11.1.1 Zero Maintenance

The most seamless experience. The application contacts your server to release the activation slot instantly.

1. User clicks **Deactivate** in your application.
2. The app sends a secure request to your Activation Server.
3. The server frees the license slot in the database.
4. The user activates on their new computer immediately.


Deactivation

Licensing Demo

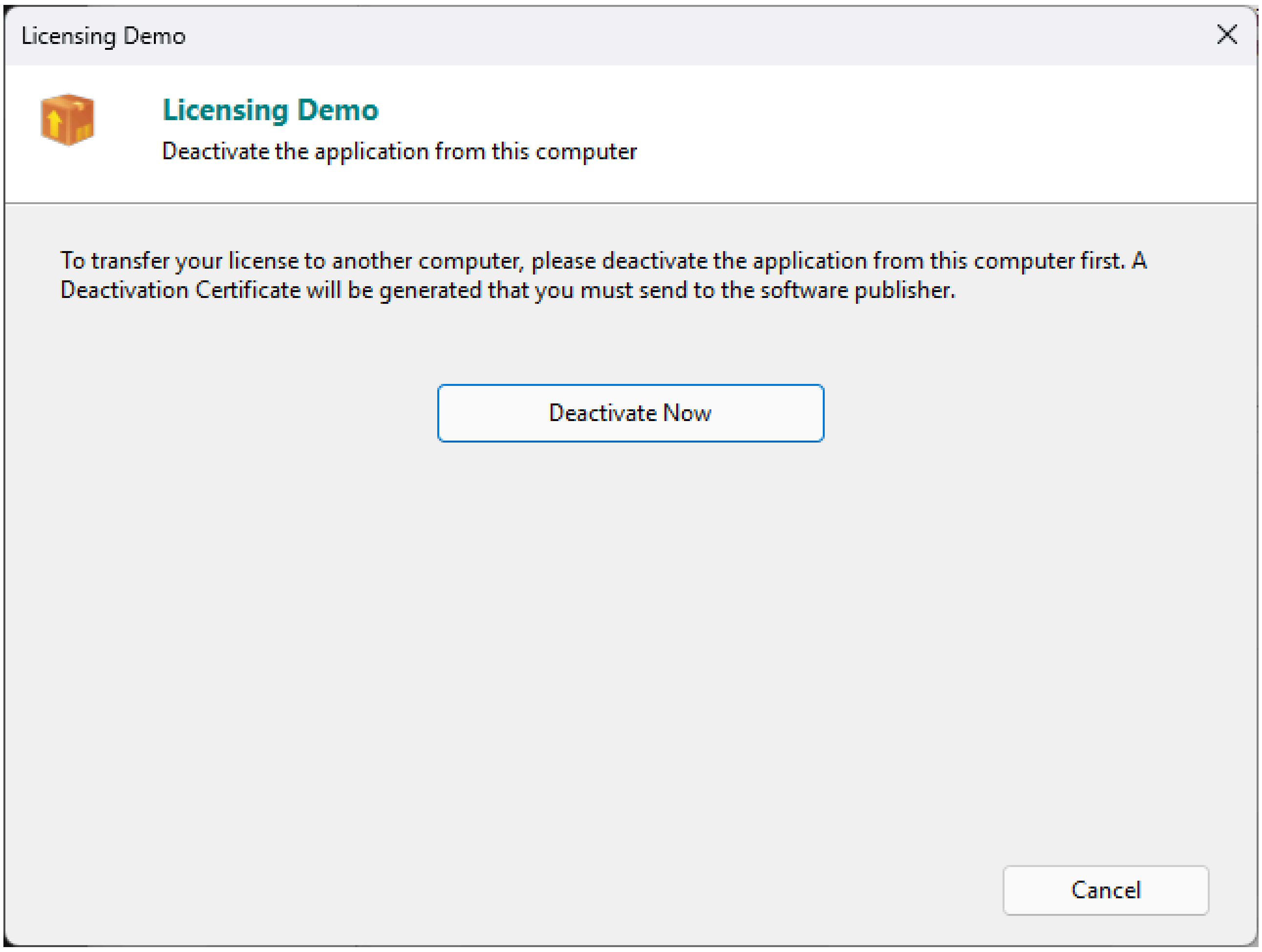
Deactivate the application from this computer.

To transfer your license to another computer, please deactivate the application from this computer first. The activation server will be contacted automatically. If you do not have an Internet connection, you can choose Manual Deactivation.

11.1.2 No Internet Required

Reliable fallback for users working in offline or restricted environments.

1. User clicks **Manual Deactivation**.
2. The app generates a **Deactivation Certificate** (a cryptographic proof of removal).
3. The user sends this text/file to you via email.
4. You verify the certificate and issue a new key for their new Machine ID.



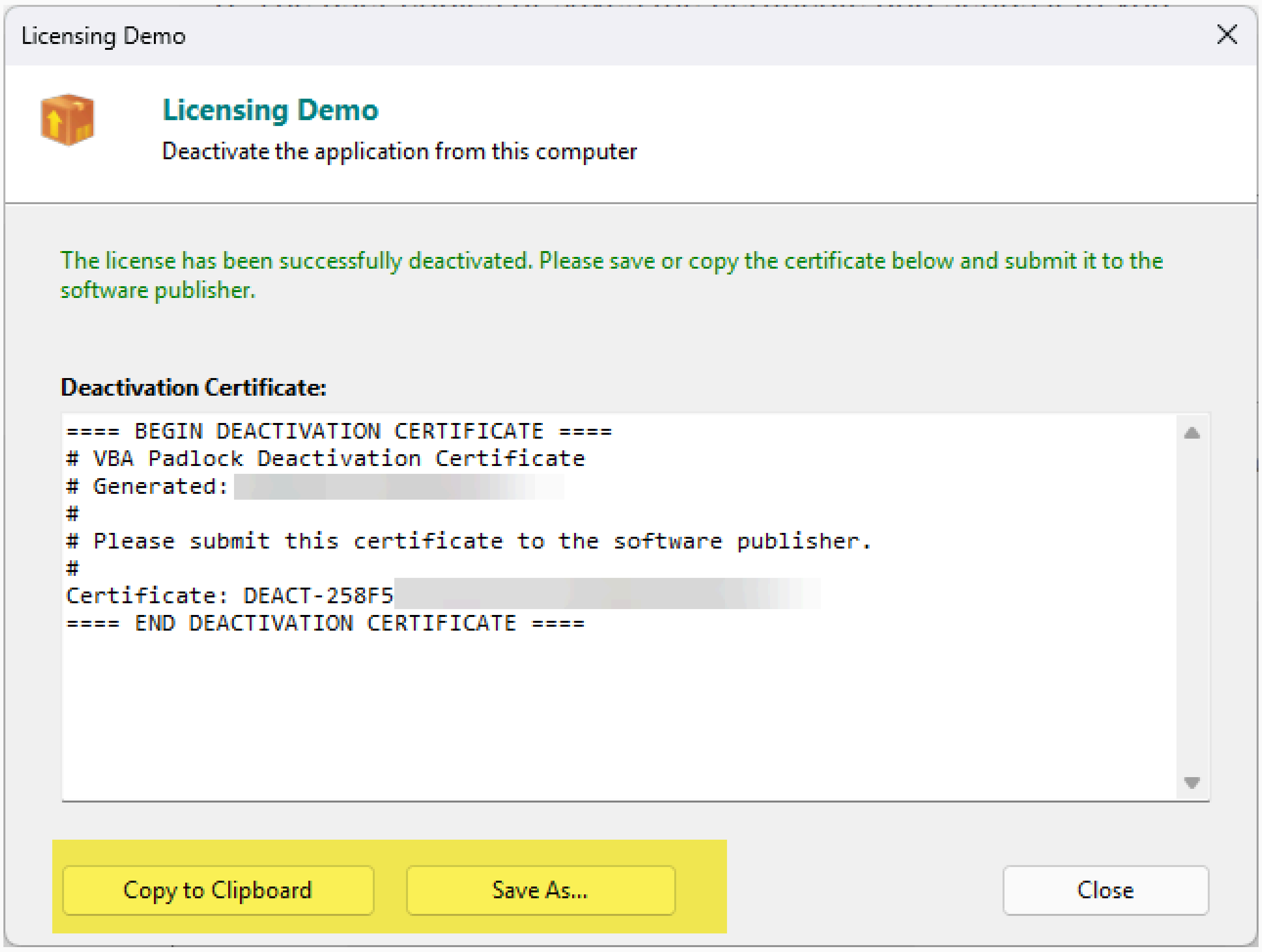
To transfer your license to another computer, please deactivate the application from this computer first. A Deactivation Certificate will be generated that you must send to the software publisher.

11.2 The Deactivation Certificate

Think of the **Deactivation Certificate** as a digital “receipt” that proves the user has actually deleted the license from their computer.

What's Inside?

The certificate is a signed block of text containing the deactivated key, the old Hardware ID, and a secure timestamp to prevent reuse.



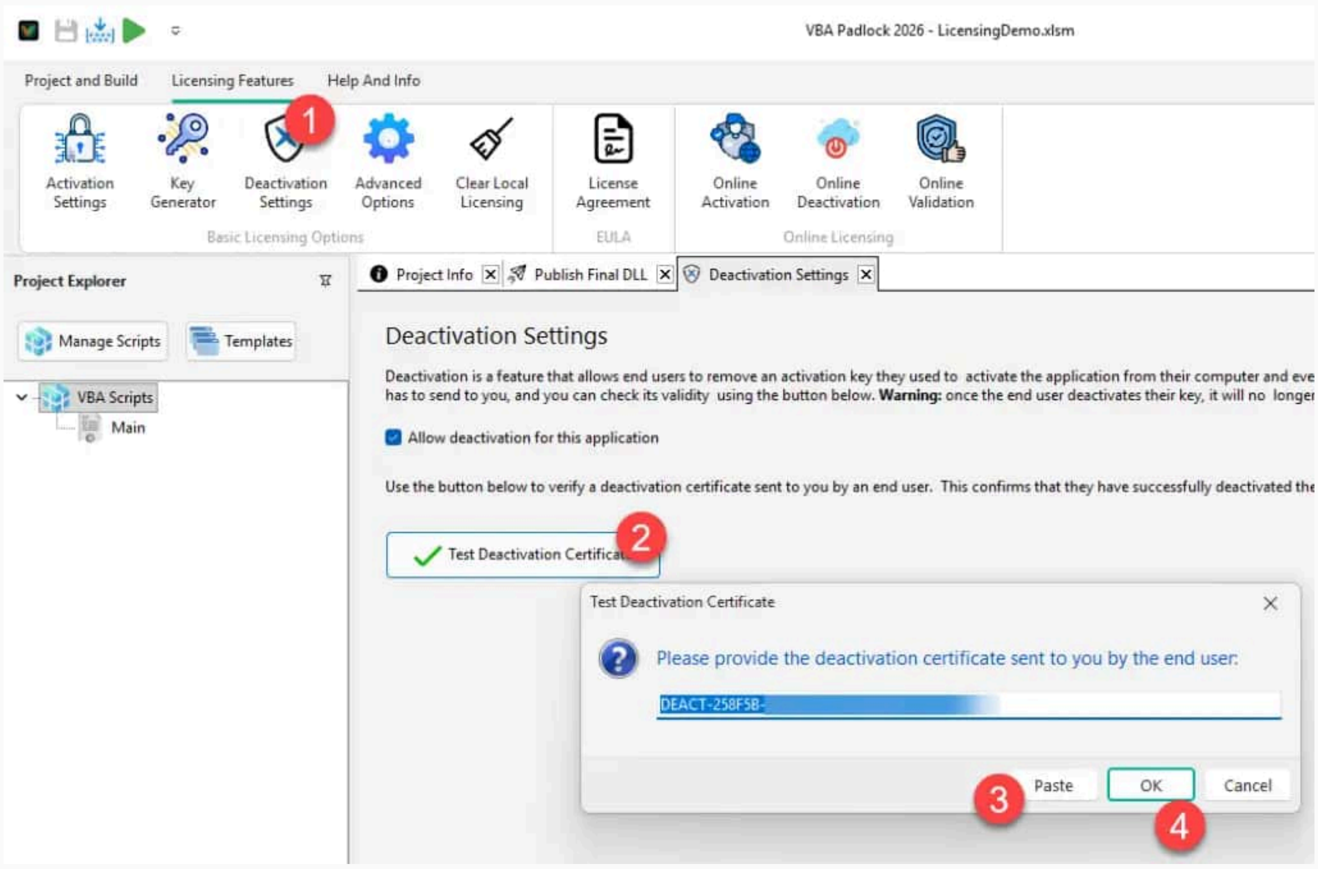
The license has been successfully deactivated. Please save or copy the certificate below and submit it to the software publisher.

Deactivation Certificate:

```
==== BEGIN DEACTIVATION CERTIFICATE ====
# VBA Padlock Deactivation Certificate
# Generated: [redacted]
#
# Please submit this certificate to the software publisher.
#
Certificate: DEACT-258F5[redacted]
==== END DEACTIVATION CERTIFICATE ====
```

How to Verify?

In **VBA Padlock Studio**, use the **Test Certificate** tool to verify the validity of a certificate received from a user before issuing a replacement.



Deactivation Settings

Deactivation is a feature that allows end users to remove an activation key they used to activate the application from their computer and eve has to send to you, and you can check its validity using the button below. **Warning:** once the end user deactivates their key, it will no longer

☑ Allow deactivation for this application

Use the button below to verify a deactivation certificate sent to you by an end user. This confirms that they have successfully deactivated the

Test Deactivation Certificate

Please provide the deactivation certificate sent to you by the end user:

DEACT-258F5B-

11.3 Implementation in VBA

You can create a custom “Transfer License” button in your Excel/Word ribbon that invokes the built-in deactivation dialog.

VBA Bridge Command

```
Public Sub DeactivateMyLicense()
    Dim result As Long
    ' This shows the combined Online/Manual dialog
    result = VBAPL_ShowDeactivation()

    If result = 0 Then
        MsgBox "Deactivation successful! You can now activate on another
computer."
    Else
        MsgBox "Process cancelled or failed."
    End If
End Sub
```

11.4 Professional Workflow

A standard license transfer typically follows these steps:

1. **Old Computer**: User performs deactivation (Online or Manual).
2. **License State**: The activation count is restored (automated on server or manual in your records).
3. **New Computer**: User installs your software and retrieves their new **Hardware ID**.
4. **Activation**: User activates via the server (instant) or you send a new locked key (manual).

11.5 Next Steps

Deactivation Settings

Enable deactivation and test your certificates. View reference →

Online Deactivation

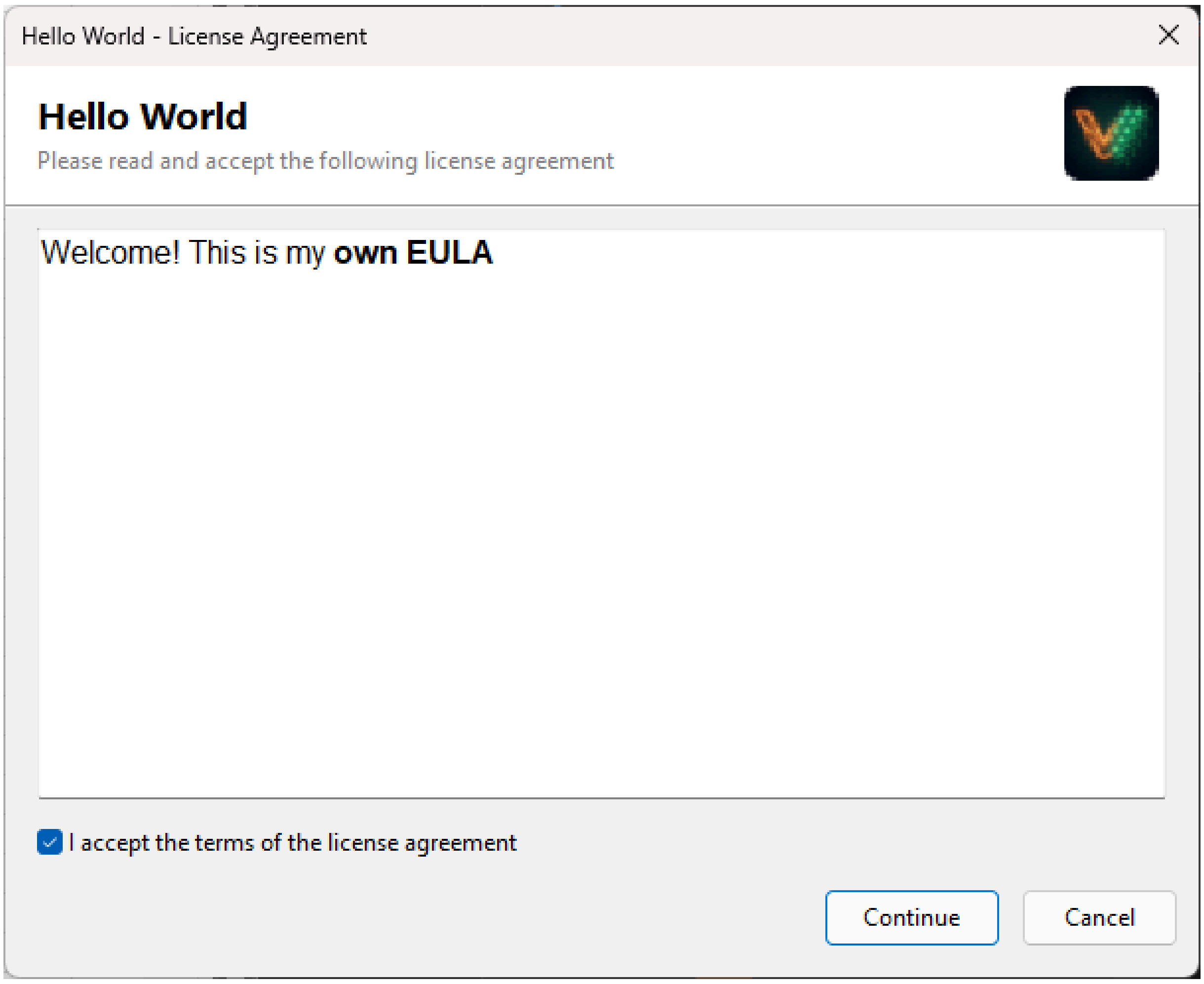
Configure automated, server-side deactivation URLs. Configure online mode →

Hardware Locking

Understand the “Machine ID” that licenses are tied to. Learn more →

12. End-User License Agreement (EULA)

Ensure your users agree to your legal terms before they even see your application's interface. VBA Padlock allows you to embed a professional **License Agreement (EULA)** dialog that acts as a secure gateway to your software.



Hello World - License Agreement

Hello World

Please read and accept the following license agreement

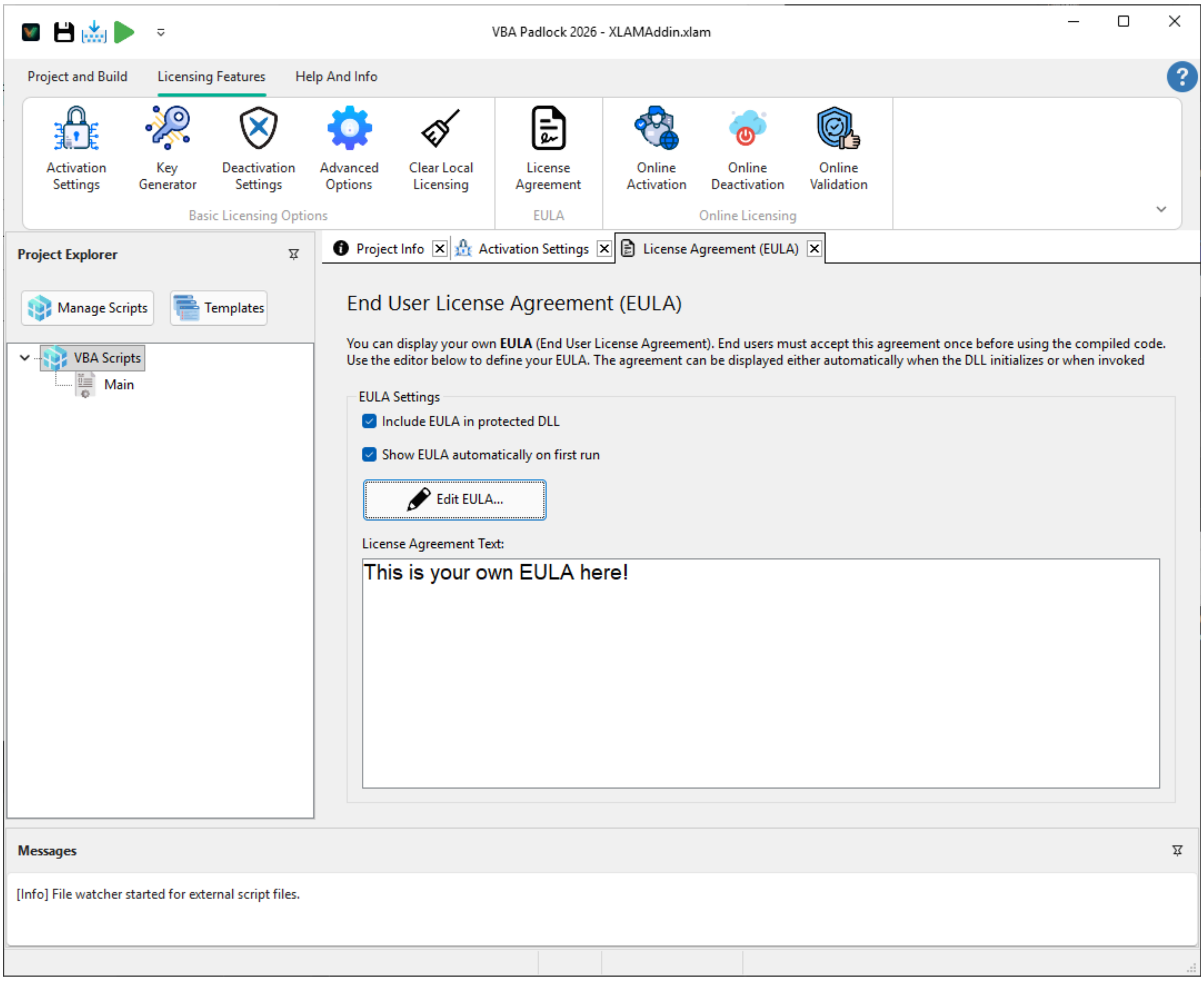
Welcome! This is my **own EULA**

☑ I accept the terms of the license agreement

12.1 Setting Up Your EULA

The EULA is designed to be the very first interaction after a user launches your protected Office file.

Write or paste your agreement directly into the **VBA Padlock Studio** editor.



End User License Agreement (EULA)

You can display your own **EULA** (End User License Agreement). End users must accept this agreement once before using the compiled code. Use the editor below to define your EULA. The agreement can be displayed either automatically when the DLL initializes or when invoked

EULA Settings

- [✓] Include EULA in protected DLL
- [✓] Show EULA automatically on first run

License Agreement Text:

This is your own EULA here!

[Info] File watcher started for external script files.

- **Rich Text Support**: Use bold, italics, bullet points, and links.
- **Import RTF**: Copy-paste formatted text from Microsoft Word flawlessly.

Configure the behavior of the dialog in the **Licensing Features → License Agreement** tab:

1. Check **Include License Agreement (EULA)**.
2. Check **Show EULA on first launch** to automate the process.
3. Set the subtitle and checkbox label (supports localization).

1 User launches the workbook.

2 EULA dialog appears instantly.

3 The **Continue** button remains disabled until the user checks “I accept”.

4 Acceptance is stored locally; the user is never asked again unless the license is reset.

12.2 Integration with Your Code

You can programmatically check if the user has accepted the terms to enable or disable specific features within your application.

Compiled Script (DLL)

```
' Inside VBA Padlock Studio
If IsEulaAccepted() Then
    ' Continue with sensitive logic
End If
```

Office VBA (Bridge)

```
' In your Excel/Word module
If Not VBAPL_IsEulaAccepted() Then
    MsgBox "Please accept the EULA first."
End If
```

12.3 Customizable Text Strings

The EULA dialog is fully localizable. You can change these labels in your JSON locale file.

Key	Default Label (English)
`eula.dialogTitle`	License Agreement
`eula.labelSubtitle`	Please read and accept the following license agreement
`eula.checkAccept`	I accept the terms of the license agreement
`eula.btnContinue`	Continue

12.4 Best Practices

- **Clarity over Legalese**: Use clear headings and bullet points to make the terms readable.
- **Combined Protection**: The EULA is shown *before* the activation dialog, ensuring users accept your terms before they even attempt to activate the software.
- **Test Before Release**: Use the Test Runner to verify the formatting of your RTF text.

Note

The EULA dialog is independent of the licensing system. You can use it even if you are not using hardware locking or activation keys.

12.5 Next Steps

Localization

Translate your EULA buttons and labels into any language. Learn how →

Licensing System

Learn about hardware locking and license key generation. View overview →

Trial Mode

Allow users to try your software before they subscribe. Configure Trial →

13. Localization

Expand your reach to international markets with ease. VBA Padlock features a robust **JSON-based localization system** that allows you to translate every runtime interaction — from activation dialogs to error messages — into any language.

13.1 The Localization Workflow

Translating your application follows a straightforward logic: copy, translate, and embed.

VBA Padlock projects start with a default English file (`locale_en-US.json`).

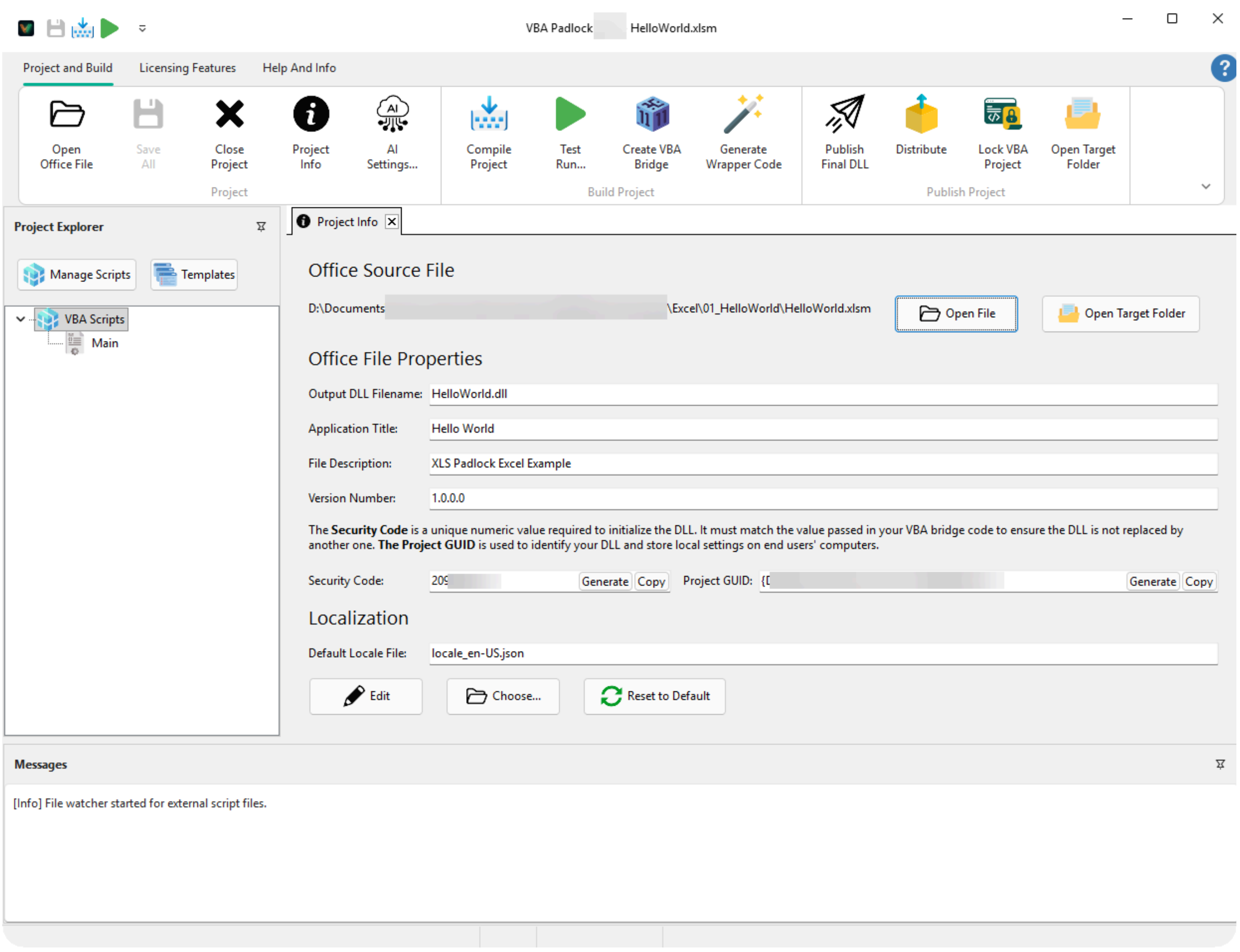
1. Copy the default file in your project folder.
2. Rename it using the target locale code (e.g., `locale_fr-FR.json`).
3. Translate the values in the JSON file.

AI-Assisted Translation

Since locale files are standard JSON, you can use **AI Agents** (Claude, Gemini, ChatGPT) to translate your entire project in seconds. Simply provide the JSON to the AI and ask for a translation while preserving the keys and placeholders (`%s`, `%d`).

Connect your new locale file to your project in **VBA Padlock Studio**:

1. Navigate to **Project and Build → Project Info**.
2. Select your JSON file in the **Locale File** field.
3. At compile time, this file is securely embedded into your satellite DLL.



Office Source File

D:\Documents \Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description: XLS Padlock Excel Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 209 Project GUID: {[redacted]

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

The primary language is embedded inside the DLL, so you don't need to ship the JSON file with your workbook.

```
▼ MyWorkbook.vbapadlock/   ← Project Source
      Main.bas
      { } locale_en-US.json
      { } locale_fr-FR.json
  MyWorkbook.xlsm   ← Compiled Result
▼ bin/
      MyWorkbook.dll   ← Contains embedded locale
```

13.2 Working with Locale Files

Locale files use a flat key-value structure. Here is a simplified example for a French translation:

```json
{
  "locale": "fr-FR",
  "strings": {
    "common": {
      "close": "Fermer",
      "activate": "Activer"
    },
    "trial": {
      "dialogTitle": "Version d'évaluation",
      "dayRemaining_one": "1 jour restant",
      "dayRemaining_other": "%d jours restants"
    }
  }
}
```

13.2.1 Placeholders & Plurals

To maintain dynamic logic, you must preserve placeholders:

- `%s`: String placeholder (e.g., `"Script: %s"`)
- `%d`: Number placeholder (e.g., `"%d days"`)
- `_one` / `_other`: Suffixes used for automatic pluralization based on a count.

13.3 Programming with Locales

Inside your compiled scripts, you can access the localization engine directly to display custom messages in the user's language.

Get Basic String

```
' Get "activate" label
Dim label
label = L("common.activate")
```

Formatted String

```
' "14 days remaining"
Dim msg
msg = LFmt("trial.dayRemaining", 14)
```

Plural Forms

```
' Handles _one/_other automatically
Dim msg
msg = LPlural("trial.dayRemaining", count)
```

Runtime Loading

```
' Switch language on the fly
LoadLocale(jsonContent)
```

13.4 Best Practices

1. **Keep Keys Intact**: Never modify the keys (left side); only translate the values (right side).
2. **Preserve Space & Case**: Ensure that spaces around placeholders and the casing of labels match your UI design.
3. **Internal vs External**: Use the embedded locale for the primary language, and use `LoadLocale` in your VBA code if you want to support users switching languages dynamically.

13.5 Next Steps

EULA

Localize your License Agreement dialog. View EULA details →

Trial Mode

Translate trial status and countdown messages. View Trial Mode →

UI Reference

Explore the Project Info dialog where locales are set. View reference →

14. Protecting Your First Project

This guide walks you through creating your first VBA Padlock project. You'll learn the complete workflow from opening an Office file to distributing a protected application.

Tip

Before you start, make sure you have:

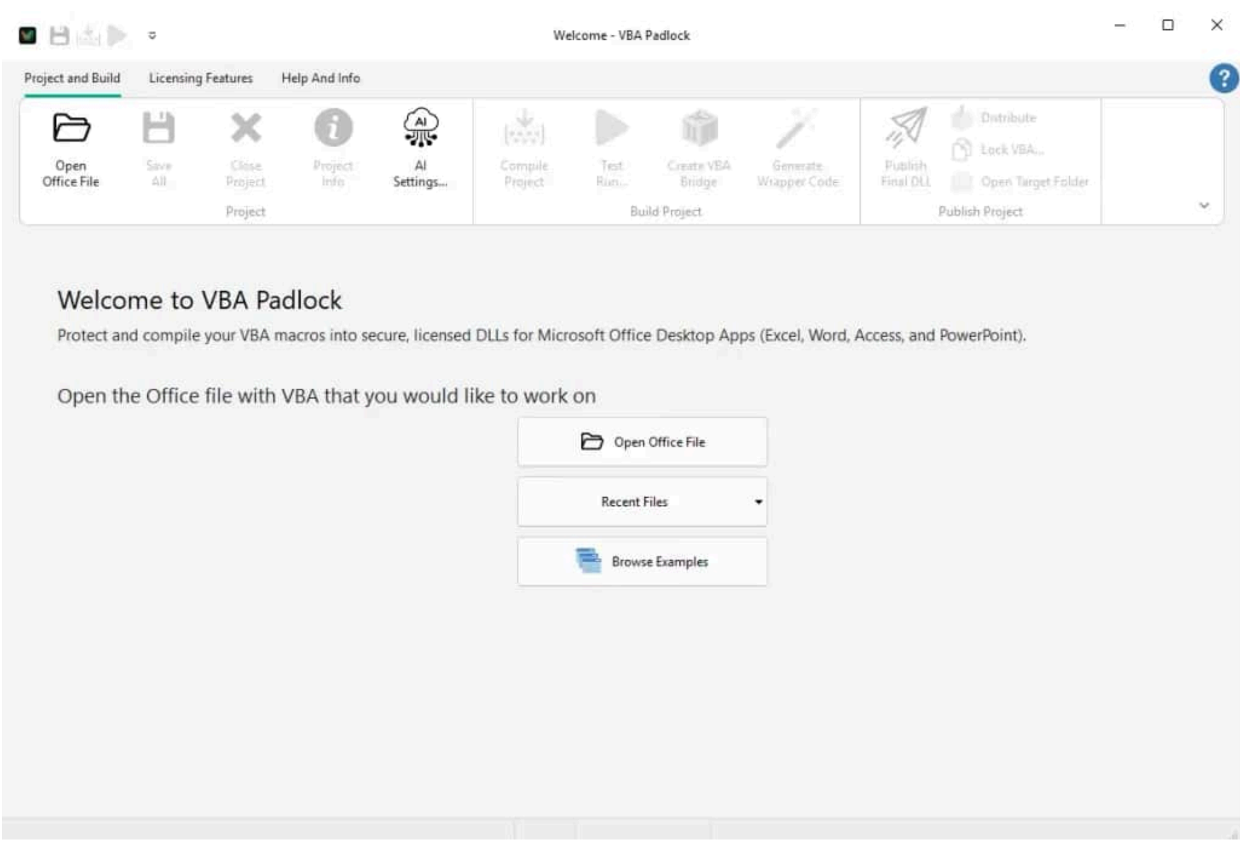
- VBA Padlock installed (installation guide)
- A macro-enabled Office file (`.xlsm`, `.docm`, `.accdb`, or `.pptm`)

14.1 Core Workflow

Follow these steps to transform your VBA code into a secure, compiled DLL.

1 **Launch VBA Padlock**

Open VBA Padlock from the Start menu. You will be greeted by the **Welcome Screen**.



Welcome to VBA Padlock

Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

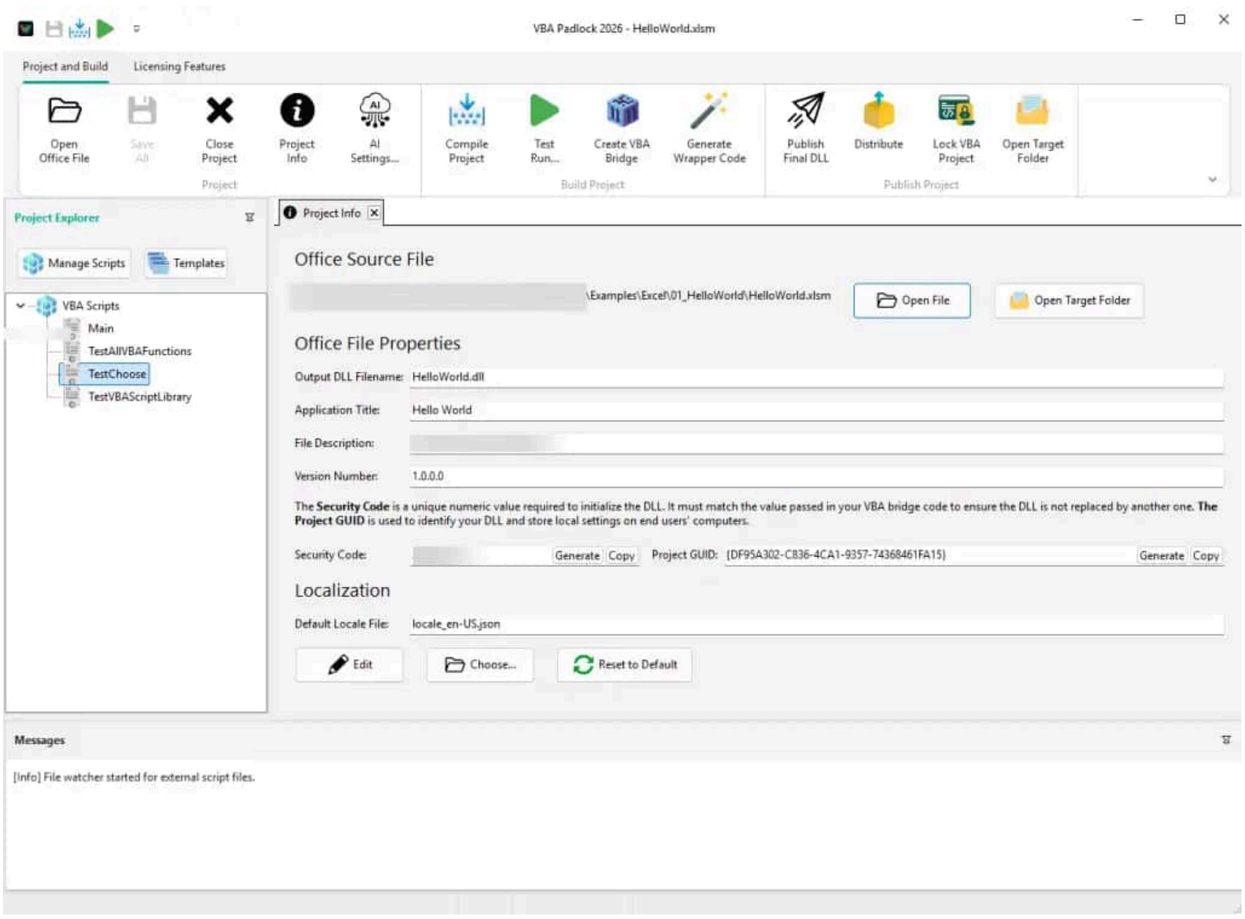
2 Open your Office file

Click **Open Office File** and select your macro-enabled file (example: `SimpleTest.xlsm`). VBA Padlock creates a `.vbapadlock` project folder alongside your file to store project settings and scripts.

3 Set Project Information

In the **Project Info** tab, define your application's identity.

- **Title:** The name of your tool.
- **Security Code:** Click **Generate** to create a unique cryptographic link between your Office file and the DLL.



Office Source File

\Examples\Excel\01_HelloWorld\HelloWorld.xlsm [Open File] [Open Target Folder]

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description:

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: [Generate] [Copy] Project GUID: {DF95A302-C836-4CA1-9357-74368461FA15} [Generate] [Copy]

Localization

Default Locale File: locale_en-US.json

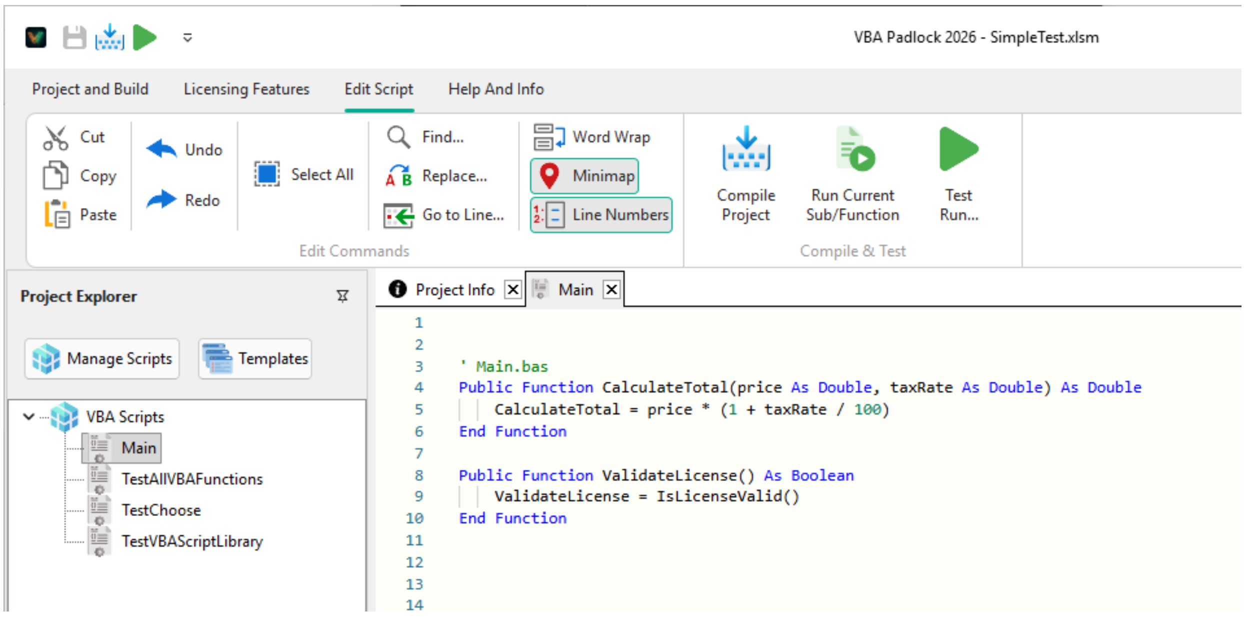
[Edit] [Choose...] [Reset to Default]

Messages

```
[Info] File watcher started for external script files.
```

4 Write your Protected Scripts

Navigate to the **Edit Script** tab. This is where you write the sensitive logic that will be compiled and hidden from users.



```


' Main.bas
Public Function CalculateTotal(price As Double, taxRate As Double) As Double
    CalculateTotal = price * (1 + taxRate / 100)
End Function

Public Function ValidateLicense() As Boolean
    ValidateLicense = IsLicenseValid()
End Function
```

Use the following example code for your first test:

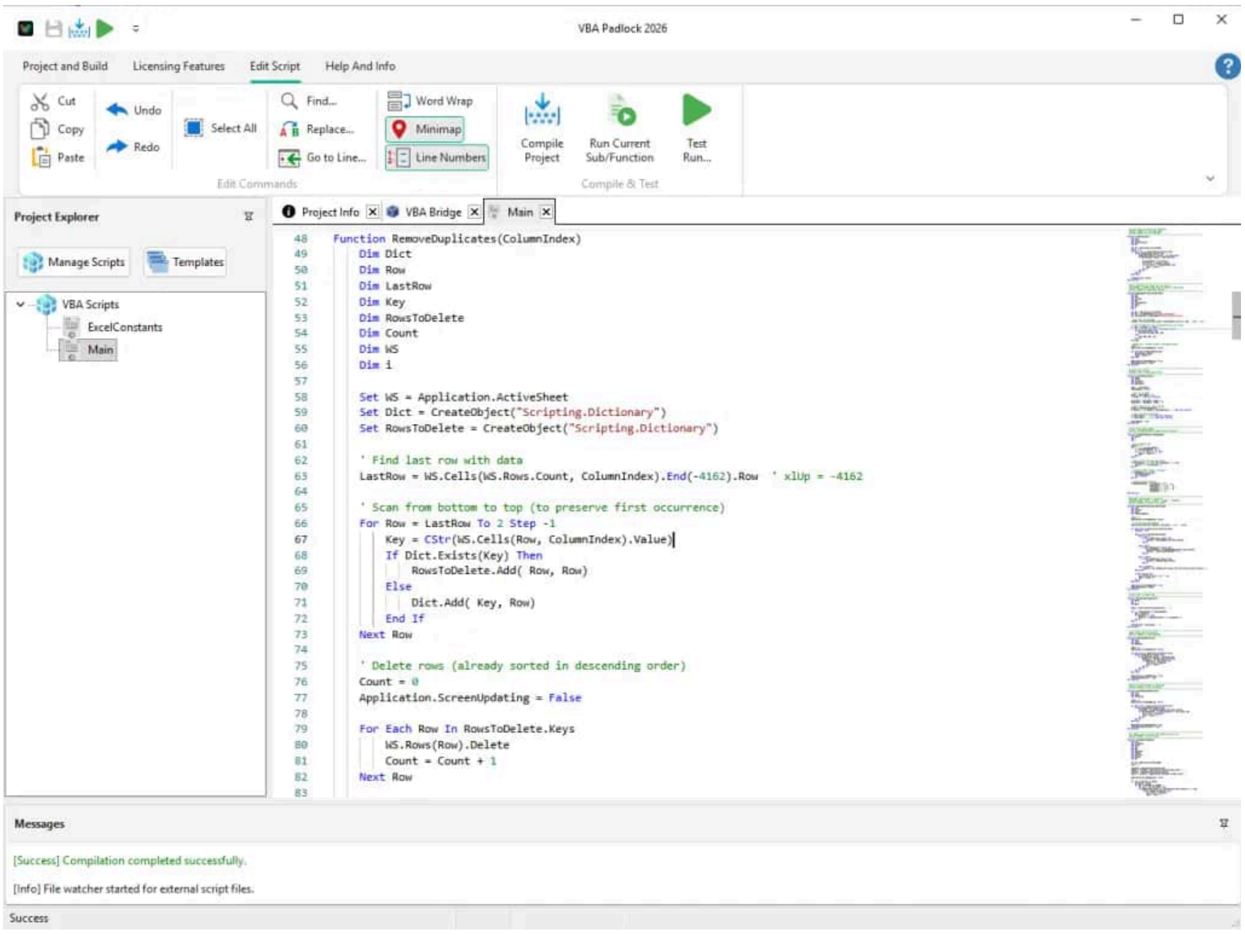
```
' Main.bas
Public Function CalculateTotal(price As Double, taxRate As Double) As
Double
    CalculateTotal = price * (1 + taxRate / 100)
End Function

Public Function ValidateLicense() As Boolean
    ValidateLicense = IsLicenseValid()
End Function
```

Tip

Compiled scripts have full access to the host application through the `Application` object. See Office Object Model Access.

5 Compile the DLL

Press `F5` or click **Compile Project**. VBA Padlock transforms your script into a high-performance, encrypted DLL.



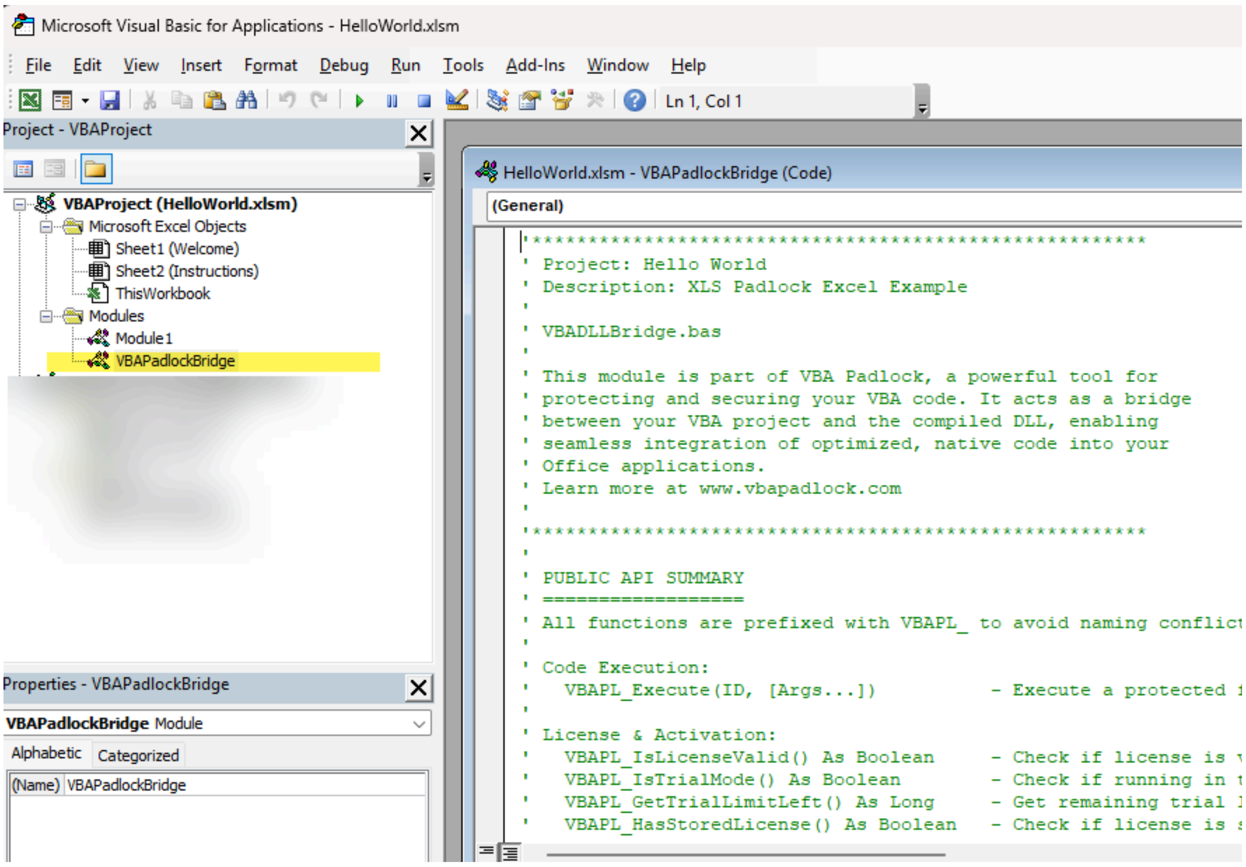
```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Dim Row
    Dim LastRow
    Dim Key
    Dim RowsToDelete
    Dim Count
    Dim WS
    Dim i

    Set WS = Application.ActiveSheet
    Set Dict = CreateObject("Scripting.Dictionary")
    Set RowsToDelete = CreateObject("Scripting.Dictionary")

    ' Find last row with data
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp = -4162

    ' Scan from bottom to top (to preserve first occurrence)
    For Row = LastRow To 2 Step -1
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Dict.Exists(Key) Then
            RowsToDelete.Add( Row, Row)
        Else
            Dict.Add( Key, Row)
        End If
    Next Row

    ' Delete rows (already sorted in descending order)
    Count = 0
    Application.ScreenUpdating = False

    For Each Row In RowsToDelete.Keys
        WS.Rows(Row).Delete
        Count = Count + 1
    Next Row

```

```
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

6 Inject the VBA Bridge

To call your compiled functions from Excel (or Word/Access), you need a “Bridge”.

- Click **Create VBA Bridge → Inject Into Office**.
- This adds a module named `VBAPadlockBridge` to your Office file.


Microsoft Visual Basic for Applications - HelloWorld.xlsm
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - VBAProject
VBAProject (HelloWorld.xlsm)
Microsoft Excel Objects
Sheet1 (Welcome)
Sheet2 (Instructions)
ThisWorkbook
Modules
Module1
VBAPadlockBridge
Properties - VBAPadlockBridge
VBAPadlockBridge Module
Alphabetic
Categorized
(Name) VBAPadlockBridge
HelloWorld.xlsm - VBAPadlockBridge (Code)
(General)

```
'*******************************************************
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' VBADLLBridge.bas
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*******************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflict
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])           - Execute a protected f
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean      - Check if license is v
'   VBAPL_IsTrialMode() As Boolean         - Check if running in t
'   VBAPL_GetTrialLimitLeft() As Long      - Get remaining trial l
'   VBAPL_HasStoredLicense() As Boolean    - Check if license is s
```

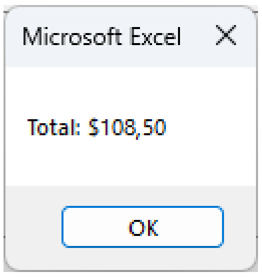
7

Test the Protection

Open your Excel file, go to the VBA Editor (`Alt+F11`), and create a new macro to call your compiled code using `VBAPL_Execute`:

```
Sub TestProtectedCode()
    Dim total As Variant
    ' Calling the compiled function "CalculateTotal" from the "Main"
module
    total = VBAPL_Execute("Main|CalculateTotal", 100, 8.5)

    MsgBox "Total: $" & Format(total, "0.00")
End Sub
```

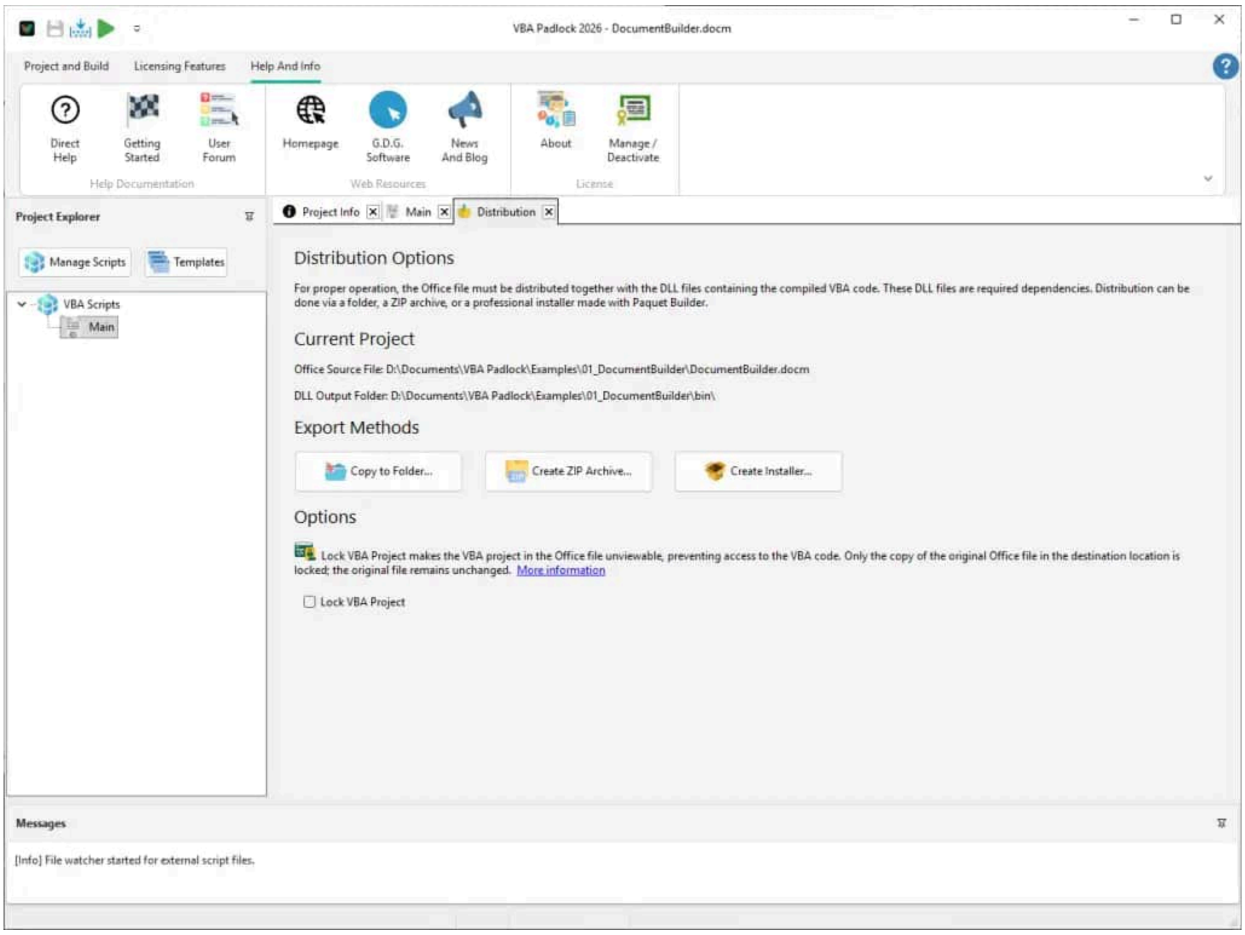
When you run this macro, Excel executes the code **inside the DLL**, keeping your logic completely hidden.



Total: $108,50

8 **Distribution**

When you are ready to share your work, go to the **Distribution** tab. Your application now consists of your Office file and the `bin` folder containing the compiled DLLs.



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

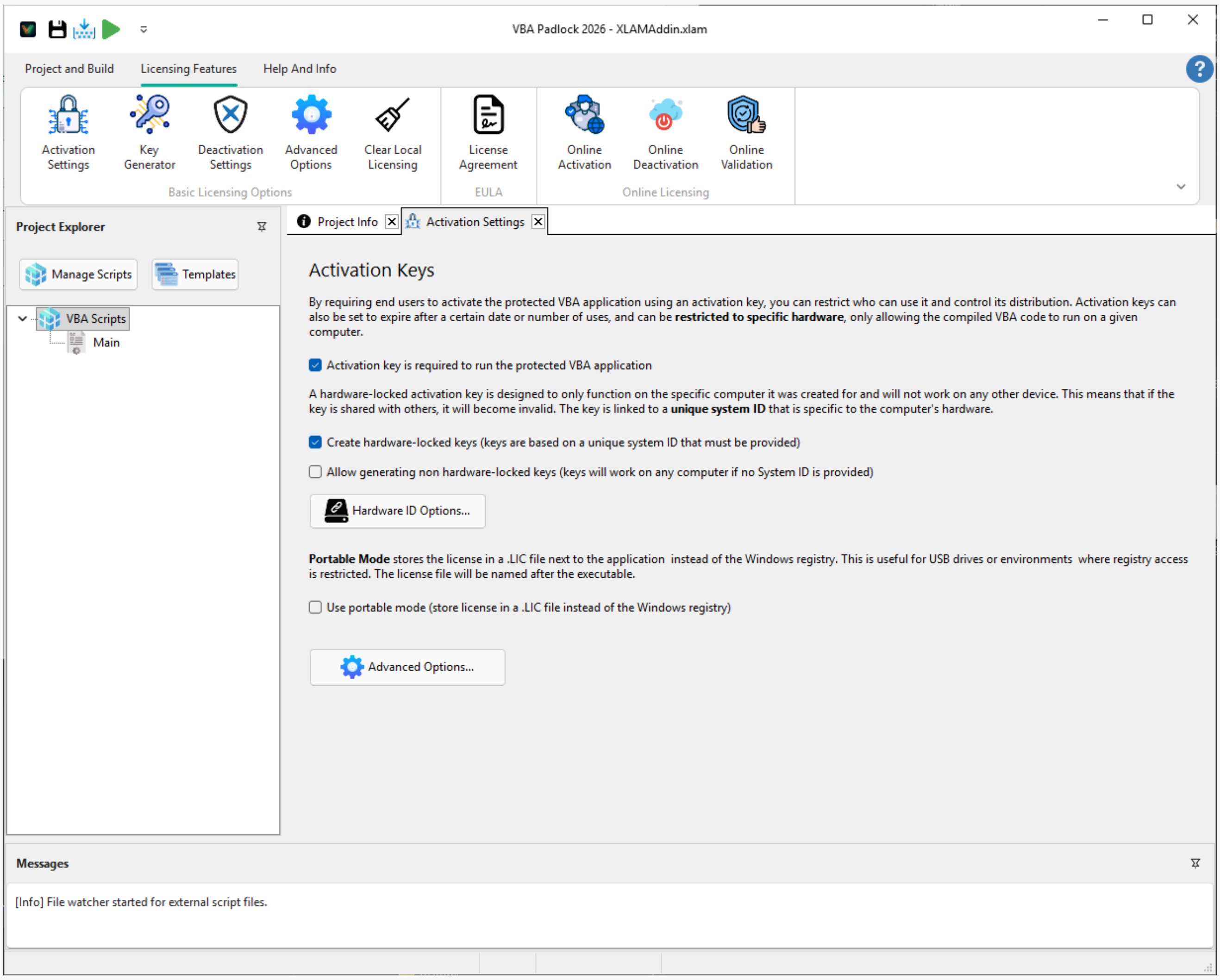
[Info] File watcher started for external script files.

14.2 Enhancing Security (Optional)

You can easily add licensing and trial restrictions to your project:

Activation Required

Require a license key to run the code.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

- [✓] Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

- [✓] Create hardware-locked keys (keys are based on a unique system ID that must be provided)
- [ ] Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

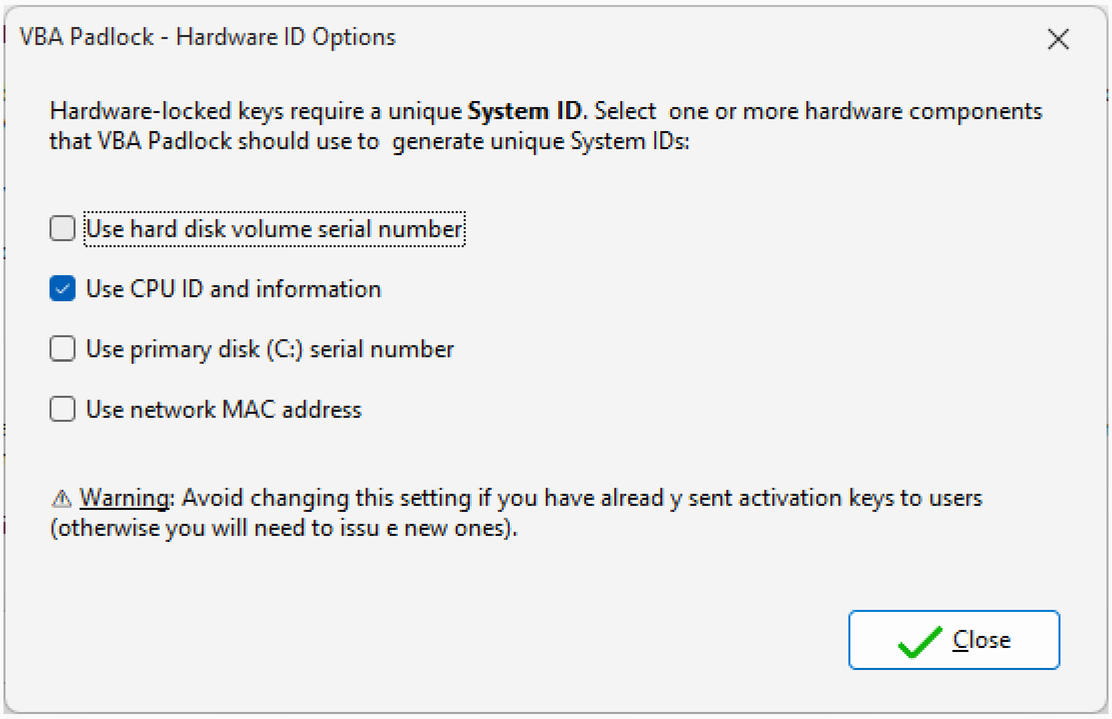
Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

- [ ] Use portable mode (store license in a .LIC file instead of the Windows registry)

[Info] File watcher started for external script files.

Hardware Locking

Lock the license to a specific computer.



VBA Padlock - Hardware ID Options

Hardware-locked keys require a unique **System ID**. Select one or more hardware components that VBA Padlock should use to generate unique System IDs:

- [ ] Use hard disk volume serial number
- [✓] Use CPU ID and information
- [ ] Use primary disk (C:) serial number
- [ ] Use network MAC address

⚠ Warning: Avoid changing this setting if you have alread y sent activation keys to users (otherwise you will need to issu e new ones).

14.3 Next Steps

Licensing Setup

Learn how to configure trial periods and generate license keys. Read the guide →

VBA Bridge API

Explore all available `VBAPL_*` functions for your projects. View reference →

VBA Compatibility

Check which VBA features and statements are supported. View reference →

15. Protect an Excel Workbook

You've built a powerful Excel application with VBA macros, and now you want to distribute it without exposing your source code. In this tutorial, you'll walk through the complete workflow: compiling your VBA logic into a protected DLL, wiring it up to Excel through the auto-generated VBA Bridge, and optionally adding activation keys with hardware locking.

What you'll build:

- A compiled VBA DLL that replaces your original macros
- A VBA Bridge that lets your Excel workbook call the protected code
- Caller macros in Excel that invoke your compiled functions
- (Optional) An activation system with hardware-locked license keys

Time needed: 15-20 minutes

Tip

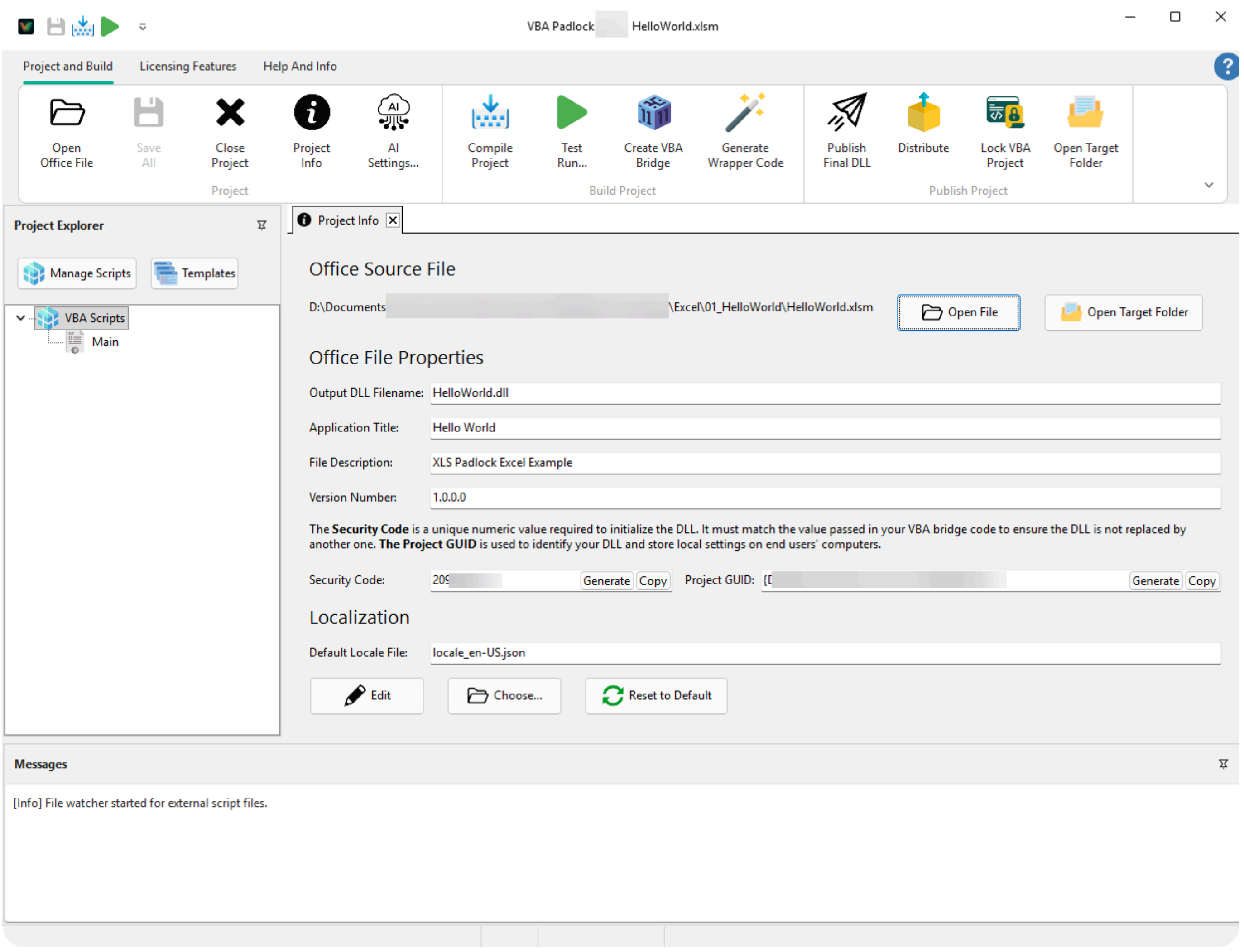
Before you start, make sure you have: - VBA Padlock installed - An Excel workbook or add-in with VBA macros (`.xlsm` or `.xlam` file), or a blank `.xlsm` to start fresh - Basic understanding of VBA and Excel macros

---

15.1 Step 1: Create a VBA Padlock Project

Start by opening your Excel file in VBA Padlock. This creates a project that will hold your compiled code, licensing settings, and build configuration.

1. **Launch VBA Padlock**
2. **Click "Open Office File"** in the ribbon (or press `Ctrl+O`)
3. **Select your Excel workbook or add-in** (`.xlsm` or `.xlam` file) — for this tutorial, we'll use a file called `MyExcelApp.xlsm`
4. VBA Padlock analyzes your file and displays the **Project Info** panel



Office Source File

D:\Documents[redacted]\Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description: XLS Padlock Excel Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 209[redacted] Project GUID: {[redacted]

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

15.1.1 Configure Project Settings

In the **Project Info** panel, fill in these fields:

Field	Example Value	Purpose
DLL Name	`MyExcelApp`	Base name for all output DLL files
Application Title	`My Excel Application`	Shown in activation dialogs and title bars
Version Number	`1.0.0.0`	Your application's version
File Description	`Protected Excel VBA Macros`	Appears in Windows file properties

Tip

The **Security Code** and **Project GUID** are generated automatically by VBA Padlock. You don't need to change them, but keep note of them — they uniquely identify your project and are embedded into every license key.

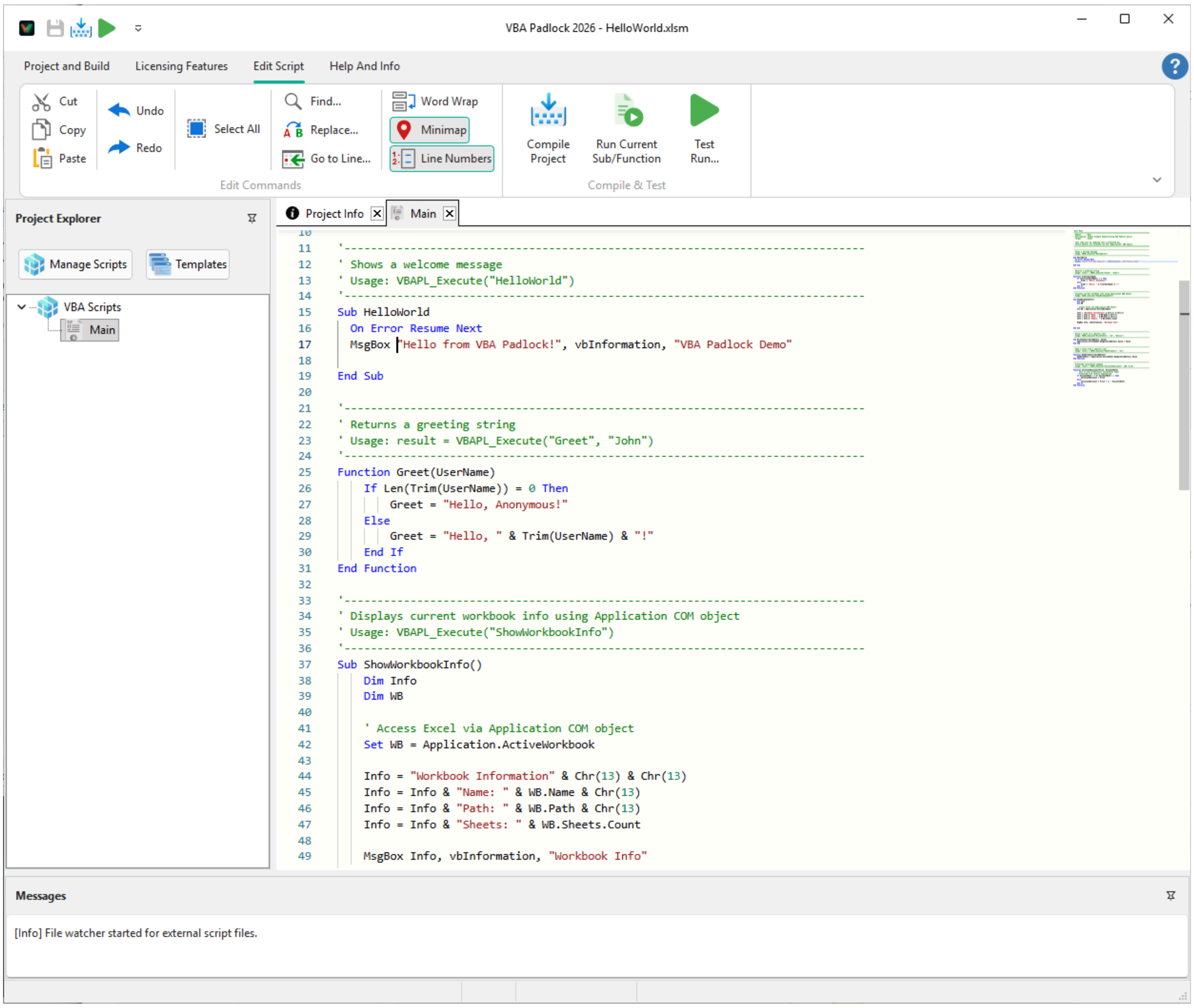
15.2 Step 2: Write Your Protected Code

Now switch to the **Code Editor** in VBA Padlock. This is where you write the VBA code that will be compiled into the DLL. The code you write here will no longer be visible as plain-text VBA — it runs inside the protected DLL.

> **Tip**
>
> Compiled scripts have full access to the host application through the `Application` object. See Office Object Model Access.

Here is an example script that demonstrates the key patterns — subs, functions with parameters, reading/writing cells, and interacting with the workbook:

```vb
Sub HelloWorld()
    On Error Resume Next
    MsgBox "Hello from VBA Padlock!", vbInformation, "MyExcelApp"
End Sub

Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

Sub ShowWorkbookInfo()
    Dim Info, WB
    Set WB = Application.ActiveWorkbook
    Info = "Workbook Information" & Chr(13) & Chr(13)
    Info = Info & "Name: " & WB.Name & Chr(13)
    Info = Info & "Path: " & WB.Path & Chr(13)
    Info = Info & "Sheets: " & WB.Sheets.Count
    MsgBox Info, vbInformation, "Workbook Info"
End Sub

Sub WriteToCell(CellAddress, Value)
    Application.ActiveSheet.Range(CellAddress).Value = Value
End Sub

Function ReadFromCell(CellAddress)
    ReadFromCell = Application.ActiveSheet.Range(CellAddress).Value
End Function

Function CalculateDiscount(Price, DiscountRate)
    If DiscountRate < 0 Or DiscountRate > 1 Then
        CalculateDiscount = Price
    Else
        CalculateDiscount = Price * (1 - DiscountRate)
    End If
End Function
```



```
'-------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'-------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'-------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'-------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'-------------------------------------------------------------------------------
' Displays current workbook info using Application COM object
' Usage: VBAPL_Execute("ShowWorkbookInfo")
'-------------------------------------------------------------------------------
Sub ShowWorkbookInfo()
    Dim Info
    Dim WB

    ' Access Excel via Application COM object
    Set WB = Application.ActiveWorkbook

    Info = "Workbook Information" & Chr(13) & Chr(13)
    Info = Info & "Name: " & WB.Name & Chr(13)
    Info = Info & "Path: " & WB.Path & Chr(13)
    Info = Info & "Sheets: " & WB.Sheets.Count

    MsgBox Info, vbInformation, "Workbook Info"
```

```
[Info] File watcher started for external script files.
```

Type or paste this code into the VBA Padlock Code Editor. You can adapt it to your own application — the important thing is that each `Sub` or `Function` you define here becomes callable from Excel through `VBAPL_Execute`.

Tip

Compiled scripts have full access to the host application through the `Application` object. See Office Object Model Access.

15.3 Step 3: Compile and Publish

Time to build the protected DLL.

1. **Click "Publish Final DLL"** in the ribbon (Project and Build tab)
2. **Click "Build Final DLL Files"** in the Publish dialog

3. VBA Padlock will:
   - Parse your VBA code
   - Compile it into protected bytecode
   - Embed it into a protected DLL
   - Generate the runtime DLLs
4. When the build completes, you'll see a success message. Three DLL files are created in the `bin\` subfolder next to your Excel workbook:

```
MyExcelApp\
├── MyExcelApp.xlsm
└── bin\
    ├── MyExcelApprun32.dll   (32-bit runtime)
    ├── MyExcelApprun64.dll   (64-bit runtime)
    └── MyExcelApp.dll        (your compiled code - satellite DLL)
```

Note

The runtime DLL files are digitally signed by G.D.G. Software using Authenticode certificates, so Windows SmartScreen and antivirus software recognize them as legitimate. The satellite DLL is integrity-verified by an internal cryptographic signature to prevent tampering. You can optionally sign it with your own Authenticode certificate.

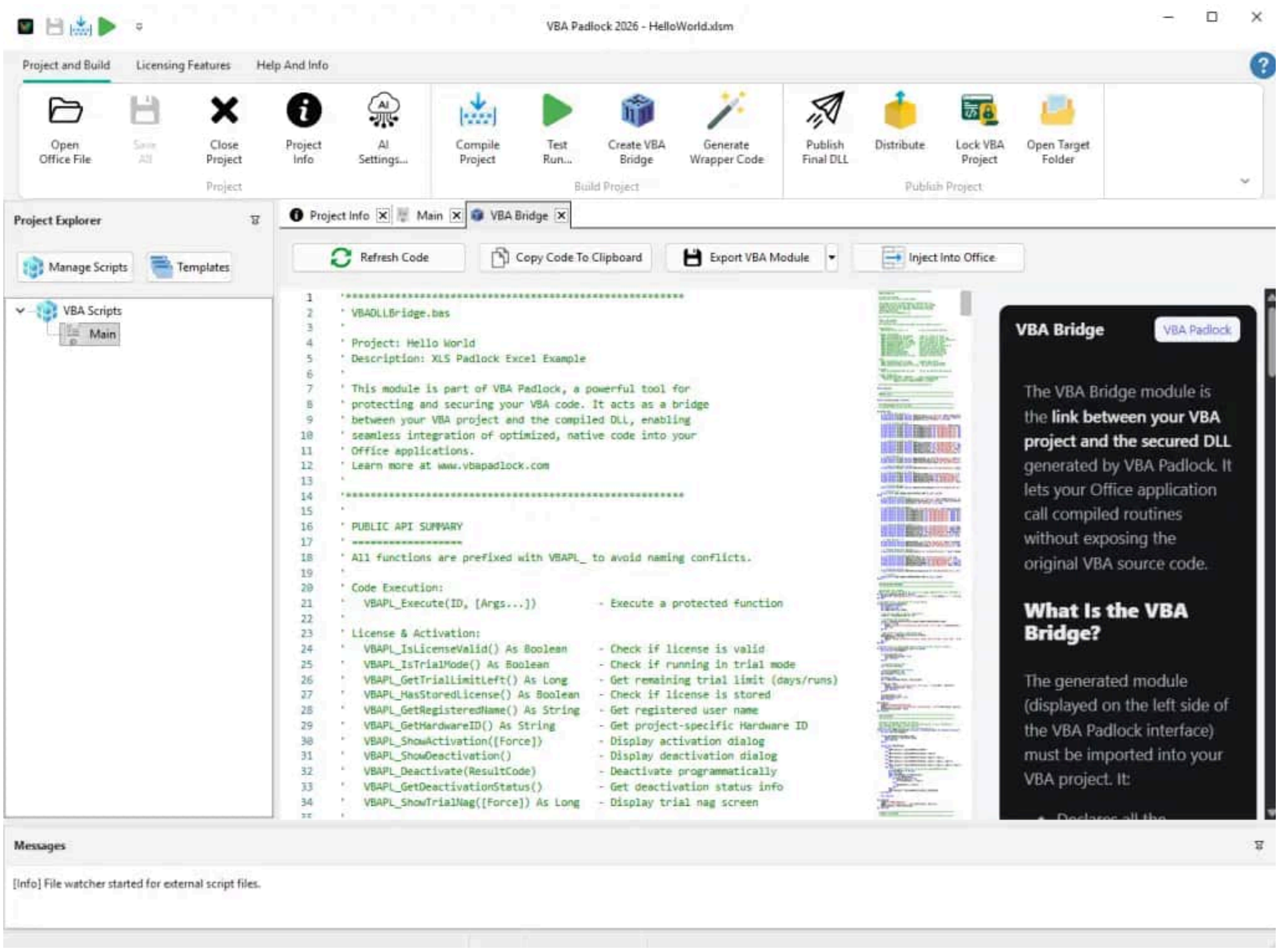
15.4 Step 4: Create the VBA Bridge

The VBA Bridge is a standard VBA module that VBA Padlock generates and injects into your Excel workbook. It contains all the `Declare` statements and wrapper functions (`VBAPL_Execute`, `VBAPL_IsLicenseValid`, etc.) that let your Excel VBA code communicate with the compiled DLL.

You do **not** write the bridge code manually — VBA Padlock creates it for you.

1. **Open your `MyExcelApp.xlsm` file in Excel** (enable macros when prompted)
2. **Go back to VBA Padlock** and click **"Create VBA Bridge"** in the ribbon
3. Click **"Inject Into Office"**

4 VBA Padlock inserts a new module named `VBADLLBridge` (or similar) into your workbook's VBA project. This module contains all the `VBAPL_*` functions.



```
'**********************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'**********************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean        - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is stored
'   VBAPL_GetRegisteredName() As String   - Get registered user name
'   VBAPL_GetHardwareID() As String       - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])         - Display activation dialog
'   VBAPL_ShowDeactivation()              - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)          - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()         - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long   - Display trial nag screen
'
```

VBA Bridge VBA Padlock

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

```
[Info] File watcher started for external script files.
```

Open the VBA Editor (`Alt+F11`) in Excel and you'll see the new bridge module. It includes functions such as:

- `VBAPL_Execute` — call any Sub or Function compiled in the DLL
- `VBAPL_IsLicenseValid` — check whether the application is activated
- `VBAPL_ShowActivation` — display the activation dialog
- `VBAPL_GetHardwareID` — retrieve the user's hardware identifier
- `VBAPL_IsTrialMode` — check if running in trial mode
- And more (see the VBA Bridge API Reference for the complete list)

⚠ **Caution**

Do not modify the generated bridge code manually. If you need to regenerate it later (for example, after changing project settings), VBA Padlock will replace it automatically.

15.5 Step 5: Write Caller Code in Excel

Now that the VBA Bridge is in place, you can call your compiled functions from regular Excel VBA code. The key function is `VBAPL_Execute` — you pass it the name of a Sub or Function (as a string), followed by any arguments.

In the VBA Editor, insert a **new standard module** (for example, `Module1`) and add your caller macros:

```vba
Sub Demo_HelloWorld()
    Call VBAPL_Execute("HelloWorld")
End Sub

Sub Demo_Greet()
    Dim UserName As String, result As Variant
    UserName = InputBox("Enter your name:", "Greet Demo")
    If Len(UserName) > 0 Then
        result = VBAPL_Execute("Greet", UserName)
        MsgBox result, vbInformation, "Greeting"
    End If
End Sub

Sub Demo_WorkbookInfo()
    Call VBAPL_Execute("ShowWorkbookInfo")
End Sub

Sub Demo_WriteToCell()
    Call VBAPL_Execute("WriteToCell", "A1", "Hello from DLL!")
    MsgBox "Value written to A1", vbInformation
End Sub

Sub Demo_ReadFromCell()
    Dim Value As Variant
    Value = VBAPL_Execute("ReadFromCell", "A1")
    MsgBox "Value in A1: " & CStr(Value), vbInformation
End Sub

Sub Demo_CalculateDiscount()
    Dim Price As Double, DiscountRate As Double, FinalPrice As Variant
    Price = 100: DiscountRate = 0.15
    FinalPrice = VBAPL_Execute("CalculateDiscount", Price, DiscountRate)
    MsgBox "Original: $" & Format(Price, "0.00") & vbCrLf & _
           "Discount: " & Format(DiscountRate * 100, "0") & "%" & vbCrLf & _
           "Final: $" & Format(FinalPrice, "0.00"), vbInformation, "Discount
Calculator"
End Sub
```

Notice the pattern:

- **For Subs** (no return value): `Call VBAPL_Execute("SubName", arg1, arg2, ...)`
- **For Functions** (returns a value): `result = VBAPL_Execute("FunctionName", arg1, arg2, ...)`

The first argument is always the name of the compiled Sub or Function as a string. Any additional arguments are passed through to that Sub or Function.

Tip

You can also add a `Workbook_Open` event in `ThisWorkbook` to run initialization code when the workbook opens. For example, to check licensing on startup:

```
Private Sub Workbook_Open()
    ' The VBA Bridge auto-initializes on first VBAPL_Execute call.
    ' Optionally check licensing here:
    If Not VBAPL_IsLicenseValid() Then
        VBAPL_ShowActivation
    End If
End Sub
```

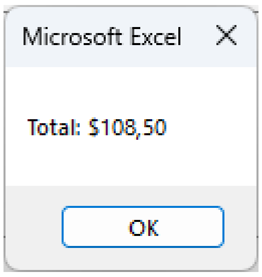
This is entirely optional — if you don’t configure licensing (Step 7), you can skip this.

Save the workbook.

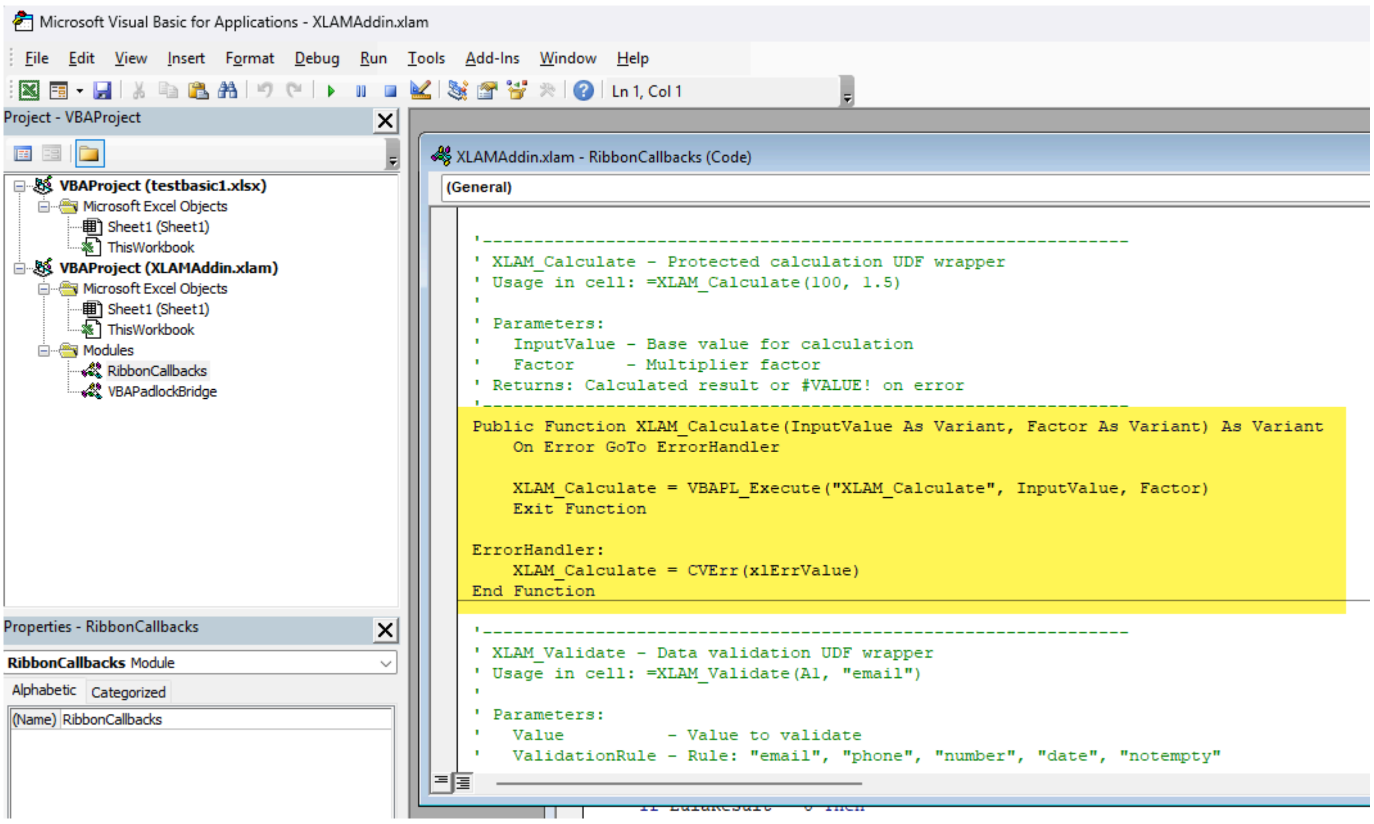
15.6 Step 6: Test Your Protected Application

Let’s verify everything works end to end.

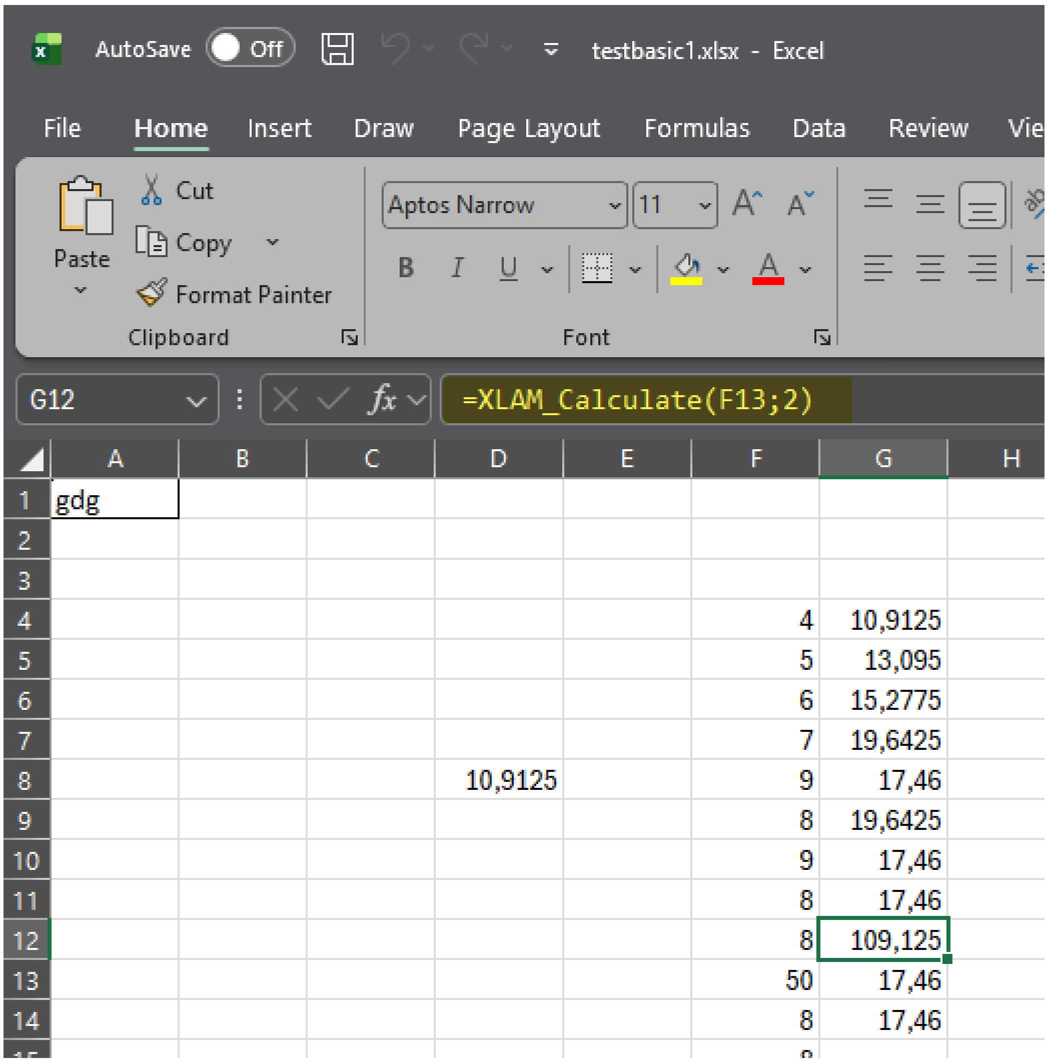
1. **Close and reopen `MyExcelApp.xlsm`** in Excel
2. **Enable macros** when prompted (click “Enable Content”)
3. **Press `Alt+F8`** to open the Macro dialog
4. **Select `Demo_CalculateDiscount`** and click **Run**
5. You should see a message box displaying the calculation result:



Total: $108,50



```
'-------------------------------------------------------------------
' XLAM_Calculate - Protected calculation UDF wrapper
' Usage in cell: =XLAM_Calculate(100, 1.5)
'
' Parameters:
'   InputValue - Base value for calculation
'   Factor     - Multiplier factor
' Returns: Calculated result or #VALUE! on error
'-------------------------------------------------------------------
Public Function XLAM_Calculate(InputValue As Variant, Factor As Variant) As Variant
    On Error GoTo ErrorHandler

    XLAM_Calculate = VBAPL_Execute("XLAM_Calculate", InputValue, Factor)
    Exit Function

ErrorHandler:
    XLAM_Calculate = CVErr(xlErrValue)
End Function
'-------------------------------------------------------------------
' XLAM_Validate - Data validation UDF wrapper
' Usage in cell: =XLAM_Validate(A1, "email")
'
' Parameters:
'   Value          - Value to validate
'   ValidationRule - Rule: "email", "phone", "number", "date", "notempty"
```



G12 =XLAM_Calculate(F13;2)

	A	B	C	D	E	F	G	H
1	gdg							
2								
3								
4						4	10,9125	
5						5	13,095	
6						6	15,2775	
7						7	19,6425	
8				10,9125		9	17,46	
9						8	19,6425	
10						9	17,46	
11						8	17,46	
12						8	109,125	
13						50	17,46	
14						8	17,46	

If all the demos work, your code is successfully compiled into the DLL and running through the VBA Bridge.

Caution

If you get "Run-time error '48': File not found", make sure the `bin\` folder exists in the same directory as `MyExcelApp.xlsm` and contains all three DLL files. The VBA Bridge uses `AddDllDirectory` to register the `bin\` path — if the folder or files are missing, the DLL cannot be loaded.

15.7 Step 7: Configure Licensing (Optional)

If you want to require an activation key before your application runs, follow this step. If you're just protecting your code without licensing, you can skip ahead to Step 8.

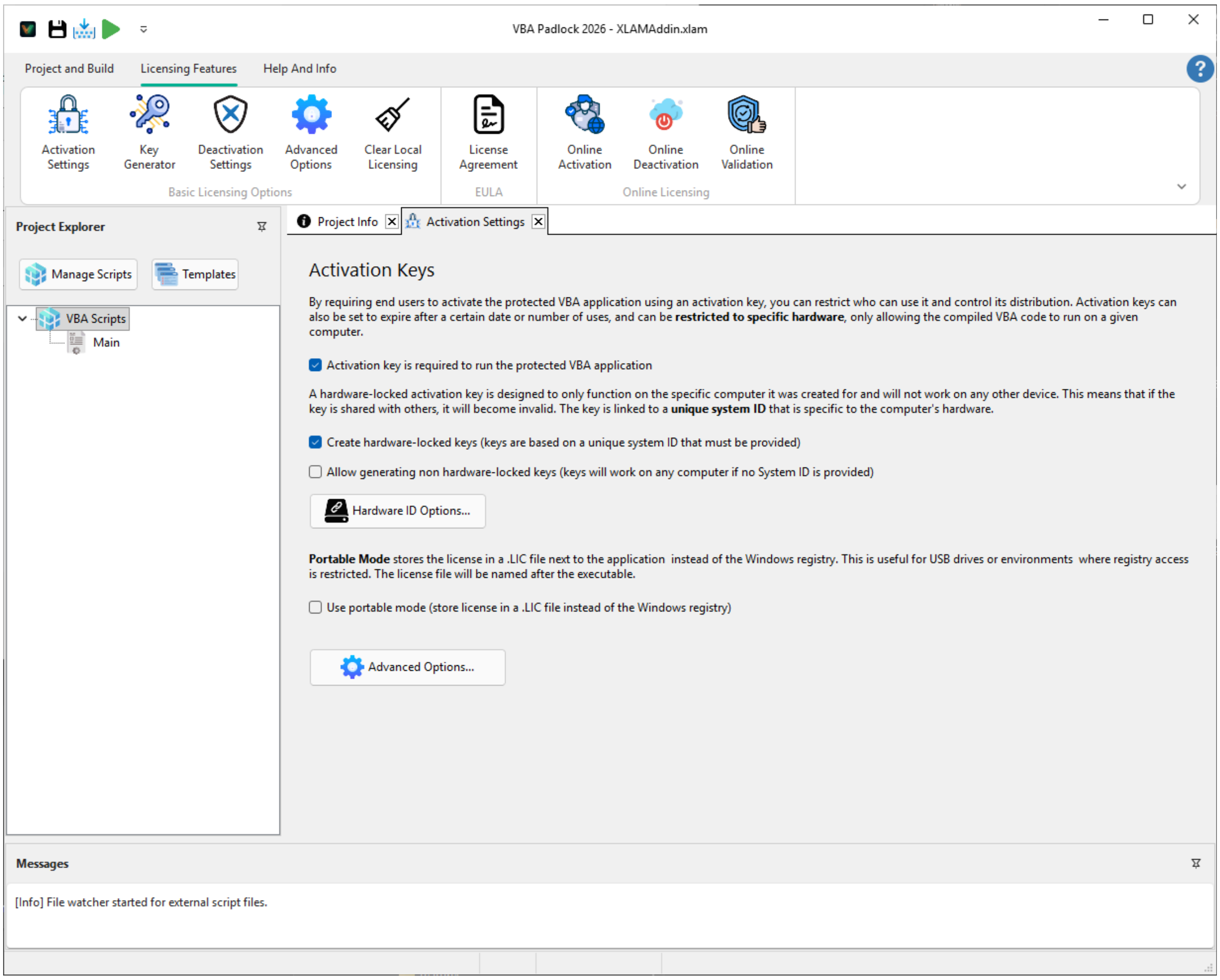
15.7.2 Enable Activation Keys

1. **Switch to the "Licensing Features" tab** in the VBA Padlock ribbon

2 **Click "Activation Settings"**

3 **Check "Activation key is required to run the protected VBA application"**

4 Click **OK**



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

☑ Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

☑ Create hardware-locked keys (keys are based on a unique system ID that must be provided)

☐ Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

☐ Use portable mode (store license in a .LIC file instead of the Windows registry)

```
[Info] File watcher started for external script files.
```

15.7.3 Enable Hardware Locking

Hardware locking ties each license key to a specific computer, preventing key sharing.

1 In the Activation Settings dialog, **check "Create hardware-locked keys"**

2. **Click "Hardware ID Options..."** to choose which hardware components to include:
   - Computer Name (recommended)
   - Windows Volume Serial (recommended)
   - BIOS Serial Number (recommended)
   - CPU Serial (optional — some CPUs don't report it)
   - MAC Address (not recommended — changes with network adapters)
3. Click **OK**

Tip

Why hardware locking? Without it, users could share the same activation key with anyone. With hardware locking, each key is cryptographically bound to a specific computer's hardware profile.

15.7.4 Rebuild After Changing Licensing Settings

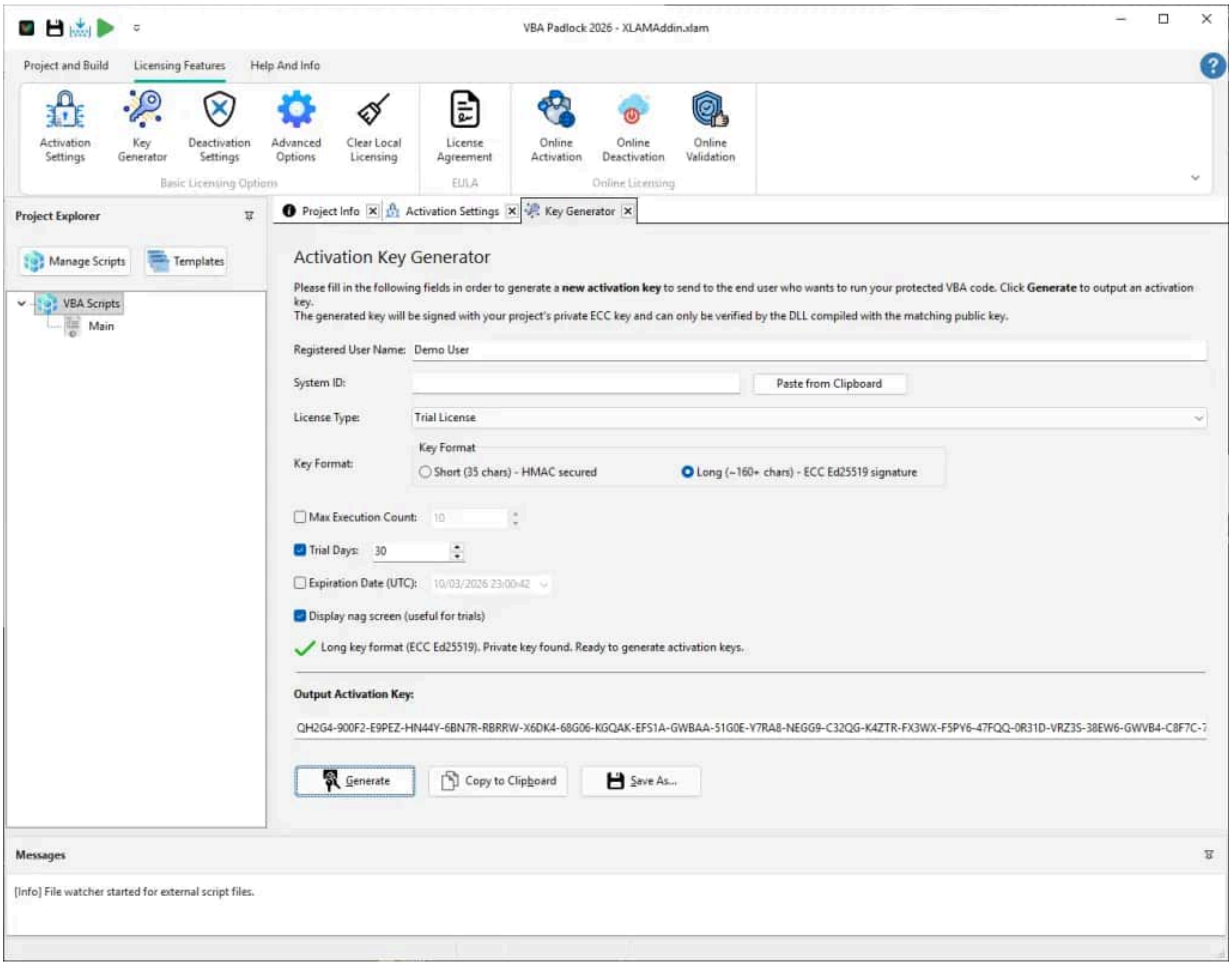
After changing licensing settings, you need to rebuild the DLL:

1. Go back to the **"Project and Build"** tab
2. Click **"Publish Final DLL"** then **"Build Final DLL Files"**
3. The new DLL now enforces activation

15.7.5 Generate a Test License Key

To test the activation flow on your own computer:

1. **Click "Key Generator"** in the Licensing Features ribbon
2. The **Hardware ID** field shows your computer's ID
3. Click **"Generate Key"**
4. Copy the generated key — you'll enter it when testing



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format: Key Format ( ) Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

[Info] File watcher started for external script files.

15.7.6 Add Licensing Check to Your Workbook

Update the `Workbook_Open` event in `ThisWorkbook`:

```
Private Sub Workbook_Open()
    ' Check if the application is activated
    If Not VBAPL_IsLicenseValid() Then
        VBAPL_ShowActivation
    End If

    ' After activation, the user can use the application normally
End Sub
```

You can also let users retrieve their Hardware ID so they can send it to you:

```
Sub ShowMyHardwareID()
    Dim hwid As String
    hwid = VBAPL_GetHardwareID()
    MsgBox "Your Hardware ID is:" & vbCrLf & hwid, vbInformation, "Hardware
ID"
End Sub
```

Rebuild the DLL one more time after making these changes, then test by opening the workbook — you should see the activation dialog.

15.8 Step 8: Distribute Your Application

You're ready to ship. VBA Padlock includes a **Distribution** dialog to package your files with the correct folder structure.

1. **Click "Distribute"** in the ribbon (Project and Build tab)
2. Choose one of two options:
    - **Create ZIP archive** — creates a ready-to-ship ZIP file
    - **Copy to folder** — copies all files to a specified directory
3. For this tutorial, select **"Create ZIP archive"**
4. Choose a destination and filename (e.g., `MyExcelApp_v1.0.zip`)
5. VBA Padlock creates a ZIP containing:

```
MyExcelApp_v1.0.zip
├── MyExcelApp.xlsm
└── bin\
    ├── MyExcelApprun32.dll
    ├── MyExcelApprun64.dll
    └── MyExcelApp.dll
```

Caution

The `bin\` folder structure is mandatory. The VBA Bridge expects to find the DLL files in a `bin\` subfolder relative to the Excel file. If you distribute the files without this structure, the application will fail to load.

15.8.7 Installation Instructions for Users

Include this in your distribution (e.g., in a `README.txt`):

```
INSTALLATION INSTRUCTIONS
=========================

1. Extract all files to a folder on your computer (e.g., C:\MyExcelApp\)

2. IMPORTANT: Keep the folder structure intact. The 'bin' folder must remain
   in the same directory as MyExcelApp.xlsm

3. Open MyExcelApp.xlsm in Excel

4. If you see a security warning, click "Enable Content"

5. If prompted, enter your activation key and click "Activate"

6. You're ready to use the application!

IMPORTANT:
- Keep MyExcelApp.xlsm and the 'bin' folder together at all times
- Do not move or rename files in the 'bin' folder
- The 'bin' folder contains 3 DLL files that are required for operation
```

Note

Excel Macro Security: Users must enable macros for your application to work. If their security settings block macros, they should add your application's folder to Excel's Trusted Locations: **File > Options > Trust Center > Trust Center Settings > Trusted Locations**.

15.9 What's Next?

Congratulations! You've successfully compiled your Excel VBA code into a protected DLL and wired it up through the VBA Bridge. Your source code is no longer visible in the workbook.

Online Activation

Automate activation with a PHP server so users can activate over the internet.

Read the guide

Trial Mode

Let users try your application before requiring a license key.

Learn more

License Deactivation

Allow users to transfer their license to a new computer.

Learn more

Advanced Features

Explore EULA display, localization, and code obfuscation.

Browse features

15.10 Troubleshooting

15.10.8 "Run-time error '48': File not found"

Cause: Excel cannot find the DLL files in the `bin\` folder.

Solution:

1. Make sure the `bin\` folder exists in the same directory as `MyExcelApp.xlsm`
2. Verify that the `bin\` folder contains all three DLL files:
   - `MyExcelApprun32.dll`
   - `MyExcelApprun64.dll`
   - `MyExcelApp.dll`
3. Do not rename or move the `bin\` folder or any files inside it

15.10.9 "Activation failed: Invalid key"

Cause: The activation key does not match the user's Hardware ID.

Solution:

1. Ask the user for their Hardware ID (they can run the `ShowMyHardwareID` macro or call `VBAPL_GetHardwareID()`)
2. Generate a new key specifically for that Hardware ID using the Key Generator
3. Do not reuse keys generated for a different computer

15.10.10 "The application is not activated"

Cause: The stored license was deleted (e.g., Windows reinstall, registry cleanup).

Solution: The user needs to reactivate with their existing key. If the hardware has not changed, the same key will work again.

15.10.11 Macros are disabled

Cause: Excel's security settings block macros.

Solution: Add the application folder to Excel's Trusted Locations:

1. Go to **File > Options > Trust Center > Trust Center Settings**
2. Click **Trusted Locations**
3. Click **Add new location** and browse to your application folder

15.10.12 "Type mismatch" or unexpected return value

Cause: The return value from `VBAPL_Execute` is a `Variant`. If you assign it to a strongly typed variable without conversion, you may get a type mismatch.

Solution: Use `CStr()`, `CDbl()`, `CLng()`, or similar conversion functions when assigning the result to a typed variable, or declare your receiving variable as `Variant`.

15.11 See Also

- VBA Bridge API Reference — Complete list of all `VBAPL_*` functions
- Protect a Word Document — Similar tutorial for Word
- Protect an Access Database — Similar tutorial for Access
- Activation Settings — Detailed activation settings reference
- Key Generator — Advanced key generation options
- Publish Final DLL — Publish dialog reference
- Create VBA Bridge — VBA Bridge dialog reference

16. Protect a Word Document

You have built a Word document with VBA macros — maybe it generates contracts from templates, fills in client details automatically, or produces formatted reports — and you need to distribute it without exposing your source code.

In this tutorial, you'll walk through the complete workflow: compiling your VBA logic into a protected DLL, wiring it up to Word through the auto-generated VBA Bridge, and optionally adding activation keys with hardware locking.

Time needed: 15-20 minutes

> **Tip**
>
> **Before you start**, make sure you have:
>
> - VBA Padlock installed (installation guide)
> - A macro-enabled Word document or template (`.docm` or `.dotm` file)
> - Basic familiarity with the Word VBA Editor (`Alt+F11`)

16.1 The Scenario: Document Builder

Imagine you sell a **"Document Builder"** tool: users open a Word template containing placeholders like `{CUSTOMER_NAME}`, click a button, and the macro fills everything in, adds a formatted table, and exports the result to PDF.

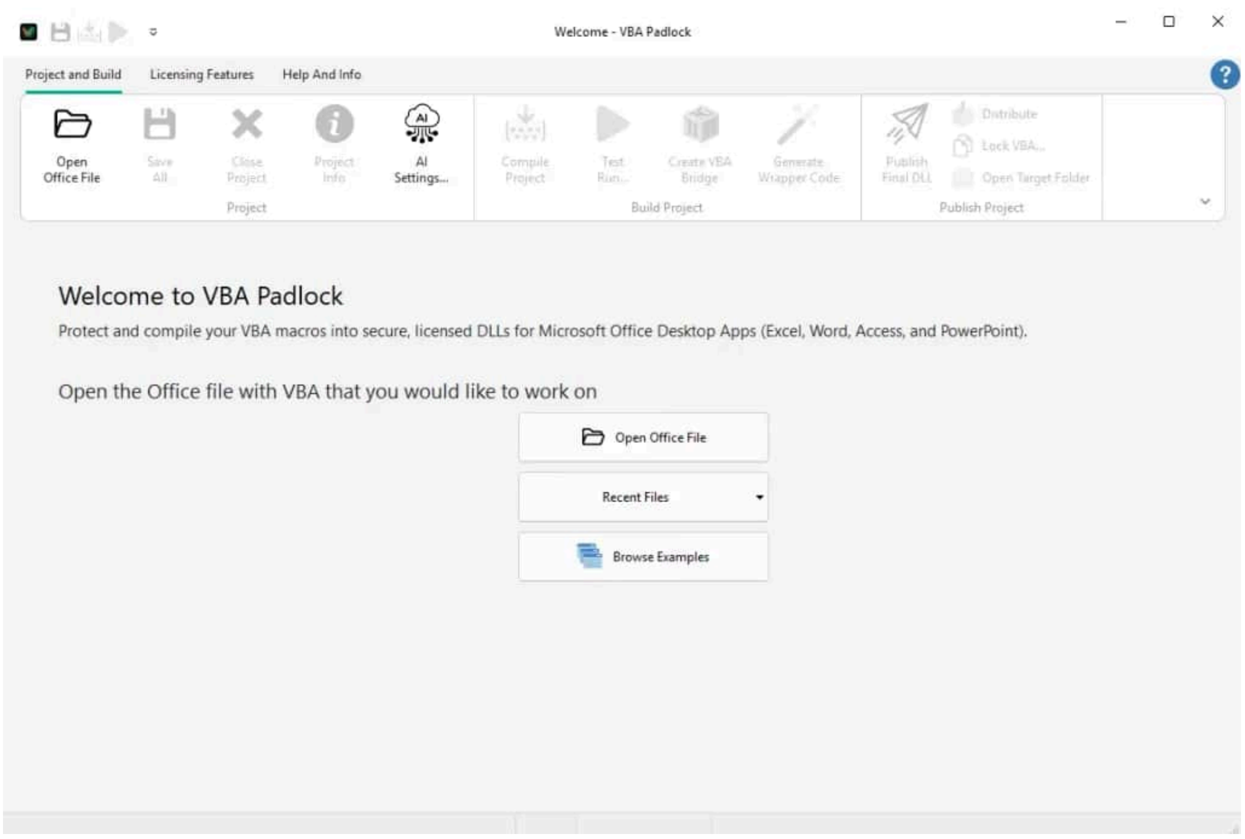
VBA Padlock allows you to commercialize this tool while keeping your proprietary logic completely hidden inside a native DLL.

16.2 Step 1: Create a VBA Padlock Project

Start by opening your Word file in VBA Padlock to create a project that will hold your compiled code and licensing settings.

1. **Launch VBA Padlock** from the Start menu.

2 **Click "Open Office File"** in the ribbon and select your `.docm` or `.dotm` file (for example, `DocumentBuilder.docm`).



Welcome to VBA Padlock

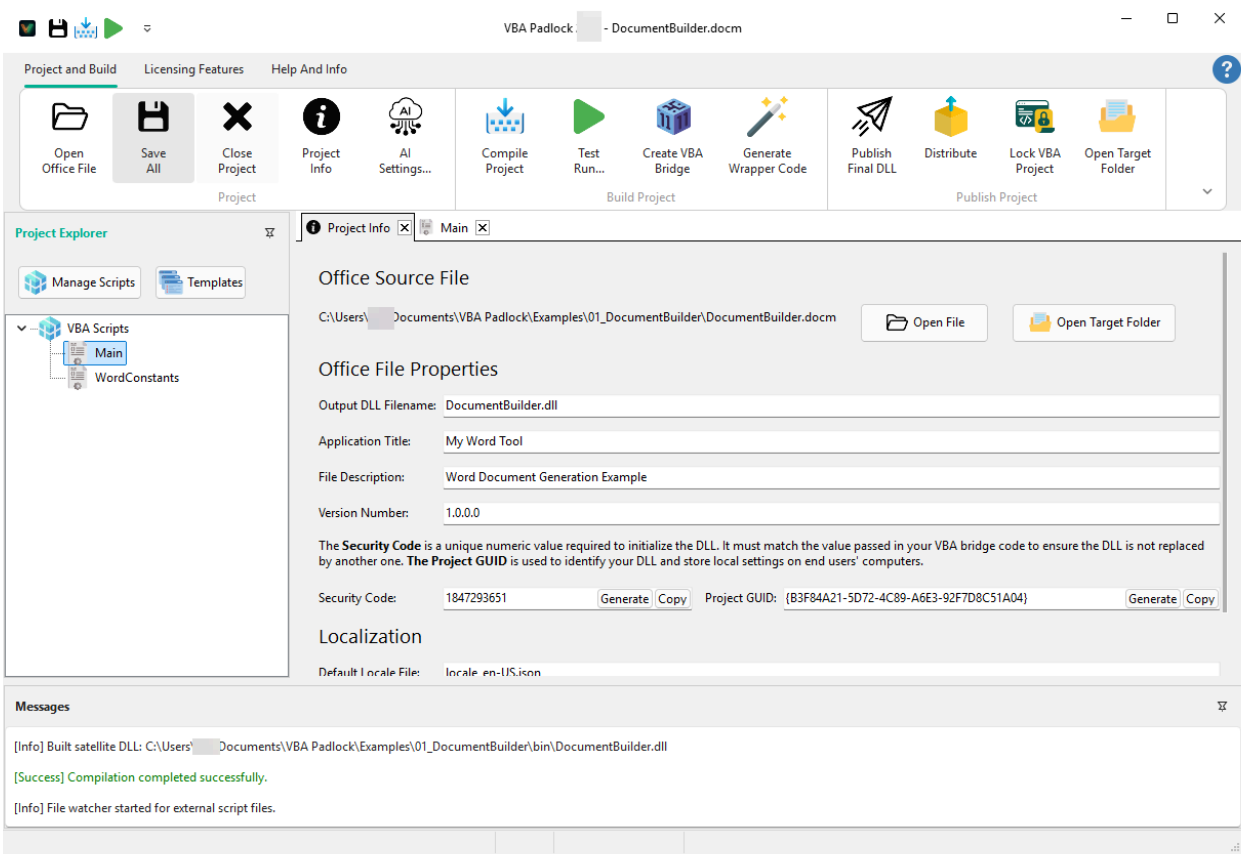
Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

3 **Set Project Information**

In the **Project Info** tab:

- **Application Title:** Give your tool a name (e.g., `Document Builder`).
- **Security Code:** Click **Generate** to create the unique cryptographic link between your document and the DLL.



Office Source File

C:\Users\[redacted]Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

Office File Properties

Output DLL Filename: DocumentBuilder.dll

Application Title: My Word Tool

File Description: Word Document Generation Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 1847293651 Project GUID: {B3F84A21-5D72-4C89-A6E3-92F7D8C51A04}

Localization

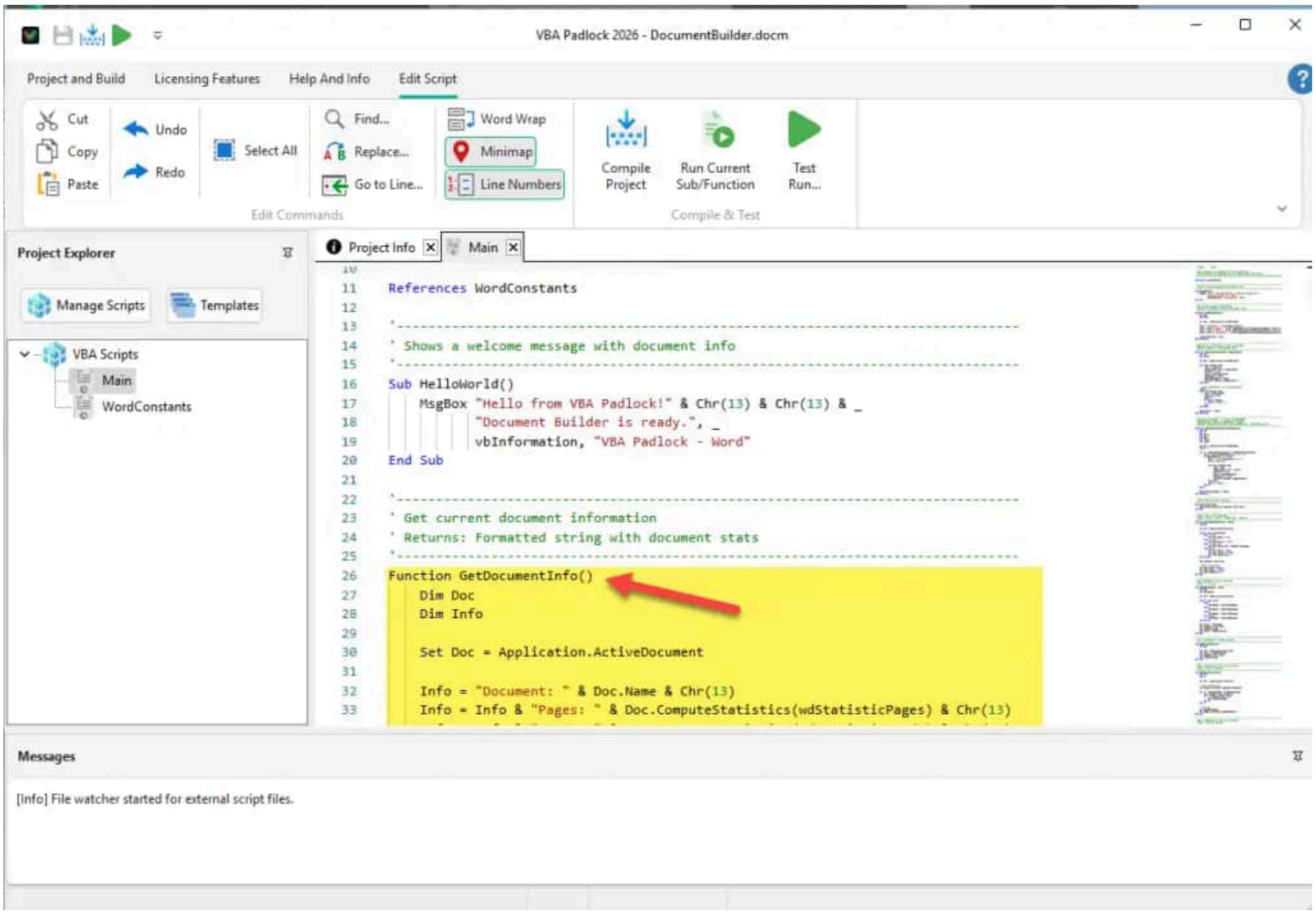
Default Locale File: locale_en-US.json

Messages

```
[Info] Built satellite DLL: C:\Users\[redacted]Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\DocumentBuilder.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

16.3 Step 2: Write Your Protected Scripts

Switch to the **Code Editor**. This is where you write the VBA logic you want to hide. Below is an excerpt based on the real *DocumentBuilder* example included with VBA Padlock.



```
References WordConstants

'-------------------------------------------------------------------------------
' Shows a welcome message with document info
'-------------------------------------------------------------------------------
Sub HelloWorld()
    MsgBox "Hello from VBA Padlock!" & Chr(13) & Chr(13) & _
           "Document Builder is ready.", _
           vbInformation, "VBA Padlock - Word"
End Sub

'-------------------------------------------------------------------------------
' Get current document information
' Returns: Formatted string with document stats
'-------------------------------------------------------------------------------
Function GetDocumentInfo()
    Dim Doc
    Dim Info

    Set Doc = Application.ActiveDocument

    Info = "Document: " & Doc.Name & Chr(13)
    Info = Info & "Pages: " & Doc.ComputeStatistics(wdStatisticPages) & Chr(13)
```

[Info] File watcher started for external script files.

```
' Main.bas -- Compiled into the DLL
' These functions access Word's object model via the "Application" object.

Sub HelloWorld()
    MsgBox "Hello from VBA Padlock!" & Chr(13) & Chr(13) & _
           "Document Builder is ready.", _
           vbInformation, "VBA Padlock - Word"
End Sub

'-----------------------------------------------------------------------------
--
' Get current document information
' Returns: Formatted string with document stats
'-----------------------------------------------------------------------------
--
Function GetDocumentInfo()
    Dim Doc
    Dim Info

    Set Doc = Application.ActiveDocument

    Info = "Document: " & Doc.Name & Chr(13)
    Info = Info & "Pages: " & Doc.ComputeStatistics(wdStatisticPages) &
Chr(13)
    Info = Info & "Words: " & Doc.ComputeStatistics(wdStatisticWords) &
Chr(13)
    Info = Info & "Characters: " &
Doc.ComputeStatistics(wdStatisticCharacters)

    GetDocumentInfo = Info
End Function

' Replace placeholders like {CUSTOMER_NAME} with actual data
Function ReplacePlaceholders(Placeholders)
    Dim Doc, i, Parts, Key, Value, Count
    Set Doc = Application.ActiveDocument : Count = 0

    For i = LBound(Placeholders) To UBound(Placeholders)
        Parts = Split(Placeholders(i), "=", 2)
        If UBound(Parts) >= 1 Then
```

```
                Key = "{" & Trim(Parts(0)) & "}" : Value = Parts(1)
                With Doc.Content.Find
                    .Text = Key : .Replacement.Text = Value
                    .Forward = True : .Wrap = 1 : .Execute(Replace:=2)
                End With
                Count = Count + 1
            End If
        Next i
        ReplacePlaceholders = Count
    End Function

    ' Export the active document to PDF
    Function ExportToPDF(PDFPath)
        On Error Resume Next
        Application.ActiveDocument.ExportAsFixedFormat _
            OutputFileName:=PDFPath, ExportFormat:=17, OpenAfterExport:=False
        ExportToPDF = (Err.Number = 0)
    End Function

    ' Insert text with specific formatting
    Sub InsertFormattedText(Text, Style)
        With Application.Selection
            Select Case Style
                Case "Bold": .Font.Bold = True
                Case "Italic": .Font.Italic = True
                Case Else: .Font.Bold = False: .Font.Italic = False
            End Select
            .TypeText Text:=Text
        End With
    End Sub
```

Tip

Compiled scripts have full access to the host application through the `Application` object. See Office Object Model Access.

16.4 Step 3: Compile and Publish

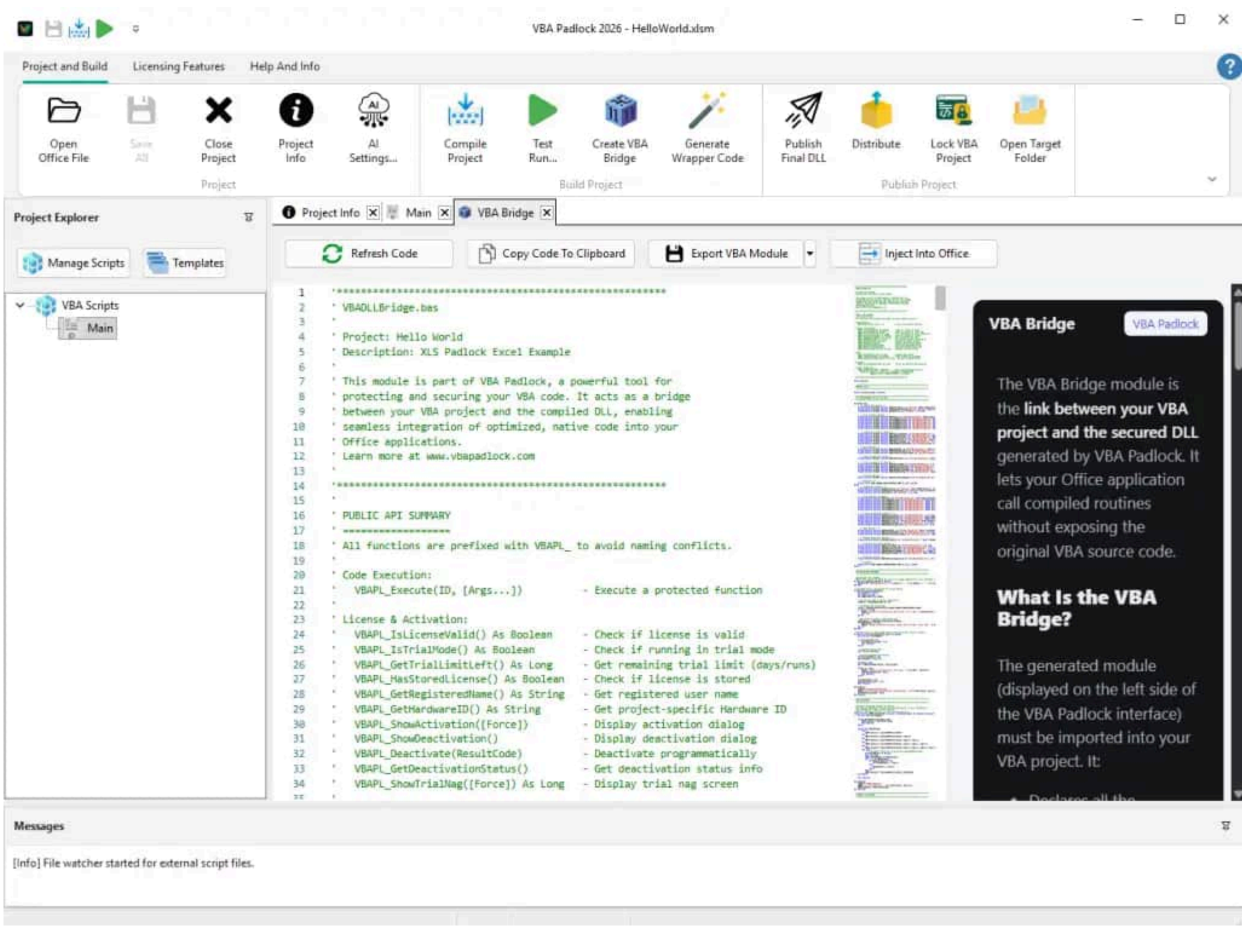
Build the protected DLL that will replace your source code.

1. **Click "Publish Final DLL"** in the ribbon.
2. **Click "Build Final DLL Files"**.
3. VBA Padlock produces three DLLs in a `bin` subfolder next to your document:
   - `DocumentBuilderrun32.dll` (32-bit runtime)
   - `DocumentBuilderrun64.dll` (64-bit runtime)
   - `DocumentBuilder.dll` (Your compiled code)

16.5 Step 4: Inject the VBA Bridge

The VBA Bridge is the connection between your Word document and the DLL.

1. Open your Word file in Microsoft Word.
2. In VBA Padlock, click **Create VBA Bridge → Inject Into Office**.
3. VBA Padlock adds the `VBADLLBridge` module to your Word project.



```
'*****************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*****************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])         - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean    - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean       - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long    - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean  - Check if license is stored
'   VBAPL_GetRegisteredName() As String  - Get registered user name
'   VBAPL_GetHardwareID() As String      - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])        - Display activation dialog
'   VBAPL_ShowDeactivation()             - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)         - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()        - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long  - Display trial nag screen
'
```

VBA Bridge

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

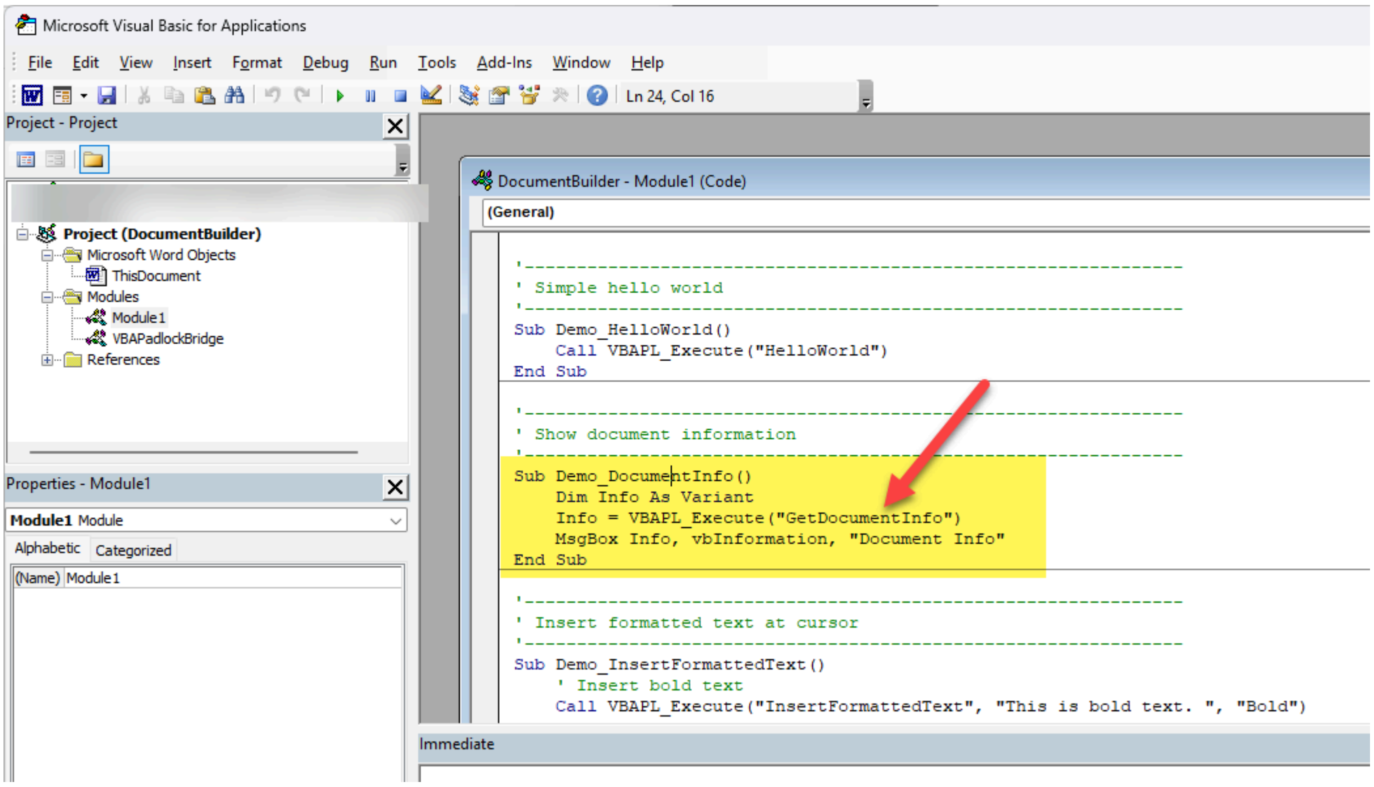
What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

[Info] File watcher started for external script files.

16.6 Step 5: Write Caller Code in Word

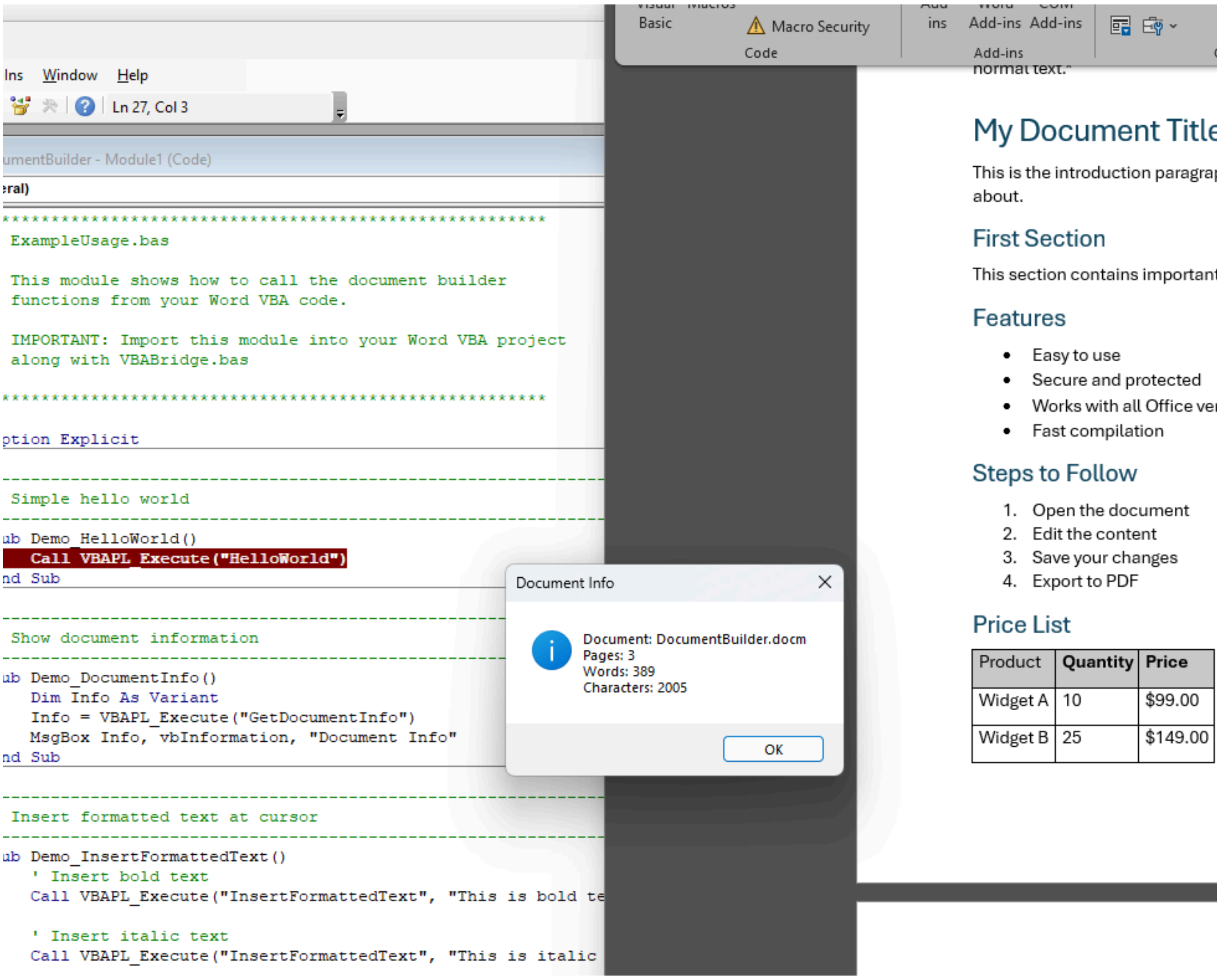
Now you can call your compiled functions from regular Word VBA code using `VBAPL_Execute`.



```
'--------------------------------------------------------------
' Simple hello world
'--------------------------------------------------------------
Sub Demo_HelloWorld()
    Call VBAPL_Execute("HelloWorld")
End Sub

'--------------------------------------------------------------
' Show document information
'--------------------------------------------------------------
Sub Demo_DocumentInfo()
    Dim Info As Variant
    Info = VBAPL_Execute("GetDocumentInfo")
    MsgBox Info, vbInformation, "Document Info"
End Sub

'--------------------------------------------------------------
' Insert formatted text at cursor
'--------------------------------------------------------------
Sub Demo_InsertFormattedText()
    ' Insert bold text
    Call VBAPL_Execute("InsertFormattedText", "This is bold text. ", "Bold")
```

In the Word VBA Editor (`Alt+F11`), create a new module and add this:

```vba
'----------------------------------------------------------------
' Simple hello world
'----------------------------------------------------------------
Sub Demo_HelloWorld()
    Call VBAPL_Execute("HelloWorld")
End Sub

'----------------------------------------------------------------
' Show document information
'----------------------------------------------------------------
Sub Demo_DocumentInfo()
    Dim Info As Variant
    ' Call the compiled DLL function
    Info = VBAPL_Execute("GetDocumentInfo")
    MsgBox Info, vbInformation, "Document Info"
End Sub

'----------------------------------------------------------------
' Insert formatted text at cursor
'----------------------------------------------------------------
Sub Demo_InsertFormattedText()
    ' Insert bold text
    Call VBAPL_Execute("InsertFormattedText", "This is bold text. ", "Bold")

    ' Insert italic text
    Call VBAPL_Execute("InsertFormattedText", "This is italic text. ",
"Italic")

    ' Insert normal text
    Call VBAPL_Execute("InsertFormattedText", "This is normal text.",
"Normal")
End Sub
```



```
**********************************************************
 ExampleUsage.bas

 This module shows how to call the document builder
 functions from your Word VBA code.

 IMPORTANT: Import this module into your Word VBA project
 along with VBABridge.bas

**********************************************************

ption Explicit

------------------------------------------------------------------
 Simple hello world
------------------------------------------------------------------
ub Demo_HelloWorld()
   Call VBAPL_Execute("HelloWorld")
nd Sub

------------------------------------------------------------------
 Show document information
------------------------------------------------------------------
ub Demo_DocumentInfo()
   Dim Info As Variant
   Info = VBAPL_Execute("GetDocumentInfo")
   MsgBox Info, vbInformation, "Document Info"
nd Sub

------------------------------------------------------------------
 Insert formatted text at cursor
------------------------------------------------------------------
ub Demo_InsertFormattedText()
   ' Insert bold text
   Call VBAPL_Execute("InsertFormattedText", "This is bold te

   ' Insert italic text
   Call VBAPL_Execute("InsertFormattedText", "This is italic
```

Document Info

Document: DocumentBuilder.docm
Pages: 3
Words: 389
Characters: 2005

OK

normal text."

My Document Title

This is the introduction paragra
about.

First Section

This section contains important

Features

- Easy to use
- Secure and protected
- Works with all Office ve
- Fast compilation

Steps to Follow

1. Open the document
2. Edit the content
3. Save your changes
4. Export to PDF

Price List

Product	Quantity	Price
Widget A	10	$99.00
Widget B	25	$149.00

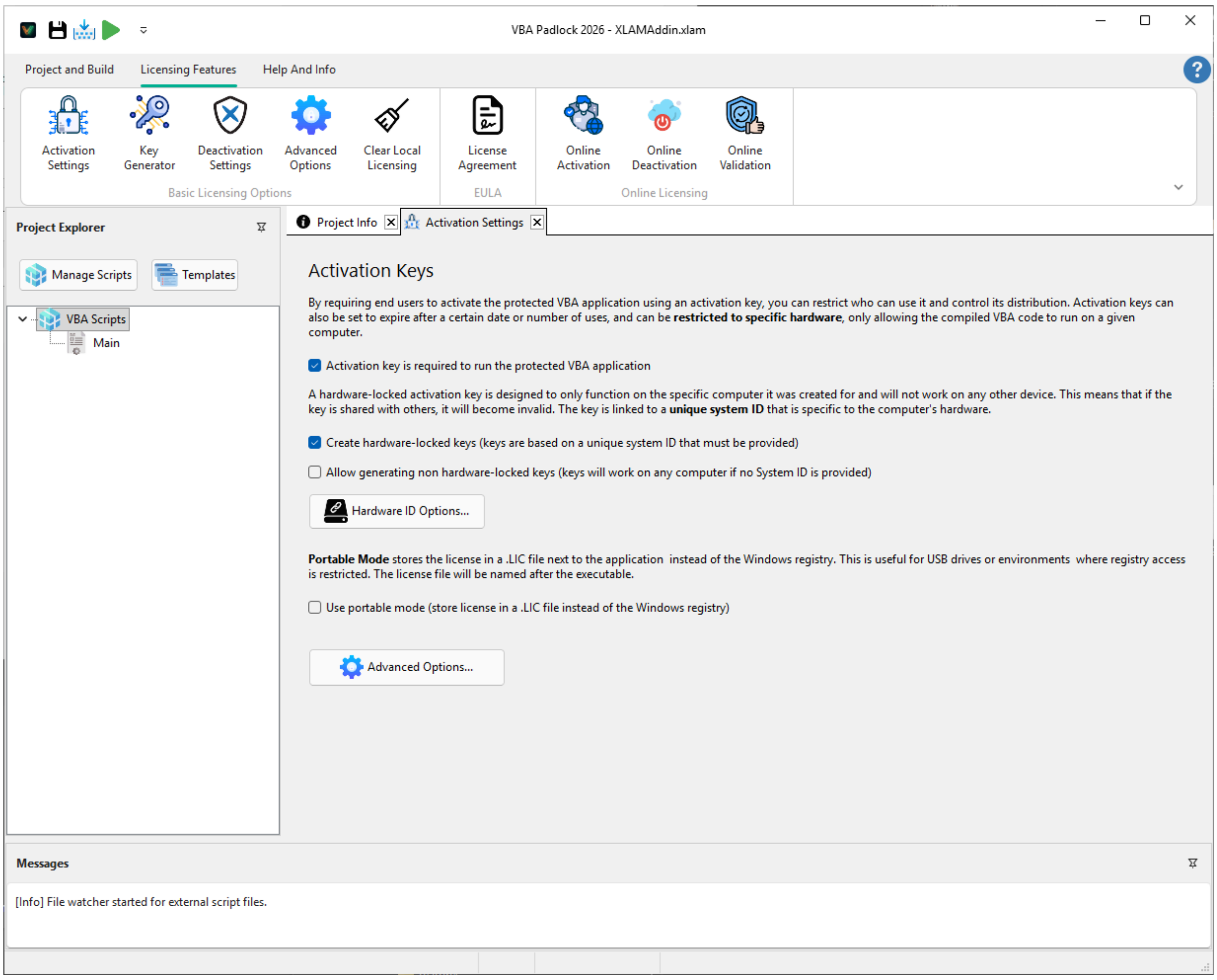
16.7 Step 6: Licensing (Optional)

To require an activation key before your document can be used:

1. Go to **Licensing Features** → Activation Settings.

2. Check **Activation key is required**.

3 Optional: Enable **Hardware Locking** to tie licenses to specific PCs.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

☑ Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

☑ Create hardware-locked keys (keys are based on a unique system ID that must be provided)

☐ Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

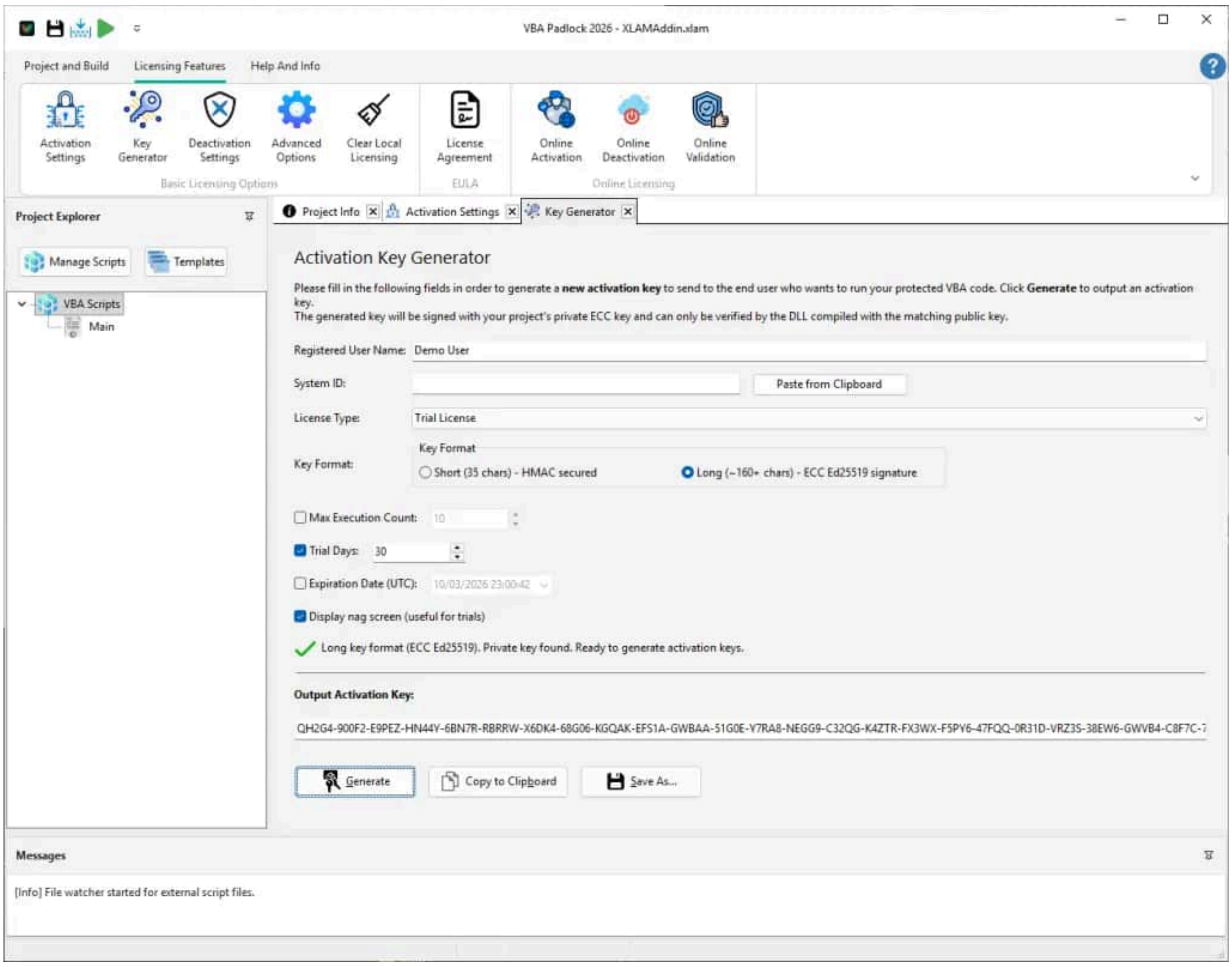
Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

☐ Use portable mode (store license in a .LIC file instead of the Windows registry)

[Info] File watcher started for external script files.

4 **Rebuild** the DLL to apply the security settings.

5 Generate a test key in the **Key Generator**.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format:
Key Format
( ) Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

```
[Info] File watcher started for external script files.
```

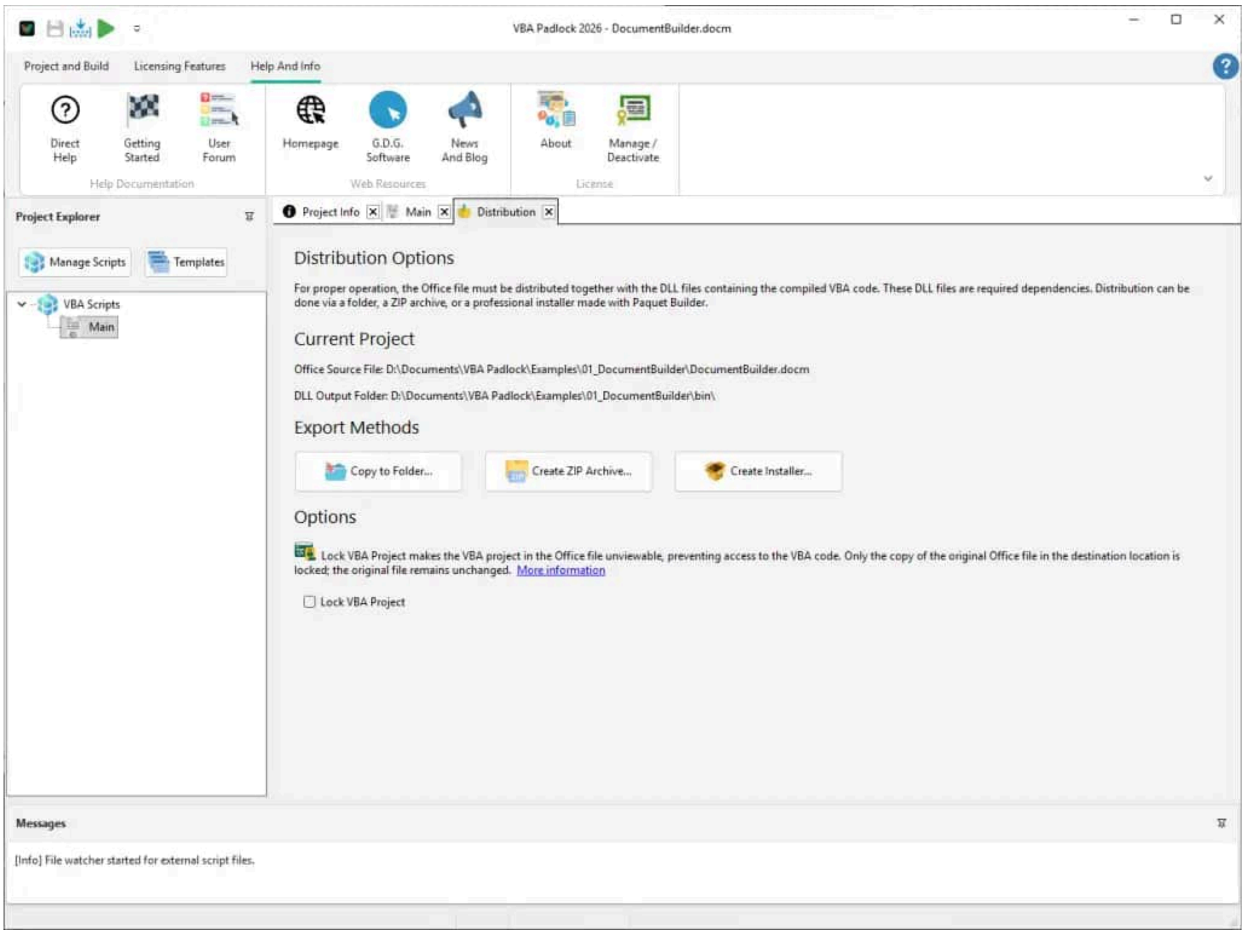
16.8 Step 7: Distribution

Package your application for release.

1 Click **Distribute** in the ribbon.

2 Select **Create ZIP Archive**.

3 The archive will contain your `.docm` file and the `bin` folder (mandatory).



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

[Info] File watcher started for external script files.

16.9 Next Steps

Protect Excel

Similar workflow tailored for Excel workbooks. Read the guide →

Online Activation

Automate license activation over the internet. Read the guide →

VBA Bridge API

Full reference for all `VBAPL_*` functions. View API reference →

VBA Compatibility

Check supported VBA features and statements. View reference →

17. Protect a PowerPoint Presentation

You have built a PowerPoint presentation that automates slide generation, applies complex layouts, or handles sensitive data exports — and you need to distribute it without exposing your source code.

In this tutorial, you'll walk through the complete workflow: compiling your PowerPoint logic into a protected DLL, wiring it up to PowerPoint through the auto-generated VBA Bridge, and optionally adding activation keys with hardware locking.

Time needed: 15-20 minutes

Tip

Before you start, make sure you have:

- VBA Padlock installed (installation guide)
- A macro-enabled PowerPoint file or add-in (`.pptm` or `.ppam` file)
- Basic familiarity with the PowerPoint VBA Editor (`Alt+F11`)

17.1 The Scenario: Slide Builder

Imagine you sell a **"Slide Builder"** tool: users open a template, click a button, and the macro automatically generates a complete slide deck based on external data, adds custom branding, and exports the result to PDF.

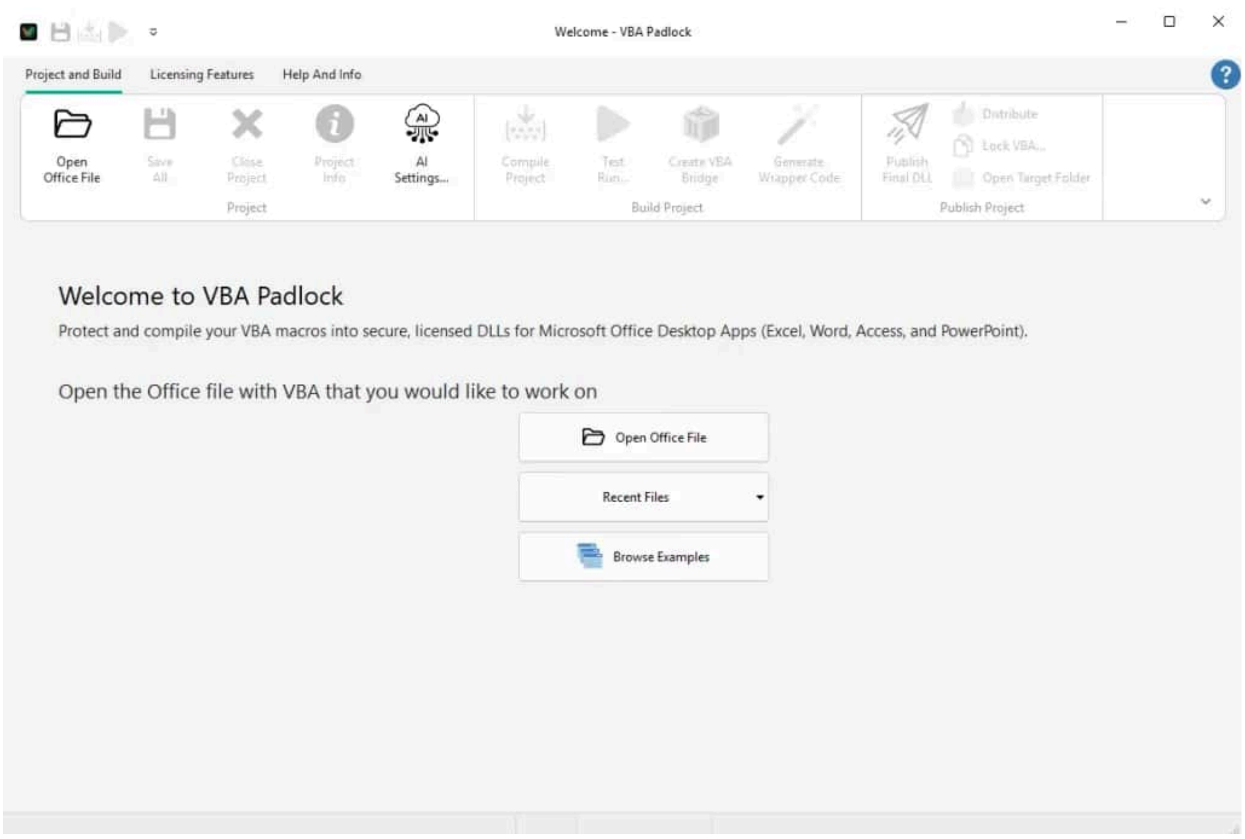
VBA Padlock allows you to commercialize this tool while keeping your proprietary layout algorithms and branding logic completely hidden inside a native DLL.

17.2 Step 1: Create a VBA Padlock Project

Start by opening your PowerPoint file in VBA Padlock to create a project that will hold your compiled code and licensing settings.

1. **Launch VBA Padlock** from the Start menu.

2 **Click “Open Office File”** in the ribbon and select your `.pptm` or `.ppam` file (for example, `SlideBuilder.pptm`).



Welcome to VBA Padlock

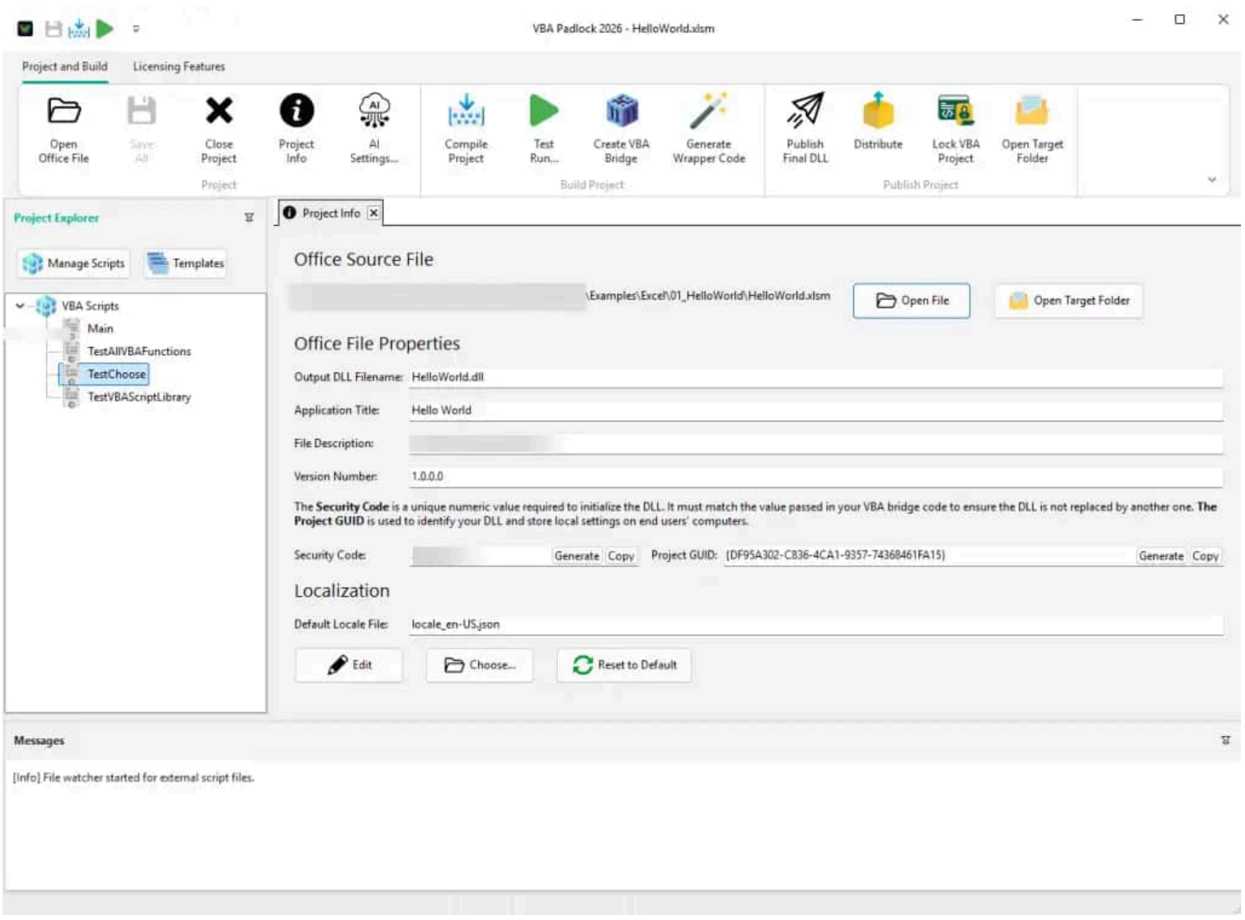
Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

3 **Set Project Information**

In the **Project Info** tab:

- **Application Title:** Give your tool a name (e.g., `Slide Builder`).
- **Security Code:** Click **Generate** to create the unique cryptographic link between your presentation and the DLL.



Office Source File

\Examples\Excel\01_HelloWorld\HelloWorld.xlsm [Open File] [Open Target Folder]

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description:

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: [Generate] [Copy] Project GUID: {DF95A302-C836-4CA1-9357-74368461FA15} [Generate] [Copy]

Localization

Default Locale File: locale_en-US.json

[Edit] [Choose...] [Reset to Default]

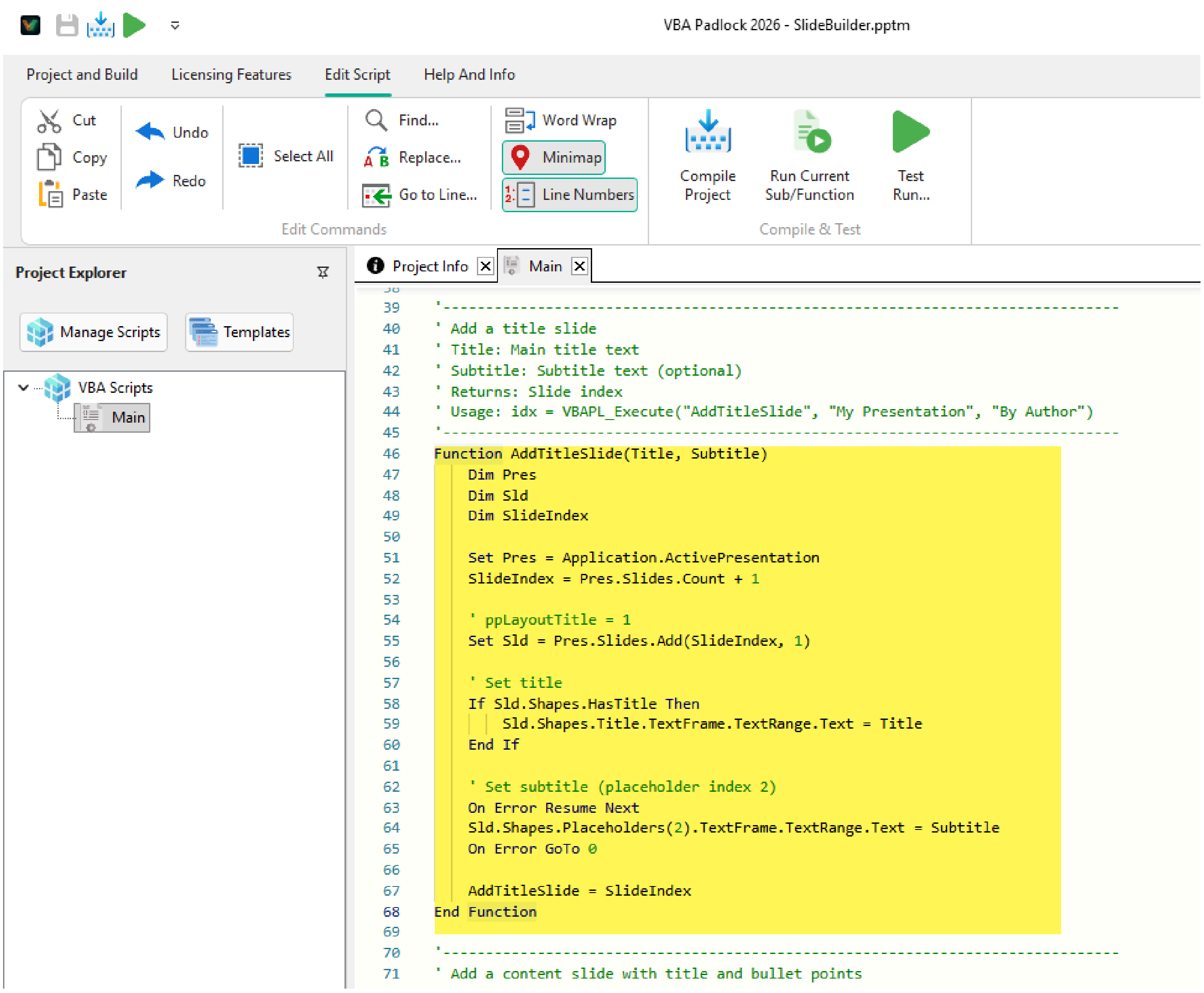
[Info] File watcher started for external script files.

17.3 Step 2: Write Your Protected Scripts

Switch to the **Code Editor**. This is where you write the VBA logic you want to hide. Below is an excerpt based on the real *SlideBuilder* example.

ⓘ **Note**

Code compiled into the DLL accesses the PowerPoint object model through the `Application` COM object. This means your DLL functions can create slides, add shapes, and export to PDF directly, provided they use the `Application.` prefix. See Office Object Model Access for details.



```
'--------------------------------------------------------------------------------
' Add a title slide
' Title: Main title text
' Subtitle: Subtitle text (optional)
' Returns: Slide index
' Usage: idx = VBAPL_Execute("AddTitleSlide", "My Presentation", "By Author")
'--------------------------------------------------------------------------------
Function AddTitleSlide(Title, Subtitle)
    Dim Pres
    Dim Sld
    Dim SlideIndex

    Set Pres = Application.ActivePresentation
    SlideIndex = Pres.Slides.Count + 1

    ' ppLayoutTitle = 1
    Set Sld = Pres.Slides.Add(SlideIndex, 1)

    ' Set title
    If Sld.Shapes.HasTitle Then
        Sld.Shapes.Title.TextFrame.TextRange.Text = Title
    End If

    ' Set subtitle (placeholder index 2)
    On Error Resume Next
    Sld.Shapes.Placeholders(2).TextFrame.TextRange.Text = Subtitle
    On Error GoTo 0

    AddTitleSlide = SlideIndex
End Function

'--------------------------------------------------------------------------------
' Add a content slide with title and bullet points
```

```vb
' PROTECTED CODE (Inside VBA Padlock)
' These functions access PowerPoint's object model via the "Application"
object.

' Add a title slide to the active presentation
Function AddTitleSlide(Title, Subtitle)
    Dim Pres, Sld, SlideIndex
    Set Pres = Application.ActivePresentation
    SlideIndex = Pres.Slides.Count + 1

    ' Create slide with Title layout (1)
    Set Sld = Pres.Slides.Add(SlideIndex, 1)

    ' Set title text
    If Sld.Shapes.HasTitle Then
        Sld.Shapes.Title.TextFrame.TextRange.Text = Title
    End If

    ' Set subtitle (usually placeholder index 2)
    On Error Resume Next
    Sld.Shapes.Placeholders(2).TextFrame.TextRange.Text = Subtitle
    On Error GoTo 0

    AddTitleSlide = SlideIndex
End Function

' Add a content slide with bullet points
Function AddContentSlide(Title, BulletPoints)
    Dim Pres, Sld, SlideIndex, i, Text
    Set Pres = Application.ActivePresentation
    SlideIndex = Pres.Slides.Count + 1

    ' Create slide with Title and Content layout (2)
    Set Sld = Pres.Slides.Add(SlideIndex, 2)
    Sld.Shapes.Title.TextFrame.TextRange.Text = Title

    ' Add bullet points from array
    Text = ""
    For i = LBound(BulletPoints) To UBound(BulletPoints)
        Text = Text & BulletPoints(i) & Chr(13)
    Next i
    Sld.Shapes.Placeholders(2).TextFrame.TextRange.Text = Text

    AddContentSlide = SlideIndex
End Function
```

17.4 Step 3: Compile and Publish

Build the protected DLL that will replace your source code.

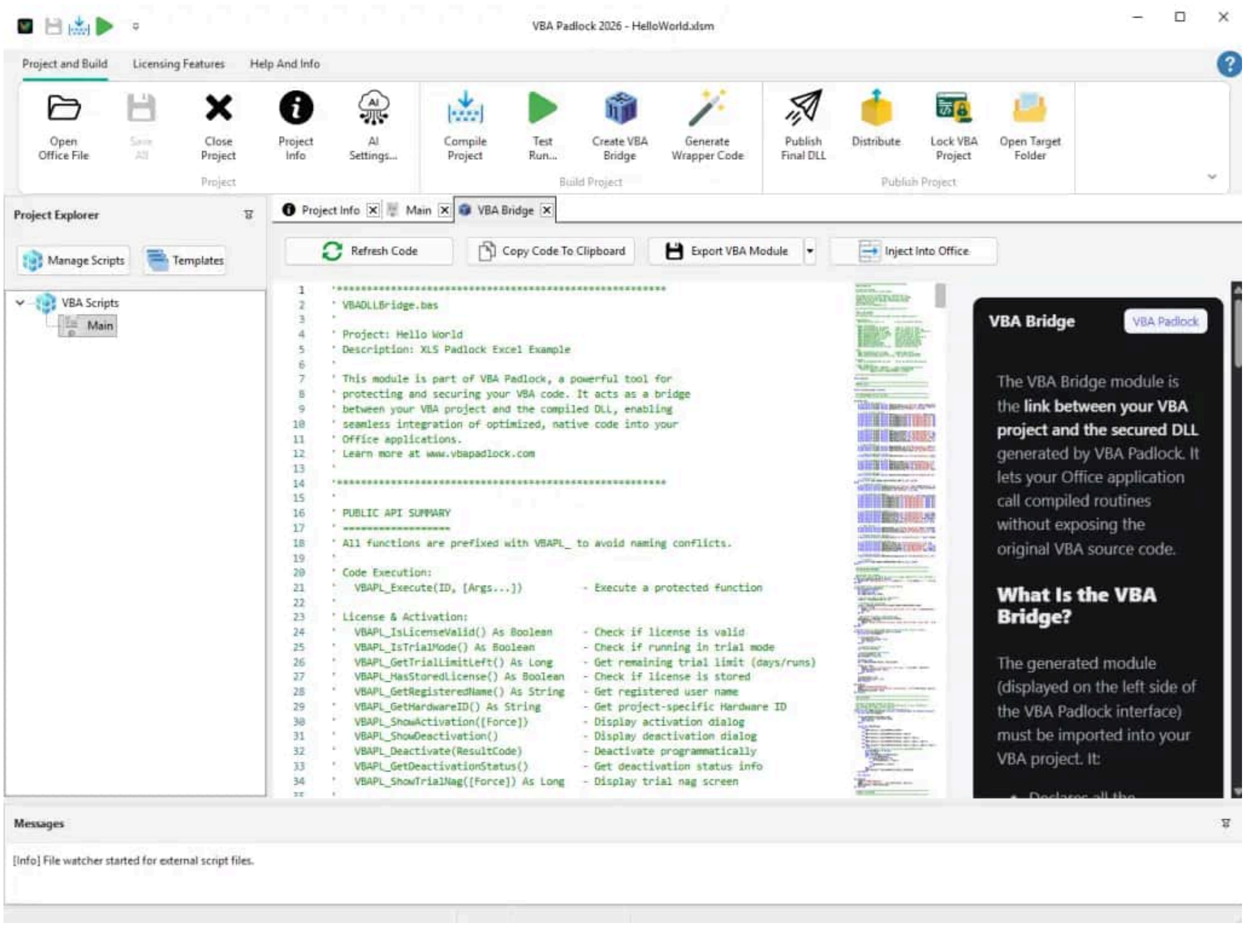
1. **Click "Publish Final DLL"** in the ribbon.
2. **Click "Build Final DLL Files"**.
3. VBA Padlock produces three DLLs in a `bin` subfolder next to your document:
   - `SlideBuilderrun32.dll` (32-bit runtime)
   - `SlideBuilderrun64.dll` (64-bit runtime)
   - `SlideBuilder.dll` (Your compiled code)

17.5 Step 4: Inject the VBA Bridge

The VBA Bridge is the connection between your PowerPoint file and the DLL.

1. Open your PowerPoint file in Microsoft PowerPoint.
2. In VBA Padlock, click **Create VBA Bridge → Inject Into Office**.

3 VBA Padlock adds the `VBADLLBridge` module to your PowerPoint project.



```
'*********************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*********************************************************
'
' PUBLIC API SUMMARY
' ------------------
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean        - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is stored
'   VBAPL_GetRegisteredName() As String   - Get registered user name
'   VBAPL_GetHardwareID() As String       - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])         - Display activation dialog
'   VBAPL_ShowDeactivation()              - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)          - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()         - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long   - Display trial nag screen
```

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

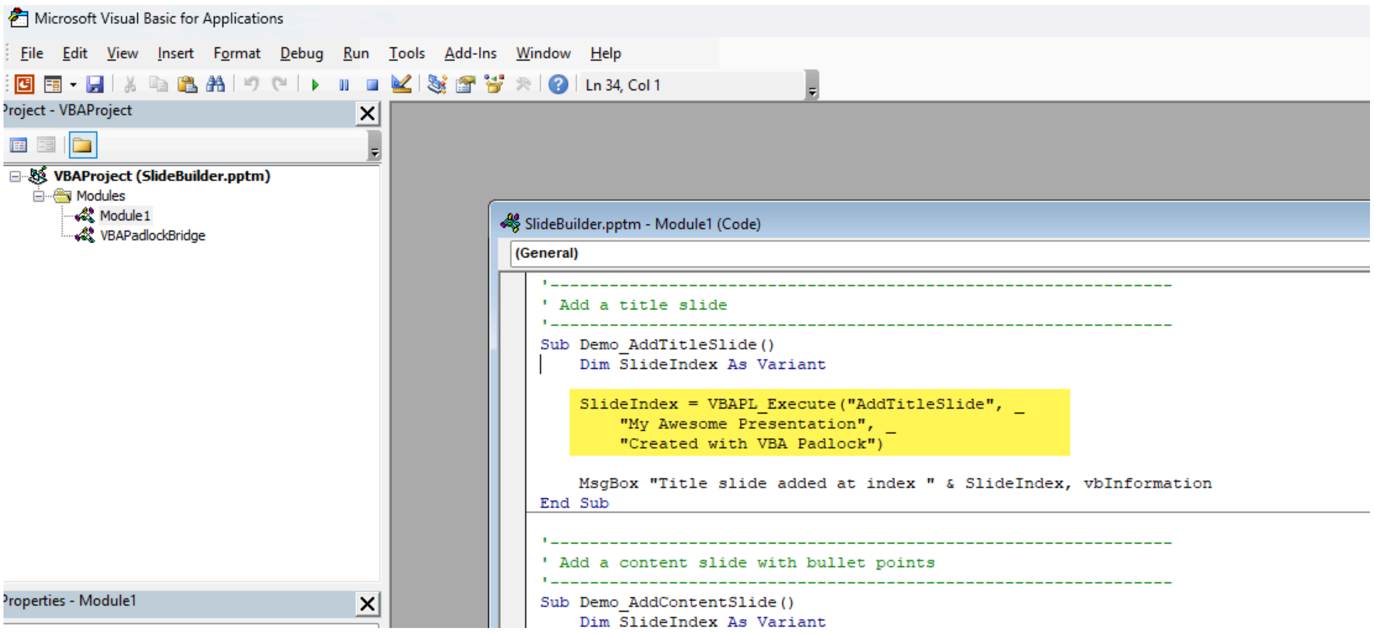
What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

[Info] File watcher started for external script files.

17.6 Step 5: Write Caller Code in PowerPoint

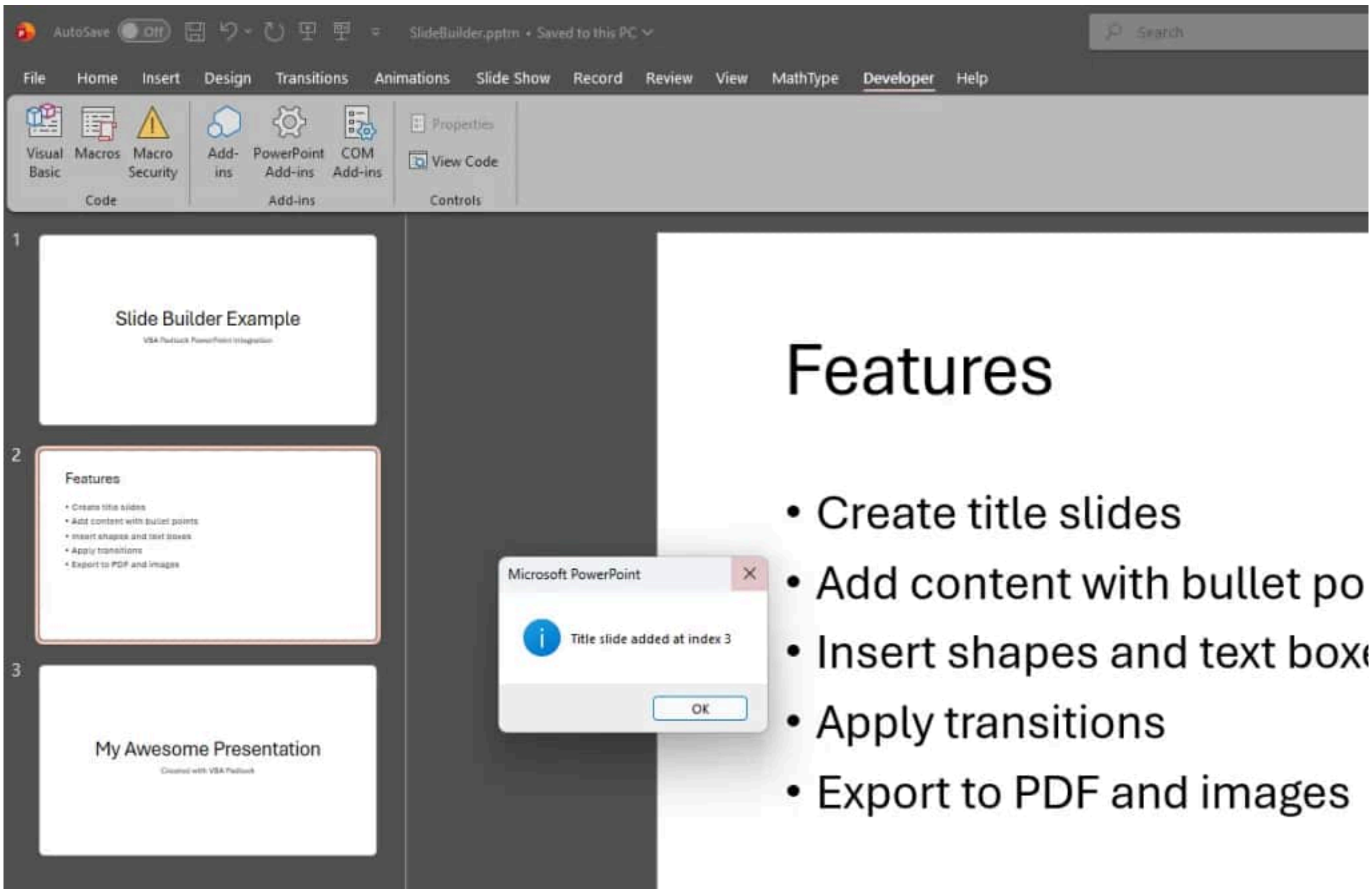
Now you can call your compiled functions from regular PowerPoint VBA code using `VBAPL_Execute`.



```
'-----------------------------------------------------------------
' Add a title slide
'-----------------------------------------------------------------
Sub Demo_AddTitleSlide()
    Dim SlideIndex As Variant

    SlideIndex = VBAPL_Execute("AddTitleSlide", _
        "My Awesome Presentation", _
        "Created with VBA Padlock")

    MsgBox "Title slide added at index " & SlideIndex, vbInformation
End Sub

'-----------------------------------------------------------------
' Add a content slide with bullet points
'-----------------------------------------------------------------
Sub Demo_AddContentSlide()
    Dim SlideIndex As Variant
```

In the PowerPoint VBA Editor (`Alt+F11`), create a new module and add this:

```
'----------------------------------------------------------------
' Call the compiled DLL functions
'----------------------------------------------------------------

Sub Demo_AddTitleSlide()
    Dim SlideIndex As Variant

    ' Call the protected DLL function
    SlideIndex = VBAPL_Execute("AddTitleSlide", _
        "My Awesome Presentation", _
        "Created with VBA Padlock")

    MsgBox "Title slide added at index " & SlideIndex, vbInformation
End Sub

Sub Demo_AddContentSlide()
    Dim SlideIndex As Variant
    Dim Items As Variant

    Items = Array("Protected Business Logic", _
                  "Native DLL Performance", _
                  "Secure Hardware Locking")

    ' Call the protected DLL function
    SlideIndex = VBAPL_Execute("AddContentSlide", "Key Features", Items)

    MsgBox "Content slide added at index " & SlideIndex, vbInformation
End Sub
```



Microsoft PowerPoint

Title slide added at index 3

Features

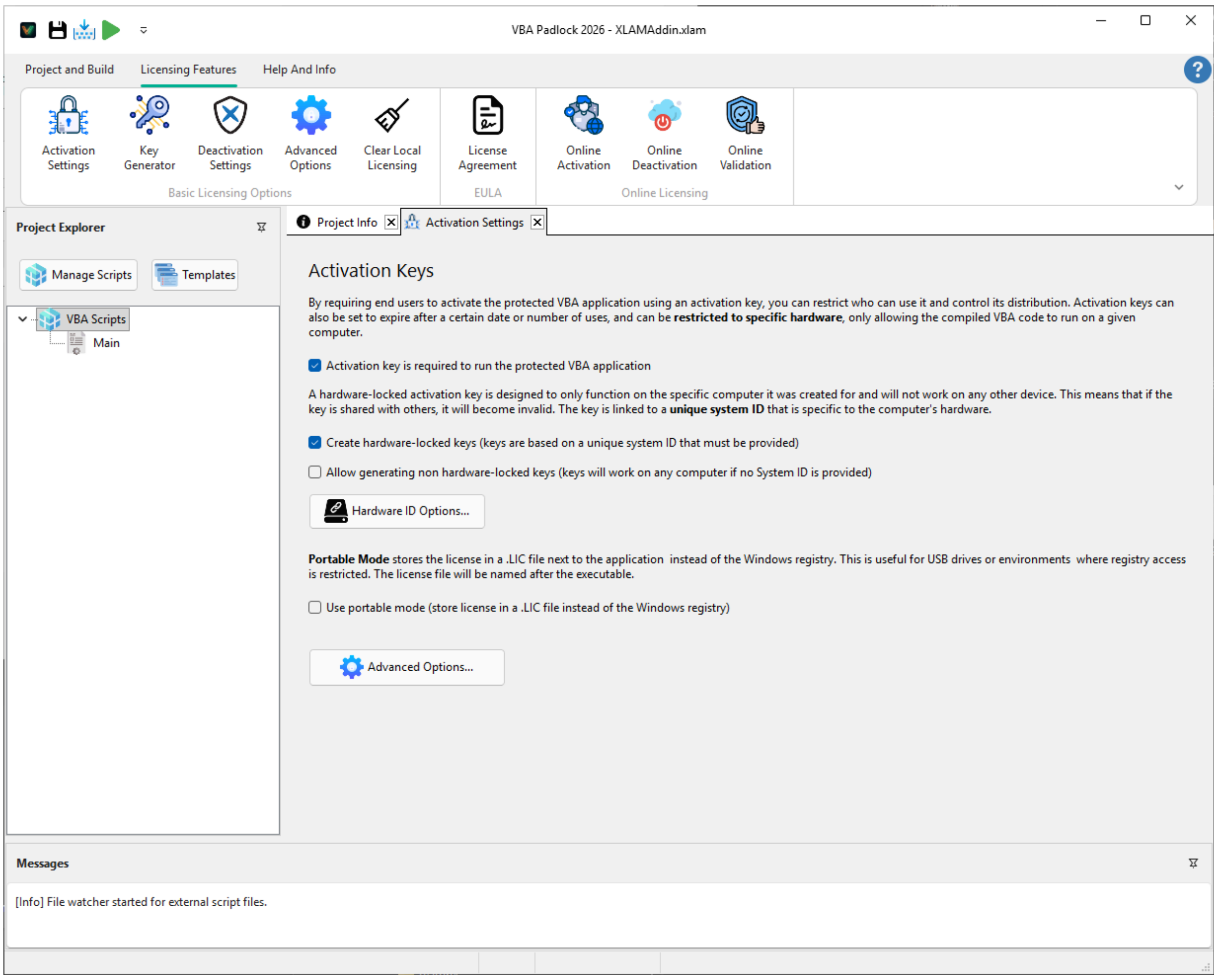
- Create title slides
- Add content with bullet po
- Insert shapes and text boxe
- Apply transitions
- Export to PDF and images

17.7 Step 6: Licensing (Optional)

To require an activation key before your presentation can be used:

1. Go to **Licensing Features** → Activation Settings.
2. Check **Activation key is required**.

3 Optional: Enable **Hardware Locking** to tie licenses to specific PCs.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

[✓] Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

[✓] Create hardware-locked keys (keys are based on a unique system ID that must be provided)

[ ] Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

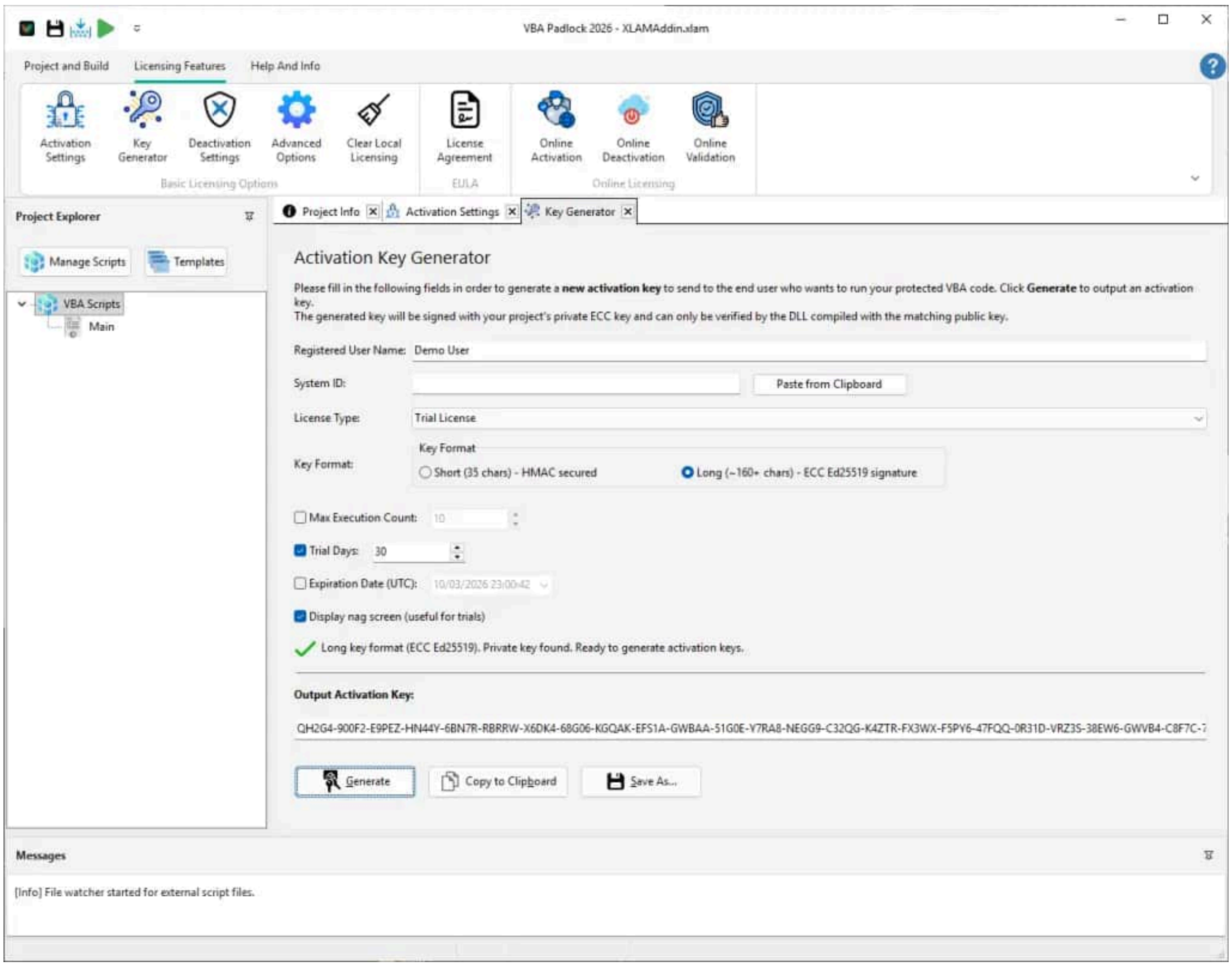
Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

[ ] Use portable mode (store license in a .LIC file instead of the Windows registry)

[Info] File watcher started for external script files.

4 **Rebuild** the DLL to apply the security settings.

5 Generate a test key in the **Key Generator**.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format: Key Format
( ) Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

[Info] File watcher started for external script files.

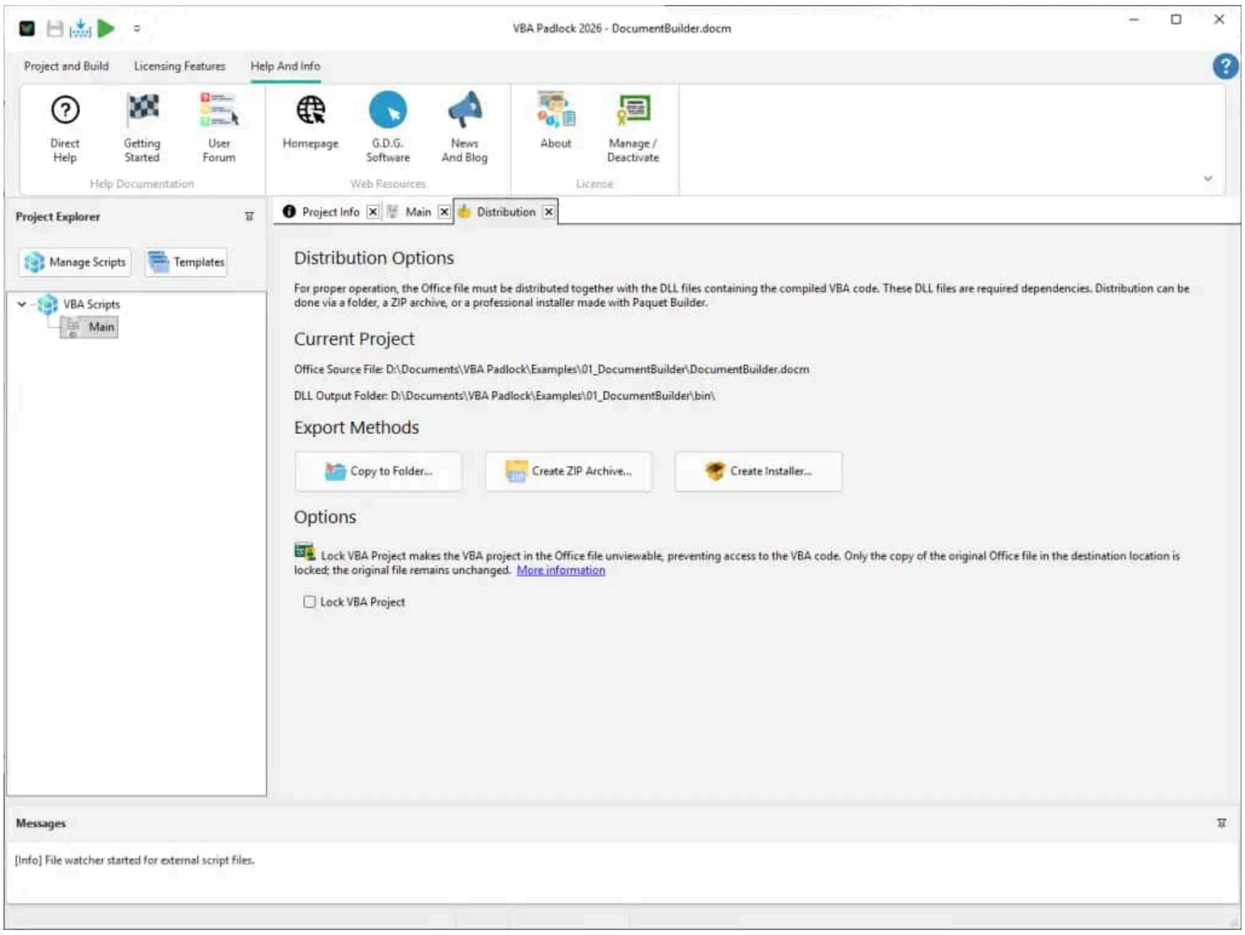
17.8 Step 7: Distribution

Package your application for release.

1 Click **Distribute** in the ribbon.

2 Select **Create ZIP Archive**.

3 The archive will contain your `.pptm` file and the `bin` folder (mandatory).



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Copy to Folder... Create ZIP Archive... Create Installer...

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

[Info] File watcher started for external script files.

17.9 Next Steps

Protect Word

Similar workflow tailored for Word documents and templates. Read the guide →

Online Activation

Automate license activation over the internet. Read the guide →

VBA Bridge API

Full reference for all `VBAPL_*` functions. View API reference →

VBA Compatibility

Check supported VBA features and statements. View reference →

18. Protect an Access Database

You have built a Microsoft Access database packed with VBA logic — maybe it handles complex data exports, manages relational integrity, or runs proprietary SQL utilities — and you need to distribute it without exposing your source code.

In this tutorial, you'll walk through the complete workflow: compiling your Access logic into a protected DLL, wiring it up to Access through the auto-generated VBA Bridge, and optionally adding activation keys with hardware locking.

Time needed: 20-25 minutes

Tip

Before you start, make sure you have:

- VBA Padlock installed (installation guide)
- A Microsoft Access database (`.accdb` file)
- Basic familiarity with the Access VBA Editor (`Alt+F11`)

18.1 The Scenario: Data Export Toolkit

Imagine you sell a **"Data Export Toolkit"**: users open your database, click a button, and the macro runs a series of SQL queries, formats the data, and exports it to CSV or JSON using optimized routines.

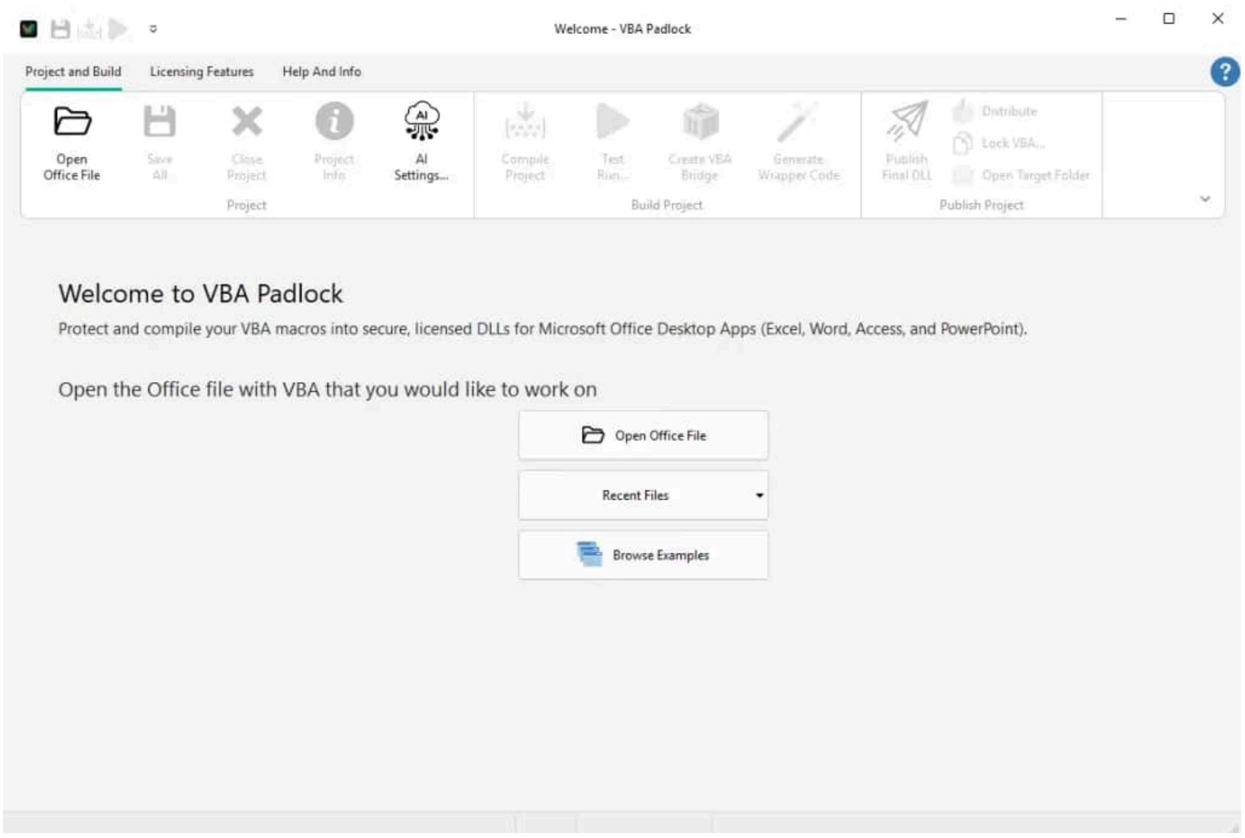
VBA Padlock allows you to commercialize this tool while keeping your proprietary SQL logic and export algorithms completely hidden inside a native DLL.

18.2 Step 1: Create a VBA Padlock Project

Start by opening your Access file in VBA Padlock to create a project that will hold your compiled code and licensing settings.

1. **Launch VBA Padlock** from the Start menu.

2 **Click “Open Office File”** in the ribbon and select your `.accdb` file (for example, `DataExport.accdb`).



Welcome to VBA Padlock

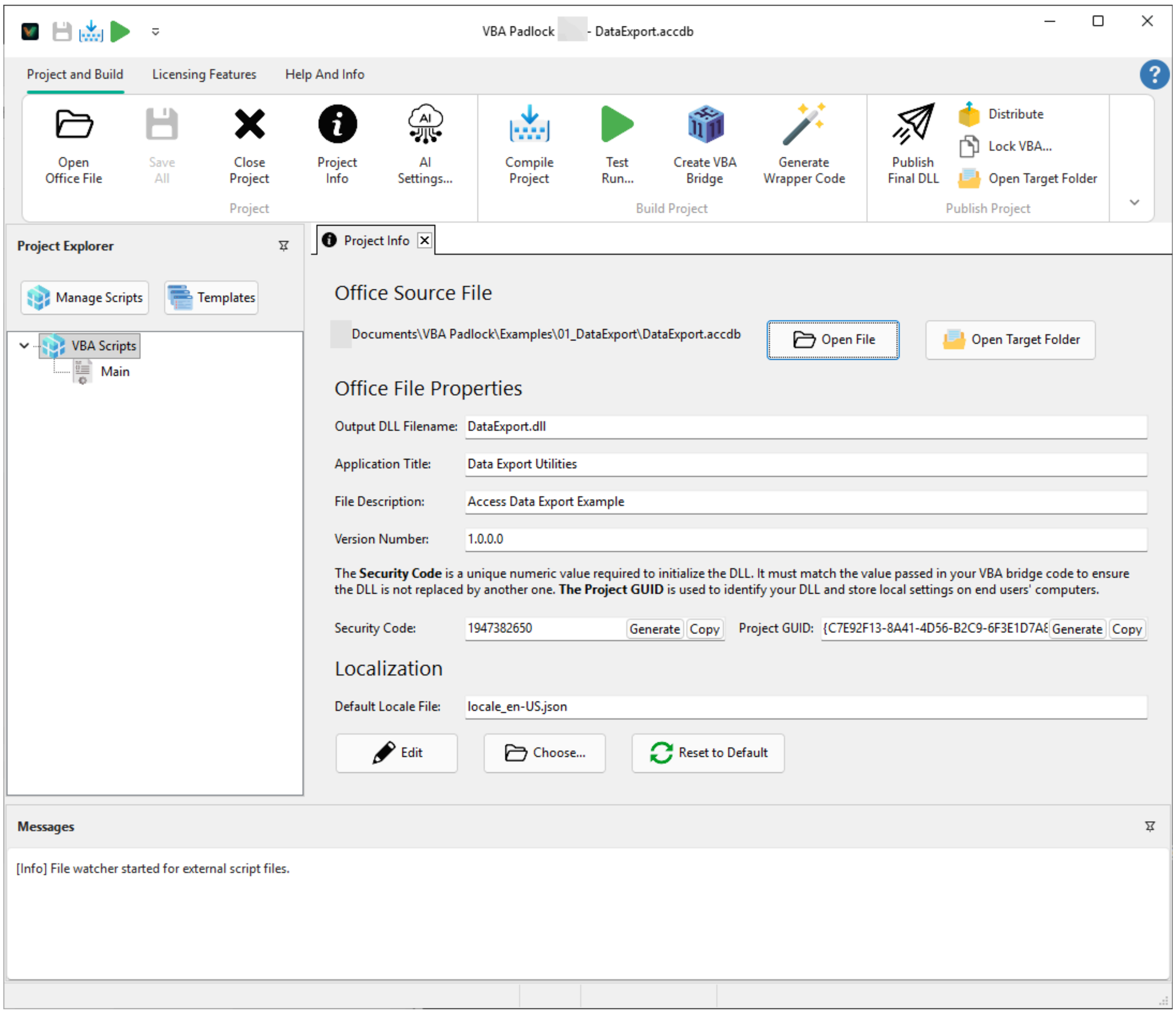
Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

3 **Set Project Information**

In the **Project Info** tab:

- **Application Title:** Give your tool a name (e.g., `Data Export Utilities`).
- **Security Code:** Click **Generate** to create the unique cryptographic link between your database and the DLL.



Office Source File

Documents\VBA Padlock\Examples\01_DataExport\DataExport.accdb

Office File Properties

Output DLL Filename:	DataExport.dll
Application Title:	Data Export Utilities
File Description:	Access Data Export Example
Version Number:	1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 1947382650 Project GUID: {C7E92F13-8A41-4D56-B2C9-6F3E1D7A

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

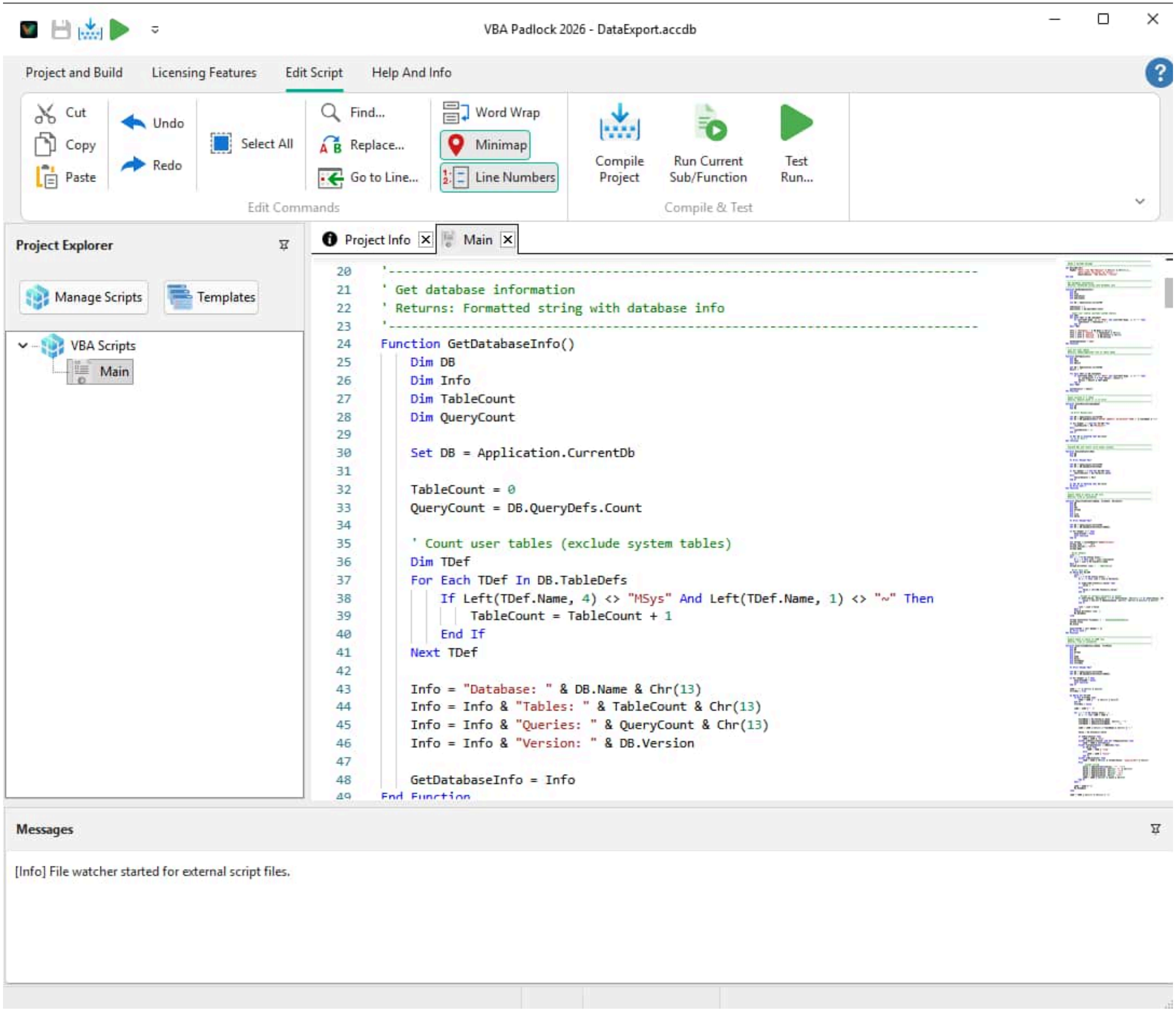
18.3 Step 2: Write Your Protected Scripts

Switch to the **Code Editor**. This is where you write the VBA logic you want to hide. Below is an excerpt based on the real *DataExport* example included with VBA Padlock.

Tip

Compiled scripts have full access to the host application through the `Application` object. See Office Object Model Access.

Important: Inside the VBA Padlock editor (protected code), you **must** use `Application.CurrentDb` (not just `CurrentDb`). In your Access VBE (unprotected code), you continue to use standard VBA.



```
'-------------------------------------------------------------------------------
' Get database information
' Returns: Formatted string with database info
'-------------------------------------------------------------------------------
Function GetDatabaseInfo()
    Dim DB
    Dim Info
    Dim TableCount
    Dim QueryCount

    Set DB = Application.CurrentDb

    TableCount = 0
    QueryCount = DB.QueryDefs.Count

    ' Count user tables (exclude system tables)
    Dim TDef
    For Each TDef In DB.TableDefs
        If Left(TDef.Name, 4) <> "MSys" And Left(TDef.Name, 1) <> "~" Then
            TableCount = TableCount + 1
        End If
    Next TDef

    Info = "Database: " & DB.Name & Chr(13)
    Info = Info & "Tables: " & TableCount & Chr(13)
    Info = Info & "Queries: " & QueryCount & Chr(13)
    Info = Info & "Version: " & DB.Version

    GetDatabaseInfo = Info
End Function
```

```
[Info] File watcher started for external script files.
```

```
' PROTECTED CODE (Inside VBA Padlock)
' These functions access Access's database engine via the
"Application.CurrentDb" object.

Sub HelloWorld()
    MsgBox "Hello from VBA Padlock!" & Chr(13) & Chr(13) & _
           "Access DataExport is ready.", _
           vbInformation, "VBA Padlock - Access"
End Sub

' Get current database information
Function GetDatabaseInfo()
    Dim DB
    Dim Info
    Dim TableCount
    Dim QueryCount

    Set DB = Application.CurrentDb

    TableCount = 0
    QueryCount = DB.QueryDefs.Count

    ' Count user tables (exclude system tables)
    Dim TDef
    For Each TDef In DB.TableDefs
        If Left(TDef.Name, 4) <> "MSys" And Left(TDef.Name, 1) <> "~" Then
            TableCount = TableCount + 1
        End If
    Next TDef

    Info = "Database: " & DB.Name & Chr(13)
    Info = Info & "Tables: " & TableCount & Chr(13)
    Info = Info & "Queries: " & QueryCount & Chr(13)
    Info = Info & "Version: " & DB.Version

    GetDatabaseInfo = Info
End Function

' Return a comma-separated list of user tables
Function GetTableList()
```

```
    Dim DB, TDef, Result
    Set DB = Application.CurrentDb
    Result = ""

    For Each TDef In DB.TableDefs
        ' Filter out system and temp tables
        If Left(TDef.Name, 4) <> "MSys" And Left(TDef.Name, 1) <> "~" Then
            If Len(Result) > 0 Then Result = Result & ","
            Result = Result & TDef.Name
        End If
    Next TDef

    GetTableList = Result
End Function

' Export a table to CSV
Function ExportToCSV(Table, Path, Delimiter)
    Dim DB, RS, Stream, i, Line, Value
    On Error Resume Next
    Set DB = Application.CurrentDb
    Set RS = DB.OpenRecordset(Table)

    If Err.Number <> 0 Then ExportToCSV = False: Exit Function

    Set Stream = CreateObject("ADODB.Stream")
    Stream.Type = 2 : Stream.Charset = "UTF-8" : Stream.Open

    ' Headers and Data rows logic...
    ' (Simplified for the tutorial, full implementation in DataExport example)
    Stream.WriteText "Header1" & Delimiter & "Header2", 1
    Stream.WriteText "Value1" & Delimiter & "Value2", 1

    Stream.SaveToFile Path, 2
    Stream.Close: RS.Close
    ExportToCSV = (Err.Number = 0)
End Function
```

18.3.1 Backup and SQL functions

```
Function BackupTable(TableName, BackupSuffix)
    Dim BackupName
    Dim SQL

    On Error Resume Next

    BackupName = TableName & BackupSuffix & "_" & Format(Now,
"yyyymmdd_hhnnss")

    ' Delete if exists
    Application.DoCmd.DeleteObject 0, BackupName  ' acTable = 0
    Err.Clear

    ' Create backup using SELECT INTO
    SQL = "SELECT * INTO [" & BackupName & "] FROM [" & TableName & "]"
    Application.CurrentDb.Execute SQL

    BackupTable = (Err.Number = 0)
    On Error GoTo 0
End Function

Function ExportAllTablesToCSV(FolderPath, Delimiter)
    Dim DB
    Dim TDef
    Dim Count
    Dim FilePath

    Set DB = Application.CurrentDb
    Count = 0

    If Right(FolderPath, 1) <> "\" Then
        FolderPath = FolderPath & "\"
    End If

    For Each TDef In DB.TableDefs
        If Left(TDef.Name, 4) <> "MSys" And Left(TDef.Name, 1) <> "~" Then
            FilePath = FolderPath & TDef.Name & ".csv"
            If ExportToCSV(TDef.Name, FilePath, Delimiter) Then
                Count = Count + 1
            End If
        End If
    Next TDef

    ExportAllTablesToCSV = Count
End Function

Function ExecuteScalar(SQL)
    Dim DB
    Dim RS

    On Error Resume Next

    Set DB = Application.CurrentDb
    Set RS = DB.OpenRecordset(SQL)

    If Err.Number = 0 And Not RS.EOF Then
        ExecuteScalar = RS.Fields(0).Value
    Else
        ExecuteScalar = Null
    End If

    If Not RS Is Nothing Then RS.Close
    On Error GoTo 0
End Function

Function ExecuteSQL(SQL)
    Dim DB

    On Error Resume Next

    Set DB = Application.CurrentDb
    DB.Execute SQL, 128  ' dbFailOnError = 128

    If Err.Number = 0 Then
        ExecuteSQL = DB.RecordsAffected
    Else
        ExecuteSQL = -1
    End If

    On Error GoTo 0
```

```
End Function
```

Tip

Notice that the compiled scripts use `Application.CurrentDb` to access the database. VBA Padlock automatically provides the Access `Application` COM object to your compiled code, so you can call `CurrentDb`, `DoCmd`, and other Access objects exactly as you would in normal VBA.

18.4 Step 4: Build and Publish the DLL

1. Click **Publish Final DLL** in the ribbon, then click **Build Final DLL Files** on the page that opens.
2. VBA Padlock compiles your scripts and creates three DLL files in a `bin` subfolder next to your database:

```
MyProject/
├── MyDatabase.accdb
├── MyDatabase.vbapadlock/
└── bin/
    ├── MyDatabaserun32.dll     ← 32-bit runtime
    ├── MyDatabaserun64.dll     ← 64-bit runtime
    └── MyDatabase.dll          ← Your compiled scripts (satellite
DLL)
```

3. The runtime DLLs are digitally signed by G.D.G. Software. The satellite DLL is integrity-verified by an internal cryptographic signature.

Note

The naming convention is `{DLLName}run32.dll` and `{DLLName}run64.dll` for the runtimes, and `{DLLName}.dll` for the satellite DLL containing your compiled code. The DLL name is set in Project Info.

18.5 Step 5: Inject the VBA Bridge into Access

Now connect the compiled DLL to your Access database:

1. Open `MyDatabase.accdb` in Microsoft Access.
2. Go back to VBA Padlock and click **Create VBA Bridge**, then click **Inject Into Office**.
3. VBA Padlock inserts a `VBADLLBridge` module into your Access VBA project. You can verify this by pressing `Alt+F11` to open the VBA Editor — you should see the bridge module listed under **Modules**.
4. The VBA Bridge provides the `VBAPL_Execute` function that you use to call your compiled functions from Access VBA.

⚠ **Caution**

Do not edit the `VBADLLBridge` module manually. It is generated by VBA Padlock and contains the DLL declarations, initialization logic, and security handshake. If you need to regenerate it, use **Create VBA Bridge** again.

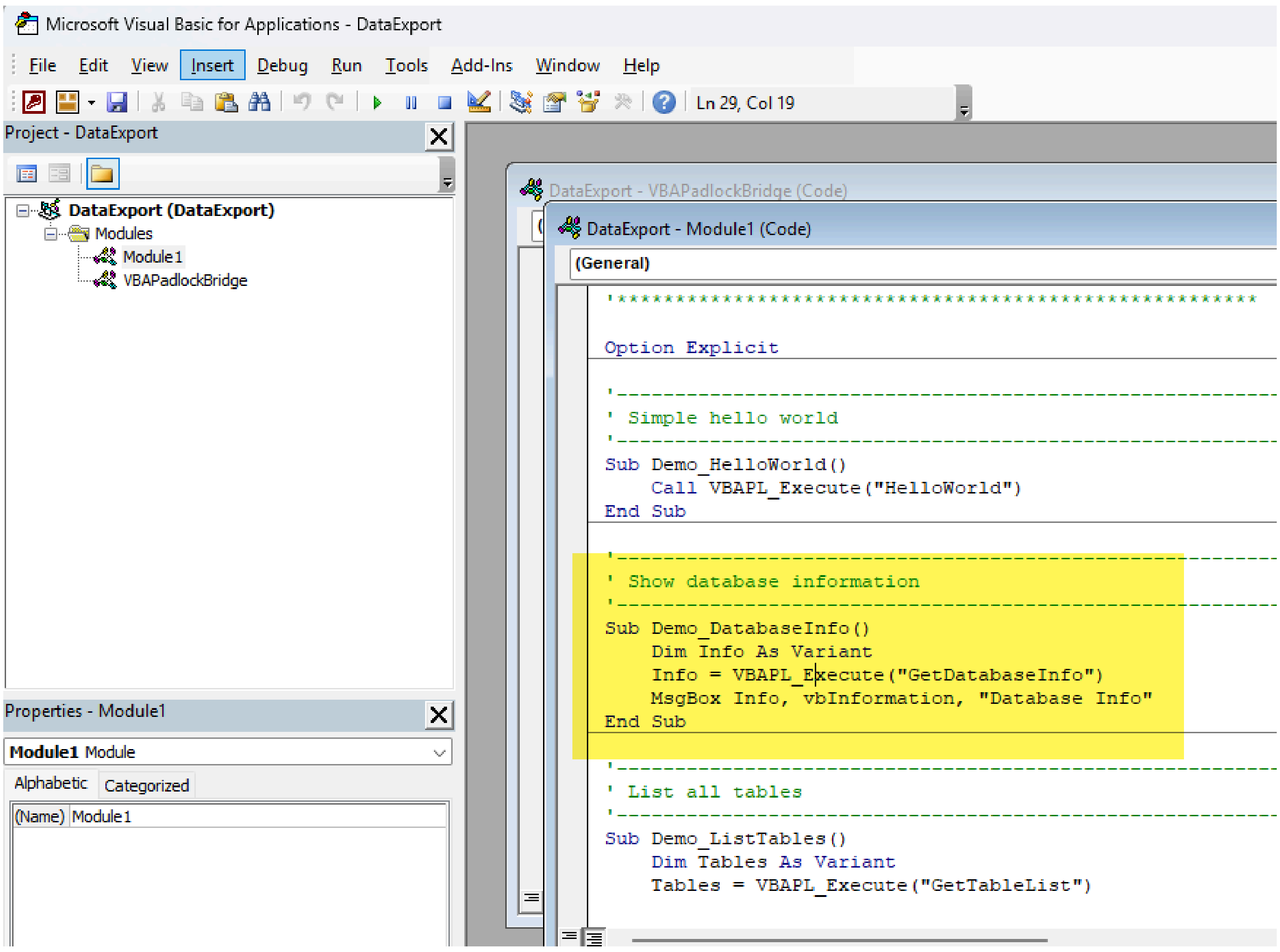
18.6 Step 6: Call Protected Functions from Access

Now that the bridge is in place, you can call your compiled functions from any Access VBA module, form, or report. The pattern is always:

```
result = VBAPL_Execute("FunctionName", arg1, arg2, ...)
```

18.6.2 Import the example usage module

In the VBA Editor (`Alt+F11`), go to **File > Import File** and import `ExampleUsage.bas`. This module contains ready-to-run demo subroutines. Here are the key examples:

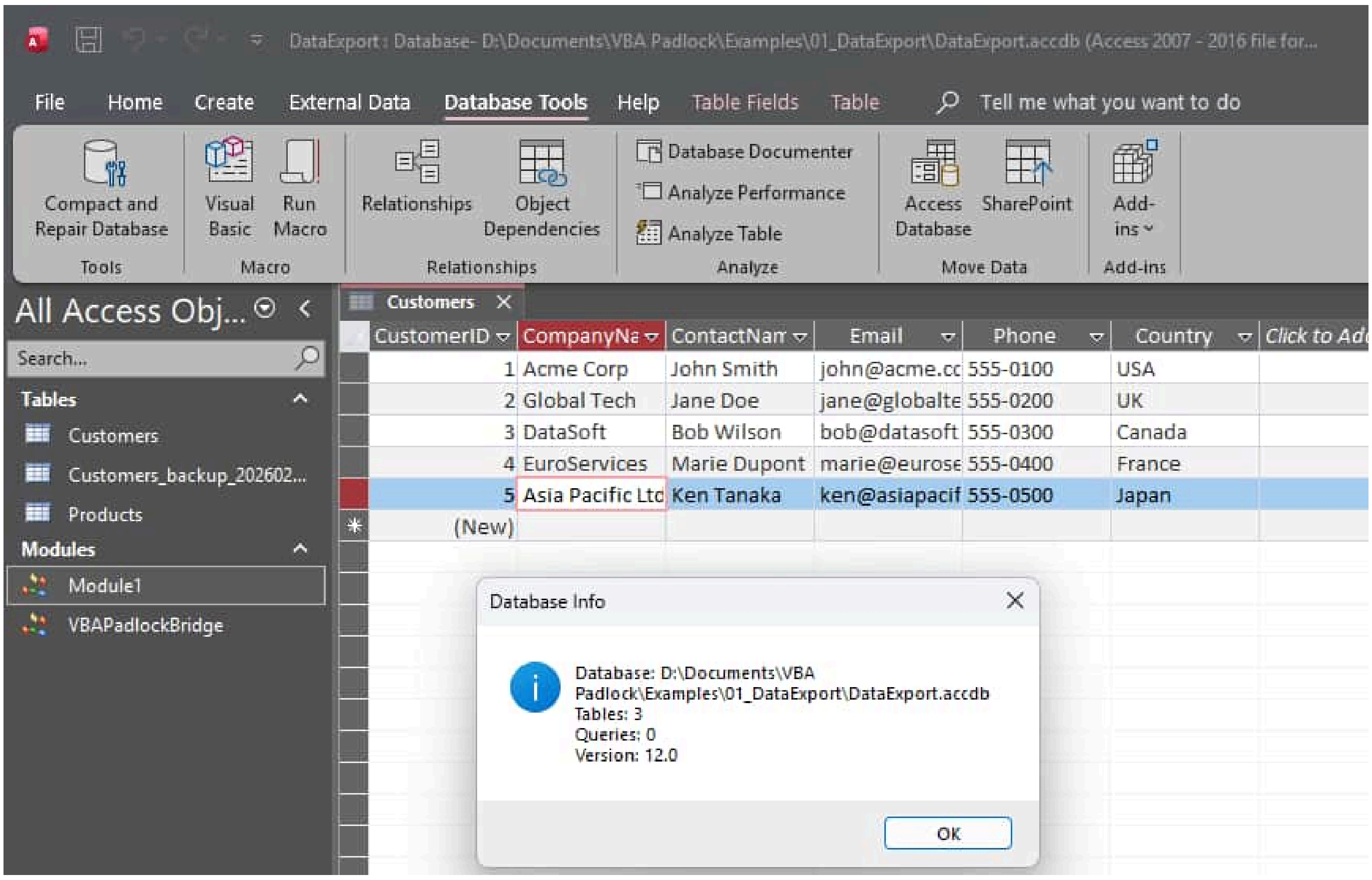

Microsoft Visual Basic for Applications - DataExport
File Edit View Insert Debug Run Tools Add-Ins Window Help
Ln 29, Col 19
Project - DataExport
DataExport (DataExport)
Modules
Module1
VBAPadlockBridge
Properties - Module1
Module1 Module
Alphabetic Categorized
(Name) Module1
DataExport - VBAPadlockBridge (Code)
DataExport - Module1 (Code)
(General)

```
'*********************************************************************

Option Explicit

'-----------------------------------------------------------------------
' Simple hello world
'-----------------------------------------------------------------------
Sub Demo_HelloWorld()
    Call VBAPL_Execute("HelloWorld")
End Sub

'-----------------------------------------------------------------------
' Show database information
'-----------------------------------------------------------------------
Sub Demo_DatabaseInfo()
    Dim Info As Variant
    Info = VBAPL_Execute("GetDatabaseInfo")
    MsgBox Info, vbInformation, "Database Info"
End Sub

'-----------------------------------------------------------------------
' List all tables
'-----------------------------------------------------------------------
Sub Demo_ListTables()
    Dim Tables As Variant
    Tables = VBAPL_Execute("GetTableList")
```

18.6.3 Get database information

```
Sub Demo_DatabaseInfo()
    Dim Info As Variant
    Info = VBAPL_Execute("GetDatabaseInfo")
    MsgBox Info, vbInformation, "Database Info"
End Sub
```

This calls `GetDatabaseInfo()` in the compiled DLL, which queries `Application.CurrentDb` and returns a formatted string with the database name, table count, query count, and version.



CustomerID	CompanyNa	ContactNam	Email	Phone	Country	Click to Ad
1	Acme Corp	John Smith	john@acme.cc	555-0100	USA	
2	Global Tech	Jane Doe	jane@globalte	555-0200	UK	
3	DataSoft	Bob Wilson	bob@datasoft	555-0300	Canada	
4	EuroServices	Marie Dupont	marie@eurose	555-0400	France	
5	Asia Pacific Ltc	Ken Tanaka	ken@asiapacif	555-0500	Japan	
(New)						

Database Info

```
Database: D:\Documents\VBA
Padlock\Examples\01_DataExport\DataExport.accdb
Tables: 3
Queries: 0
Version: 12.0
```

18.6.4 Export a table to CSV

```
Sub Demo_ExportToCSV()
    Dim TableName As String
    Dim FilePath As String
    Dim Success As Variant

    TableName = InputBox("Enter table name to export:", "Export to CSV",
"Customers")
    If Len(TableName) = 0 Then Exit Sub

    FilePath = CurrentProject.Path & "\" & TableName & ".csv"

    Success = VBAPL_Execute("ExportToCSV", TableName, FilePath, ";")

    If Success Then
        MsgBox "Exported to:" & vbCrLf & FilePath, vbInformation, "Export
Complete"
    Else
        MsgBox "Export failed. Check if table exists.", vbExclamation
    End If
End Sub
```

Notice the three arguments passed to `ExportToCSV`: the table name, the output file path, and the delimiter character. The function returns `True` on success and `False` on failure.

18.6.5 Export a table to JSON

```
Sub Demo_ExportToJSON()
    Dim TableName As String
    Dim FilePath As String
    Dim Success As Variant

    TableName = InputBox("Enter table name to export:", "Export to JSON",
"Customers")
    If Len(TableName) = 0 Then Exit Sub

    FilePath = CurrentProject.Path & "\" & TableName & ".json"

    Success = VBAPL_Execute("ExportToJSON", TableName, FilePath)

    If Success Then
        MsgBox "Exported to:" & vbCrLf & FilePath, vbInformation, "Export
Complete"
    Else
        MsgBox "Export failed. Check if table exists.", vbExclamation
    End If
End Sub
```

The JSON output is UTF-8 encoded and produces a well-formed JSON array with one object per row.

18.6.6 Create a table backup

```
Sub Demo_BackupTable()
    Dim TableName As String
    Dim Success As Variant

    TableName = InputBox("Enter table name to backup:", "Backup Table", 
"Customers")
    If Len(TableName) = 0 Then Exit Sub

    Success = VBAPL_Execute("BackupTable", TableName, "_backup")

    If Success Then
        MsgBox "Backup created successfully!", vbInformation
    Else
        MsgBox "Backup failed. Check if table exists.", vbExclamation
    End If
End Sub
```

This creates a copy of the table named `Customers_backup_20260207_143022` (with the current date and time appended automatically).

18.6.7 Export all tables at once

```
Sub Demo_ExportAllTables()
    Dim FolderPath As String
    Dim Count As Variant

    FolderPath = CurrentProject.Path & "\Export"

    ' Create folder if it does not exist
    If Dir(FolderPath, vbDirectory) = "" Then
        MkDir FolderPath
    End If

    Count = VBAPL_Execute("ExportAllTablesToCSV", FolderPath, ";")

    MsgBox Count & " table(s) exported to:" & vbCrLf & FolderPath,
vbInformation, "Export Complete"
End Sub
```

18.6.8 Run a SQL query from the DLL

```
Sub Demo_ExecuteScalar()
    Dim SQL As String
    Dim Result As Variant

    SQL = "SELECT COUNT(*) FROM Customers"
    Result = VBAPL_Execute("ExecuteScalar", SQL)

    If Not IsNull(Result) Then
        MsgBox "Result: " & CStr(Result), vbInformation
    Else
        MsgBox "Query returned no result or error.", vbExclamation
    End If
End Sub
```

18.7 Step 7: Call DLL Functions from an Access Form

A common Access pattern is to wire buttons on a form to DLL functions. Here is how you might build an export form:

```
' In a form module (e.g., frmExportTools)

Private Sub btnExportCSV_Click()
    Dim TableName As String
    Dim FilePath As String
    Dim Success As Variant

    TableName = Me.cboTableName.Value
    If Len(TableName) = 0 Then
        MsgBox "Please select a table first.", vbExclamation
        Exit Sub
    End If

    FilePath = CurrentProject.Path & "\" & TableName & ".csv"
    Success = VBAPL_Execute("ExportToCSV", TableName, FilePath, ",")

    If Success Then
        MsgBox "Exported " & TableName & " to CSV.", vbInformation
    Else
        MsgBox "Export failed.", vbCritical
    End If
End Sub

Private Sub btnExportJSON_Click()
    Dim TableName As String
    Dim FilePath As String
    Dim Success As Variant

    TableName = Me.cboTableName.Value
    If Len(TableName) = 0 Then
        MsgBox "Please select a table first.", vbExclamation
        Exit Sub
    End If

    FilePath = CurrentProject.Path & "\" & TableName & ".json"
    Success = VBAPL_Execute("ExportToJSON", TableName, FilePath)

    If Success Then
        MsgBox "Exported " & TableName & " to JSON.", vbInformation
    Else
```

```
            MsgBox "Export failed.", vbCritical
        End If
    End Sub

    Private Sub btnShowInfo_Click()
        Dim Info As Variant
        Info = VBAPL_Execute("GetDatabaseInfo")
        MsgBox Info, vbInformation, "Database Info"
    End Sub

    Private Sub Form_Load()
        ' Populate the table dropdown from the DLL
        Dim Tables As Variant
        Tables = VBAPL_Execute("GetTableList")

        If Len(Tables) > 0 Then
            Dim TableArray() As String
            TableArray = Split(Tables, ",")
            Me.cboTableName.RowSourceType = "Value List"
            Me.cboTableName.RowSource = Tables
        End If
    End Sub
```

Tip

The `GetTableList()` function returns a comma-separated string, which you can assign directly to a combo box's `RowSource` property when `RowSourceType` is set to “Value List”. This lets your forms dynamically list tables without exposing the logic.

18.8 Step 8: Distribute

Use the Distribution dialog to package your database with its DLLs:

1. Click **Distribute** in the ribbon.
2. Choose **Create ZIP Archive** to generate a ready-to-ship package.

3. The ZIP contains the correct folder structure:

```
MyDatabase.accdb
bin\
├── MyDatabaserun32.dll
├── MyDatabaserun64.dll
└── MyDatabase.dll
```

4. Send the ZIP to your users. They extract it to a local folder and open the `.accdb` file.

> **Caution**
>
> **Access databases must be in a writable location.** The DLL needs to store license data alongside the database file. Make sure your users extract the ZIP to a local folder (e.g., `C:\MyTools\`) rather than opening it directly from a read-only network share or the Windows Downloads folder without extracting first.

> **Note**
>
> The `bin\` folder with all three DLLs must always be kept alongside the `.accdb` file. The VBA Bridge automatically loads the correct runtime (32-bit or 64-bit) based on the user's Office installation.

18.9 Access-Specific Considerations

18.9.9 Application object access

Your compiled scripts have full access to the Access `Application` COM object. This means you can use:

- `Application.CurrentDb` — to open recordsets, execute queries, and read table definitions
- `Application.DoCmd` — to run Access actions like `TransferSpreadsheet`, `TransferText`, and `DeleteObject`
- `Application.CurrentProject.Path` — to get the database file location

18.9.10 System tables

When iterating over `TableDefs`, always filter out Access system tables (prefixed with `MSys`) and temporary tables (prefixed with `~`). The example code demonstrates this pattern.

18.9.11 File paths

The example uses `CurrentProject.Path` from the calling VBA code to build file paths. Since `CurrentProject` is an Access VBA object (not available inside the DLL), pass the full path as a parameter to your compiled functions.

18.9.12 NULL handling

Access fields frequently contain NULL values. The export functions in this example handle NULLs explicitly — check `IsNull()` before converting field values to strings.

18.10 Troubleshooting

Problem	Solution
"Failed to initialize DLL" error	Make sure the `bin\` folder is next to the `.accdb` file and contains all three DLLs
Export functions return `False`	Verify the table name is correct and the output path is writable
`CountRecords` returns `-1`	The table does not exist or the name is misspelled
Access security warning about macros	Click "Enable Content" when opening the database, or add the folder to Access Trusted Locations
DLL not loading on another computer	The user needs to extract the full ZIP (database + `bin\` folder) to a local, writable folder

---

18.11 What's Next?

✓ Licensing Setup

Add license key activation to your Access database so only authorized users can run the exports.

Read the guide →

⚙ Key Generation

Generate and distribute license keys to your users.

Read the guide →

Protect Excel

Similar tutorial for Excel workbooks.

Read the guide →

VBA Bridge API

Complete reference for all `VBAPL_*` bridge functions.

Read the reference →

18.12 See Also

- Protect a Word Document — Word-specific tutorial
- Protect a PowerPoint Presentation — PowerPoint-specific tutorial
- Script Functions Reference — Built-in functions available inside compiled scripts
- VBA Compatibility — Supported VBA features and limitations
- Distribution — Packaging your project for end users

19. Setting Up Licensing

You have compiled your VBA code into a protected DLL—now it's time to turn it into a commercial product or a controlled corporate tool. This guide walks you through setting up a complete licensing system, from basic activation to advanced hardware locking.

Time needed: 15-20 minutes

Tip

Before you start, make sure you have:

- A VBA Padlock project with compiled scripts.
- Basic familiarity with the Licensing Features overview.

19.1 The Goal: Control Your Distribution

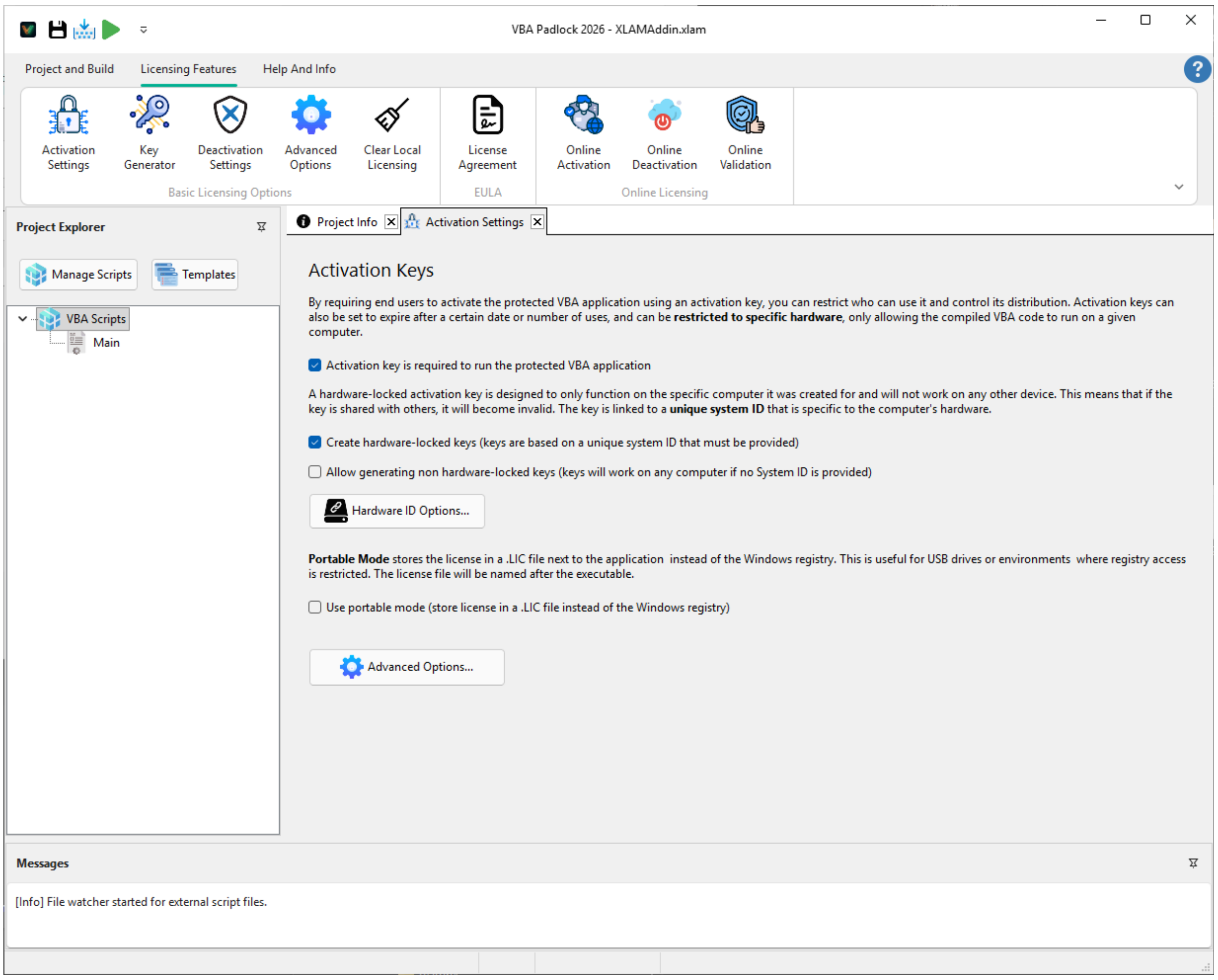
Whether you want to offer a **30-day trial** or require a **unique key for every PC**, VBA Padlock gives you the tools to enforce your licensing terms without writing complex security code in VBA.

19.2 Step 1: Enable Basic Activation

The first step is to tell VBA Padlock that your application requires a license key to run.

1. Open your project in VBA Padlock.
2. Navigate to **Licensing Features** → Activation Settings.

3 Check **Activation key is required**.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

☑ Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

☑ Create hardware-locked keys (keys are based on a unique system ID that must be provided)

☐ Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

☐ Use portable mode (store license in a .LIC file instead of the Windows registry)

[Info] File watcher started for external script files.

4 Choose your **License Storage**:

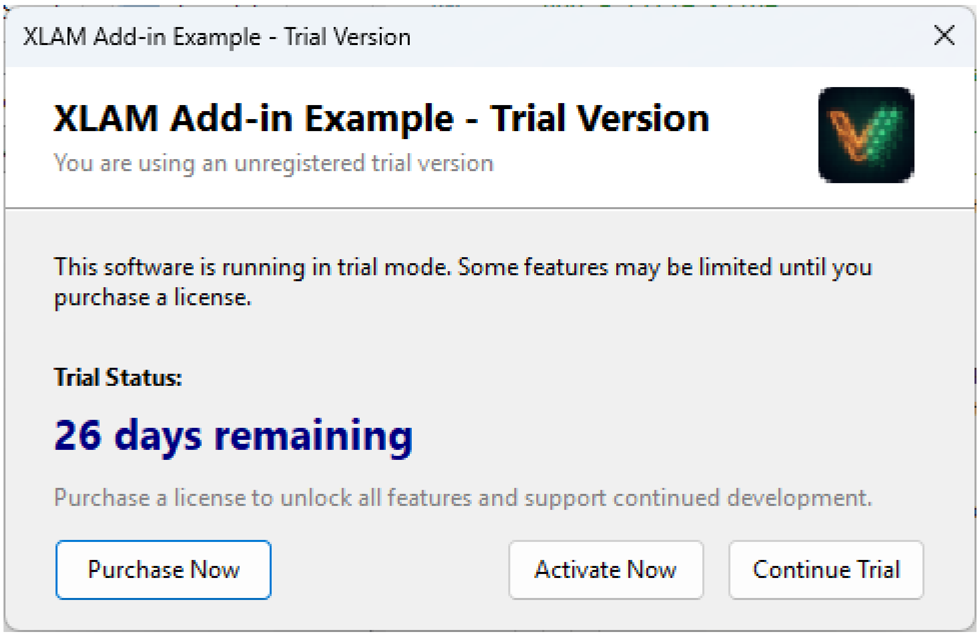
- **Registry (Default):** Stores license data in `HKEY_CURRENT_USER`. Best for standard Windows installations.
- **Portable file (.lic):** Stores the license in a file next to your document. Perfect for USB drives or "No Registry" environments.

19.3 Step 2: Configure the User Experience

How should the user see the license check? You can offer a trial period or force activation immediately.

Users can evaluate your tool before entering a full license key.

1. Go to Advanced Activation.
2. Check **Allow compiled VBA code to run without activation**. This allows the application to execute even without a license.
3. **Important:** In this unlicensed state, the application runs **silently**. No “Nag Screen” is shown automatically.
4. To display the remaining days and a purchase button (the Nag Screen), you must provide the user with a **Trial Key** generated in the Key Generator.
5. Alternatively, you can use the VBA API to create your own custom trial logic.


XLAM Add-in Example - Trial Version

XLAM Add-in Example - Trial Version

You are using an unregistered trial version

This software is running in trial mode. Some features may be limited until you purchase a license.

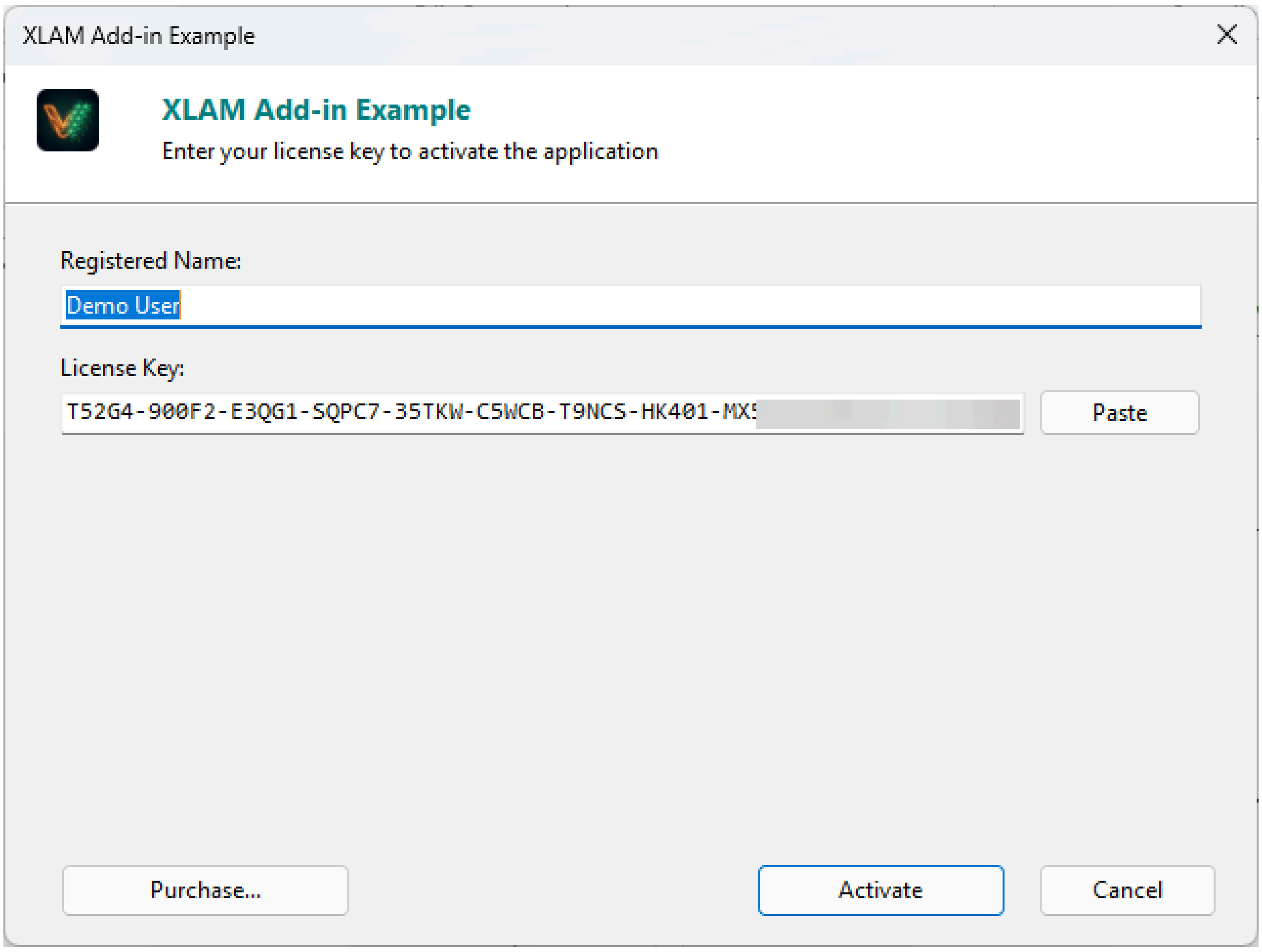
Trial Status:

26 days remaining

Purchase a license to unlock all features and support continued development.

The application will not run at all without a valid key.

1. In **Advanced Activation**, select **Always prompt for activation**.
2. This is the preferred mode for B2B tools or high-value software.


XLAM Add-in Example

XLAM Add-in Example

Enter your license key to activate the application

Registered Name:

Demo User

License Key:

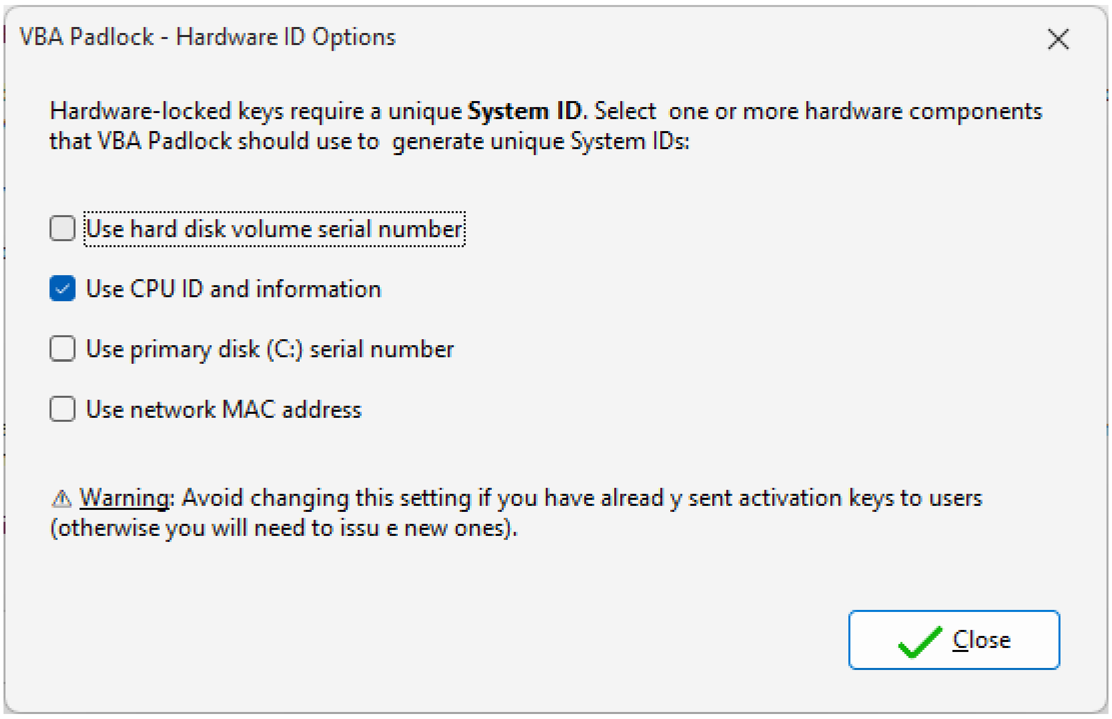
T52G4-900F2-E3QG1-SQPC7-35TKW-C5WCB-T9NCS-HK401-MX5[redacted]

19.4 Step 3: Add Hardware Locking (Optional)

To prevent users from sharing a single key across multiple computers, you can tie each license to the physical hardware.

1. In **Licensing Features**, go to Hardware ID Options.

2. Select the hardware components to monitor (CPU, Disk Serial, MAC Address).



VBA Padlock - Hardware ID Options

Hardware-locked keys require a unique **System ID**. Select one or more hardware components that VBA Padlock should use to generate unique System IDs:

- [ ] Use hard disk volume serial number
- [x] Use CPU ID and information
- [ ] Use primary disk (C:) serial number
- [ ] Use network MAC address

⚠ Warning: Avoid changing this setting if you have alread y sent activation keys to users (otherwise you will need to issu e new ones).

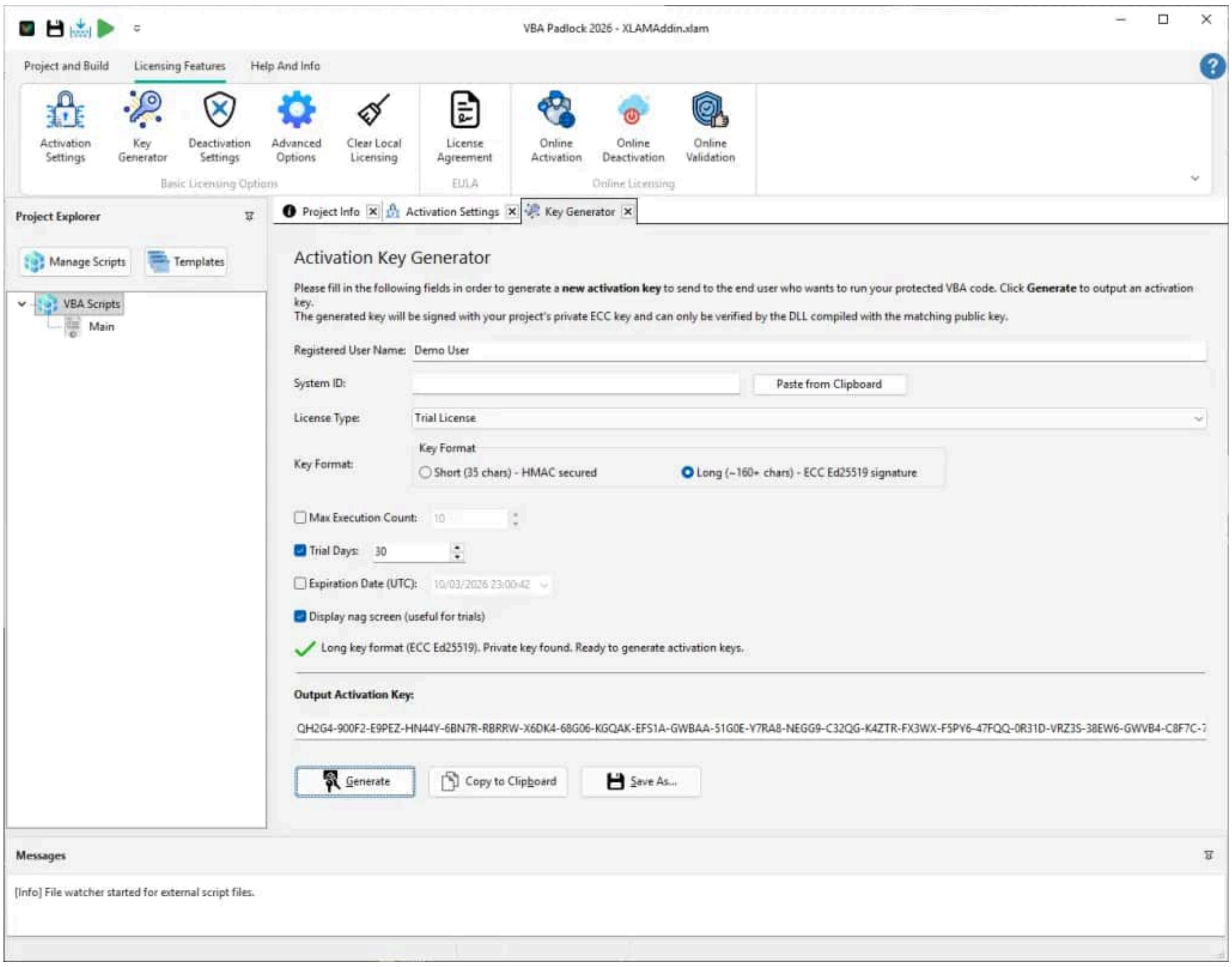
3. When enabled, the user’s activation dialog will display a unique **System ID**. They must send you this ID so you can generate a key that only works on *their* machine.

19.5 Step 4: Generate Your First License Key

Once your settings are configured and you’ve published your DLL, you can generate keys for your customers.

1. Go to **Licensing Features** → Key Generator.
2. Enter the **Registered Name** (e.g., “John Doe”).
3. Pace the user’s **System ID** if hardware locking is enabled.

4 Click **Generate Key**.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format: Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

```
QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7
```

```
[Info] File watcher started for external script files.
```

Note

VBA Padlock supports both **Short Keys** (hmac-based, easy to type) and **Long Keys** (Ed25519 signatures, extremely secure). Use Long Keys if you plan to use Online Activation.

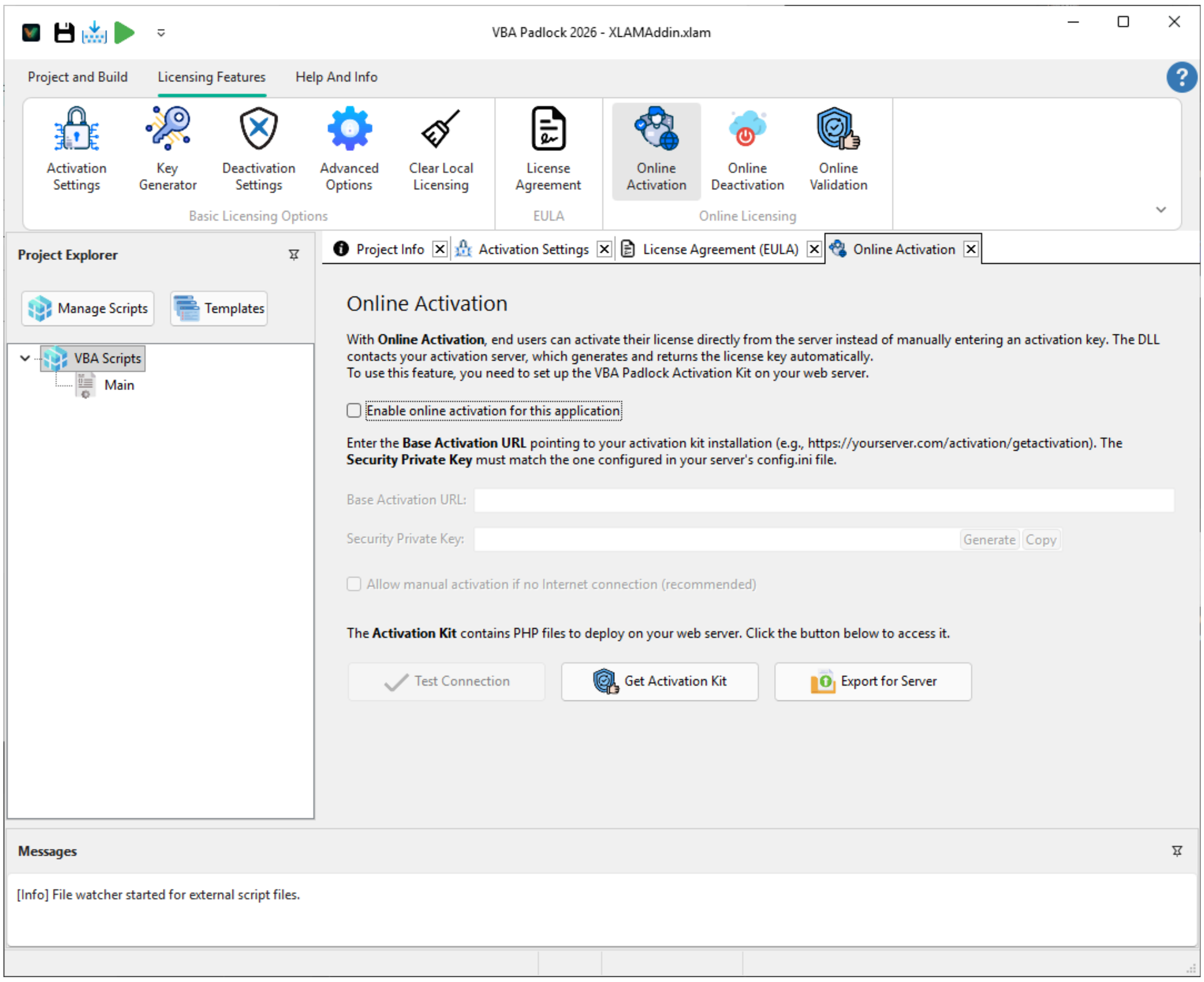
19.6 Step 5: Advanced — Online Activation

For a truly professional experience, you can automate key delivery over the internet.

1 Navigate to Online Activation Settings.

2 Enable **Online activation**.

3 Deploy the **VBA Padlock Activation Kit** (PHP/MySQL) to your web server.



Online Activation

With **Online Activation**, end users can activate their license directly from the server instead of manually entering an activation key. The DLL contacts your activation server, which generates and returns the license key automatically.
To use this feature, you need to set up the VBA Padlock Activation Kit on your web server.

☐ Enable online activation for this application

Enter the **Base Activation URL** pointing to your activation kit installation (e.g., https://yourserver.com/activation/getactivation). The **Security Private Key** must match the one configured in your server's config.ini file.

☐ Allow manual activation if no Internet connection (recommended)

The **Activation Kit** contains PHP files to deploy on your web server. Click the button below to access it.

[Info] File watcher started for external script files.

19.7 Final Checklist

Before sending your protected file to a client, perform these tests:

- ☐ **Trial Test:** Does the application open without a key? Is the remaining time correct?
- ☐ **Activation Test:** Does a generated key correctly unlock the application?
- ☐ **Locking Test:** Does a hardware-locked key fail on a different computer?
- ☐ **UI Test:** Does the activation dialog display your custom logo and links?

19.8 Next Steps

Key Generation

Learn about bulk generation and license management. Read the guide →

Activation Server

Deploy your own automated licensing server. Read the guide →

Deactivation

Allow users to transfer their license to a new PC. View feature →

Protect Excel

Ready to ship? Build your finalized Excel project. View guide →

20. Generating and Distributing Keys

Once you have published your protected application, the final step is to issue licenses to your customers. Whether you are fulfilling orders manually or automating the process through an e-commerce platform, VBA Padlock provides tools to generate secure, tamper-proof keys.

Time needed: 10–15 minutes

Tip

Before you start, ensure you have:

- A compiled VBA Padlock project with **Activation** enabled.
- Your project's **Master Key** (and ECC keys if using the Long format).

20.1 Key Format Comparison

VBA Padlock supports two formats. Choosing the right one depends on your distribution method and security needs.

Feature	Compact (Short)	Full (Long)
Typical Length	~35 chars (e.g., `A1B2-C3D4...`)	~120+ chars (Base64)
Technology	HMAC-SHA256	Ed25519 Elliptic Curve
Best for	Manual typing (phone, paper)	Digital delivery (Email, Clipboard)
Security	High	Very High (Digital Signature)
Max Runs Support	No	Yes

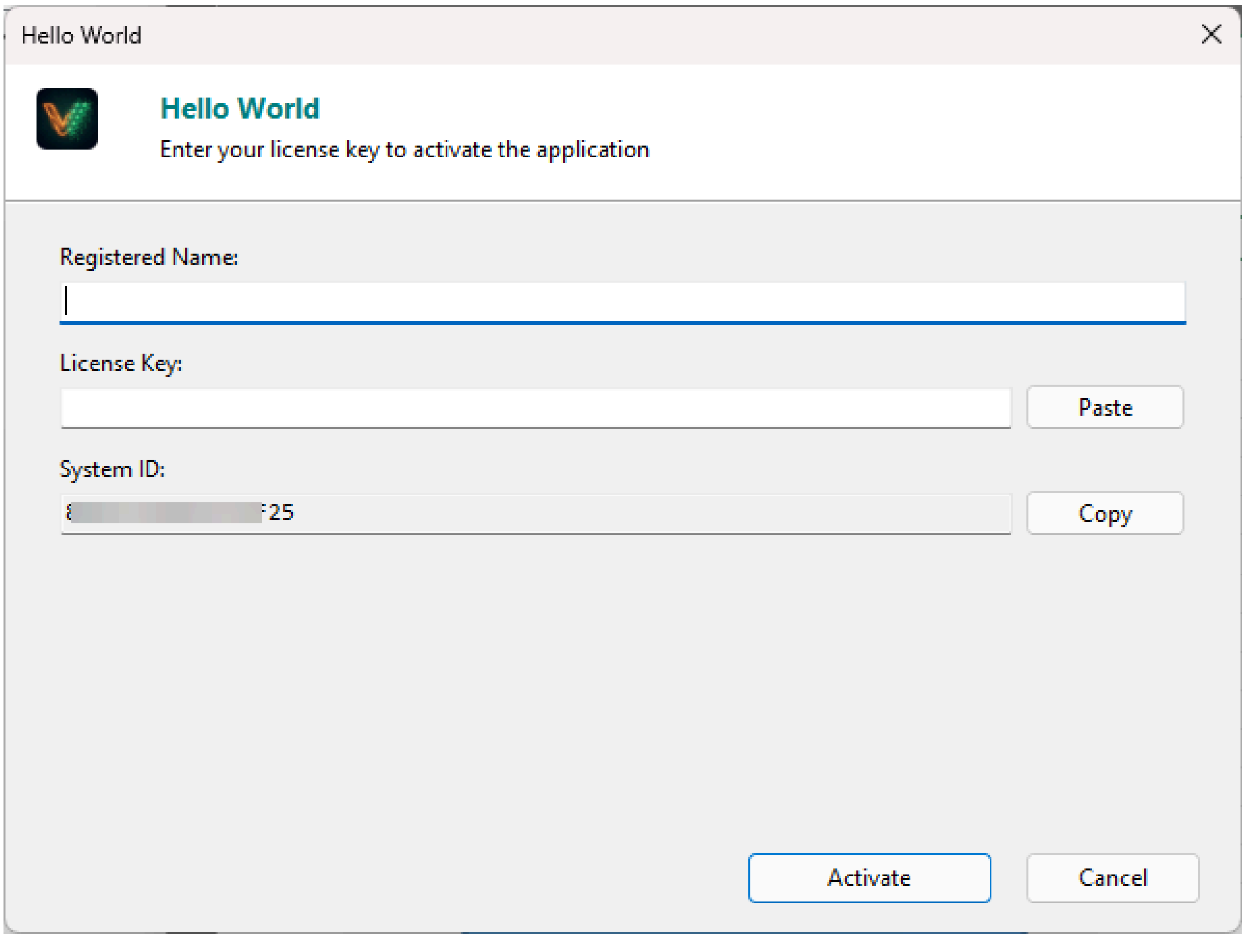
20.2 The Manual Workflow: Studio Key Generator

If you fulfill orders manually or are testing your licensing, the built-in generator in VBA Padlock Studio is the easiest path.

20.2.1 Step 1: Collect the System ID (Hardware Lock)

If you have enabled Hardware Locking, you must first obtain the unique "System ID" from your user's computer.

1. **Ask the user to open your application.**
2. The activation dialog will display their unique **System ID**.
3. The user copies this ID and sends it to you via email or your order form.


Hello World

Hello World

Enter your license key to activate the application

Registered Name:

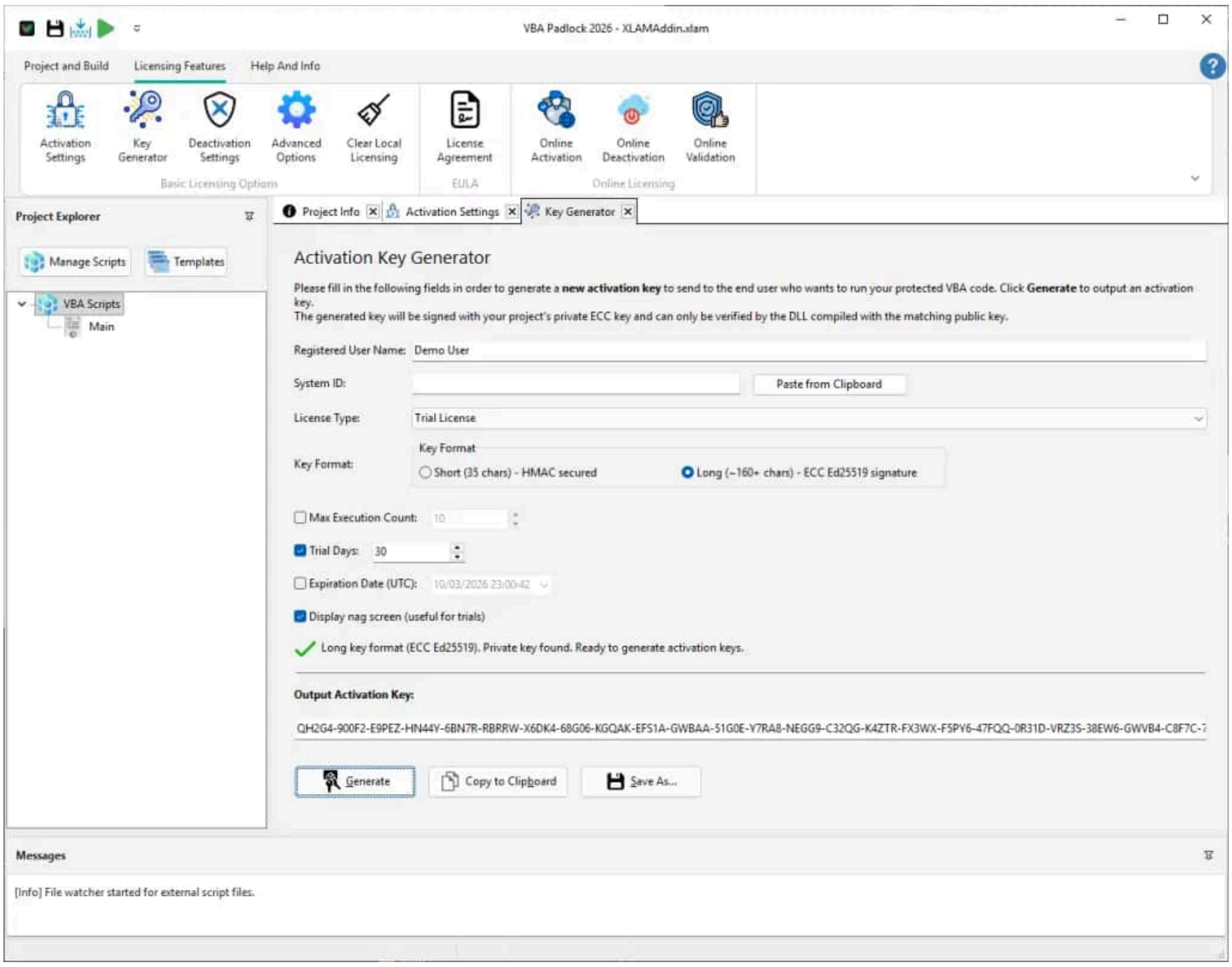
License Key:

System ID: [redacted]F25

20.2.2 Step 2: Generate the Key in VBA Padlock

1. Open your project and go to **Licensing Features** → Key Generator.
2. Enter the **Registered Name** (e.g., `Jane Doe`).
3. Paste the **System ID** provided by the user.
4. Choose the **License Type** and set any limits (Trial Days, Expiration Date).

5 Click **Generate**.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID: Paste from Clipboard

License Type: Trial License

Key Format: Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

Max Execution Count: 10

[✓] Trial Days: 30

Expiration Date (UTC): 10/03/2026 23:00:42

[✓] Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

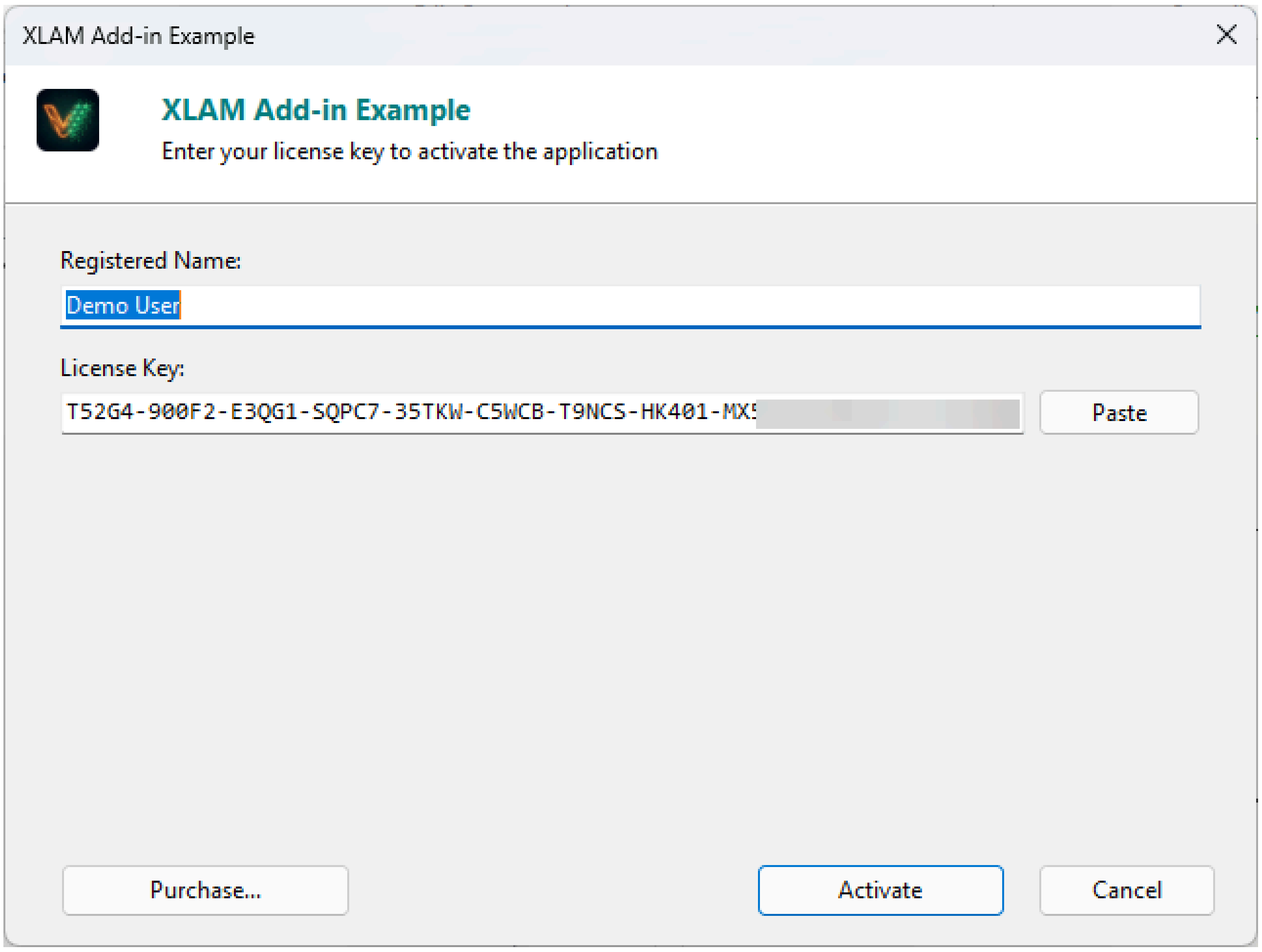
QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

Generate | Copy to Clipboard | Save As...

[Info] File watcher started for external script files.

20.2.3 Step 3: Deliver and Activate

Copy the generated key and send it to your user. They will enter it into the activation dialog to unlock the software.



XLAM Add-in Example

Enter your license key to activate the application

Registered Name: Demo User

License Key: T52G4-900F2-E3QG1-SQPC7-35TKW-C5WCB-T9NCS-HK401-MX5[redacted]

20.3 The Automated Workflow: PHP SDK

For e-commerce automation (WooCommerce, Shopify, Stripe), you can generate keys programmatically using our **PHP Key Generator SDK**.

Tip

The PHP SDK is a lightweight class included in the **VBA Padlock Activation Kit**. It allows you to generate keys on your server without having VBA Padlock Studio installed.

20.3.4 1. Basic Example (Compact Key)

```php
require_once('VBAPadlockKeygenAPI.php');

$vbp = new VBAPadlockKeygenAPI('vbp-YOUR-USER-ID');
$vbp->setApplicationMasterKey('{YOUR-PROJECT-GUID}');

// Optional: Binding to name and hardware
$vbp->setLicenseeName('Customer Name')
    ->setHardwareSystemID('ABCDE-12345')
    ->setMaxDays(365); // 1-year subscription

try {
    $licenseKey = $vbp->getActivationKey();
    echo "Your key: " . $licenseKey;
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
```

20.3.5 2. Ed25519 Signed Keys (Long Format)

To use the more secure Full format, you must provide your project's ECC keys. Export these from **Advanced Activation Options** in VBA Padlock Studio.

```php
$vbp->setFormat('full');
$vbp->setECCKeys($privateKeyHex, $publicKeyHex);
$vbp->setMaxRuns(50); // Limit to 50 executions
```

Technical Reference (SDK)

Method	Description
`setApplicationMasterKey($key)`	**Required.** Sets the Master Key from your project settings.
`setHardwareSystemID($id)`	Ties the key to a specific computer's Hardware ID.
`setMaxDays($days)`	Sets the number of days the license is valid from today.
`setExpireDate($date)`	Sets an absolute expiration date (`YYYY/MM/DD`).
`setMaxRuns($runs)`	Sets the maximum number of executions (Full format only).
`setLicenseeName($name)`	Binds the key to a specific user or company name.

Error	Cause & Resolution
`Invalid developer ID`	The User ID provided in the constructor is incorrect.
`Missing Master Key`	You forgot to call `setApplicationMasterKey()`.
`Daily request limit`	You've exceeded the generation limit for your account tier.
`CURL Error`	A network issue prevented connection to the keygen API.

20.4 Common Scenarios

Subscription Renewal

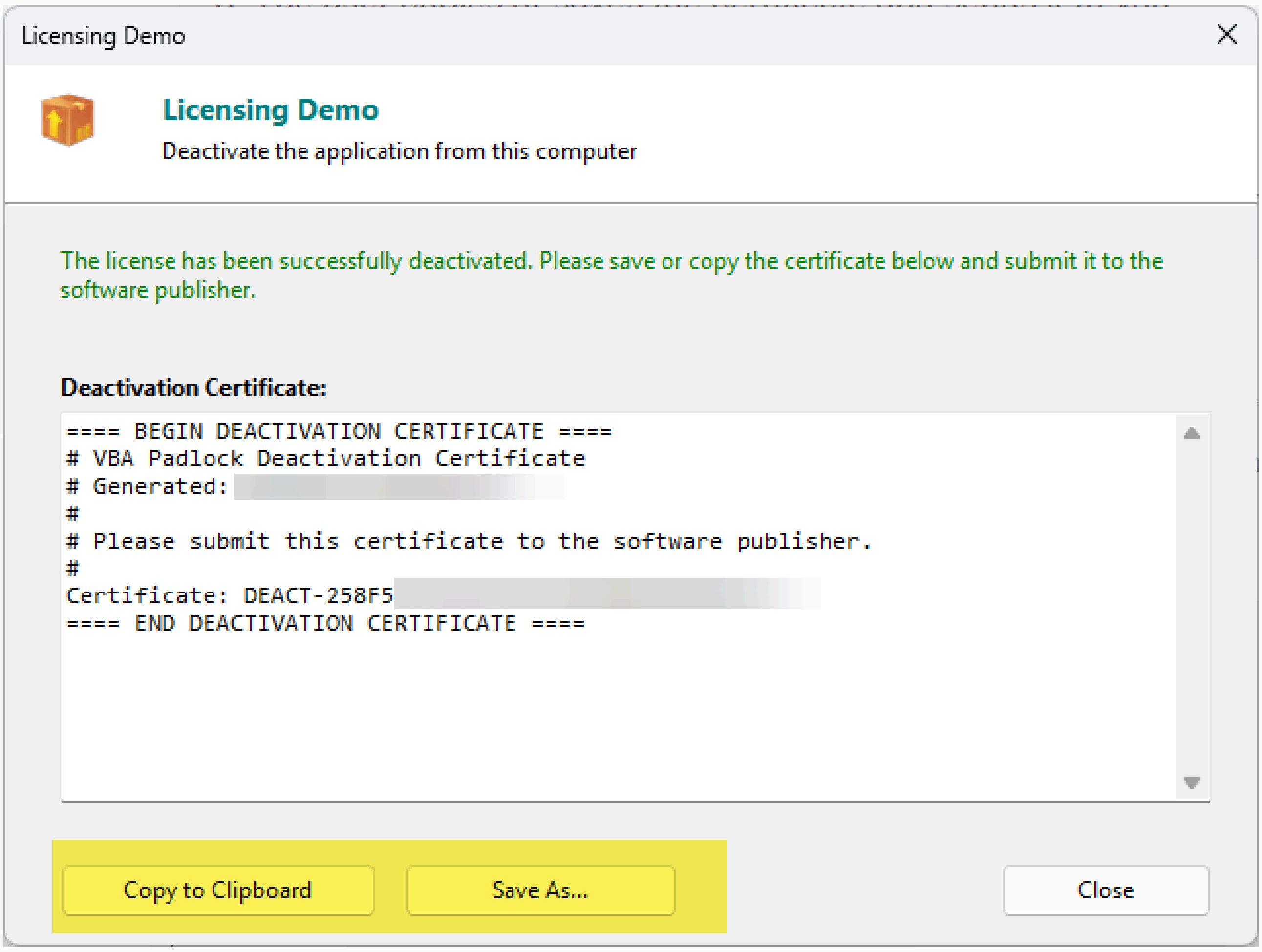
Generate a new key with a later **Expiration Date**. When the user enters the new key, it replaces the old one and extends their access.

Beta Testing

Issue a **Time-Limited** key with an absolute expiration date (e.g., `2026/12/31`). All beta keys will stop working at the same time.

License Transfer

If a user replaces their PC, collect their **Deactivation Certificate** as proof of removal before issuing a new key.



Licensing Demo

Licensing Demo
Deactivate the application from this computer

The license has been successfully deactivated. Please save or copy the certificate below and submit it to the software publisher.

Deactivation Certificate:

```
==== BEGIN DEACTIVATION CERTIFICATE ====
# VBA Padlock Deactivation Certificate
# Generated:
#
# Please submit this certificate to the software publisher.
#
Certificate: DEACT-258F5
==== END DEACTIVATION CERTIFICATE ====
```

Enterprise Bulk

Use the PHP SDK to generate dozens of keys in a loop for an enterprise client, mapping them to a list of provided System IDs.

20.5 Next Steps

Deactivation

Learn how to let users return or transfer licenses. Read the guide →

Online Activation

Automate the entire process with an activation server. Read the guide →

Localization

Translate the activation dialogs for international users. Read the guide →

Key Generator UI

Detailed reference for the Key Generator dialog. View Reference →

API Reference

Check the status of a license from your VBA code. View API →

21. Deploy the PHP Activation Server

You've protected your VBA application and now you want to automate the activation process. Instead of manually generating keys and emailing them, you can deploy the **VBA Padlock Activation Kit** — a complete PHP server that handles everything automatically: activation, deactivation, periodic validation, and even a PayPal-powered storefront.

What you'll set up:

- A PHP activation server on your web hosting
- Automated online activation, deactivation, and validation
- A web dashboard to manage clients, licenses, and activation logs
- Optional: a PayPal store for self-service license purchases

Time needed: 30–45 minutes

Tip

Before you start, make sure you have:

- A web server with PHP 8.1+, MySQL/MariaDB, and Apache with `mod_rewrite`
- Composer installed (to install PHP dependencies)
- VBA Padlock 2026 with a compiled project
- FTP or SSH access to your web server
- HTTPS enabled on your domain (mandatory for security)

21.1 What's in the Kit

The Activation Kit is a full-featured server application:

Component	Technology	Purpose
Activation API	PHP 8.1+ / Fat-Free Framework	Handles activation, validation, deactivation requests from VBA Padlock DLLs
Web Dashboard	React 18 / TypeScript / Tailwind CSS	Manage clients, licenses, logs, and settings via a modern SPA
PayPal Store	PayPal REST API v2	Optional storefront for selling licenses directly
Key Generator	Bundled PHP SDK	Generates compact (HMAC) or full (ECC Ed25519) license keys

21.1.1 Required PHP Extensions

Extension	Purpose
`pdo_mysql`	Database connectivity
`sodium`	Ed25519 key generation
`curl`	PayPal API and key generation
`openssl`	JWT token signing
`json`	JSON encoding/decoding
`mbstring`	Multi-byte string handling

21.2 Step 1: Get and Upload the Kit

1. Open your VBA Padlock project.
2. Go to the **Licensing Features** tab → **Online Activation**.
3. Check **Enable online activation**.
4. Click **Get Activation Kit** to download the server files.
5. Extract the archive. If you don't plan to customize the dashboard, **delete the `dashboard-src/` folder** before uploading — it contains React source code not needed in production.
6. Upload the `vbapadlock-activkit/` directory to your web server:

```
https://yourdomain.com/vbapadlock-activkit/
```

7. Install PHP dependencies via SSH:

```
cd /path/to/vbapadlock-activkit/inc
composer install
```

21.3 Step 2: Configure VBA Padlock Keys

Open `inc/config.ini` on your server and set your VBA Padlock credentials:

```
[globals]
vbapadlockmasterkey = "YOUR_MASTER_KEY_HERE"
vbapadlockpkey = "{YOUR-PRIVATE-KEY-GUID-HERE}"
usehardwarelocking = 1
```

⚠ Caution

These three values **must match exactly** between `config.ini` and your VBA Padlock Studio project. If any of them differ, activations will fail with “Security validation failed”.

21.3.2 Configuration Matching Reference

Setting	Where to find in VBA Padlock Studio	config.ini Key
Master Key	Licensing Features → Advanced Activation → Application Master Key	`vbapadlockmasterkey`
Security Private Key	Licensing Features → Online Activation → Security Private Key	`vbapadlockpkey` (including braces `{...}`)
Hardware Locking	Licensing Features → Activation Settings → Hardware Locking checkbox	`usehardwarelocking` (`1` = enabled, `0` = disabled)

21.3.3 ECC Keys for Full-Format Licenses (Optional)

To generate cryptographically-signed full-format keys (Ed25519) instead of compact HMAC keys, add your ECC key pair to `config.ini`:

```
ecc_private_key = "a1b2c3d4e5f6...your_64_hex_char_private_key..."
ecc_public_key = "f6e5d4c3b2a1...your_64_hex_char_public_key..."
```

```
ecc_private_key_file = "/path/to/project.vbapadlock/ecc_private.key"
ecc_public_key_file = "/path/to/project.vbapadlock/ecc_public.pub"
```

Export ECC keys from VBA Padlock Studio: **Advanced Activation Options** → **Copy ECC Keys For PHP Key Generator**.

> **Note**
>
> The key format is **auto-selected**: if valid ECC keys are configured, the server generates long (ECC) keys. Otherwise, it generates compact (HMAC) keys. No code changes needed.

21.4 Step 3: Run the Setup Wizard

The first time you access the dashboard, a setup wizard guides you through three steps.

1. Open your browser and go to:

```
https://yourdomain.com/vbapadlock-activkit/dashboard
```

2. **Database Configuration** — Enter your MySQL/MariaDB credentials. The wizard tests the connection, creates the required tables, and writes the credentials to `config.ini`.

3. **Admin Account** — Create your dashboard login (name, email, password). A JWT secret is auto-generated.

4. **Application Settings** — Set your application name and the default number of activations per license (e.g., 3).

⚠ **Caution**

The setup wizard **locks itself permanently** after completion. All setup endpoints return 403 Forbidden. To re-run setup, you must manually set `setup_complete = 0` in the `configurations` database table.

21.5 Step 4: Configure VBA Padlock Studio

1. In the **Online Activation** dialog, set the **Base Activation URL**:

```
https://yourdomain.com/vbapadlock-activkit/getactivation
```

2. Click **Test Connection** to verify. The server responds with a health-check JSON.

3. Click **Generate** next to the **Security Private Key** if you haven't already. Copy the GUID and make sure it matches `vbapadlockpkey` in `config.ini`.

21.6 Step 5: Enable Deactivation and Validation (Optional)

21.6.4 Online Deactivation

In the Online Deactivation dialog:

1. Check **Enable online deactivation**.
2. Set the **Base Deactivation URL** (e.g., `https://yourdomain.com/vbapadlock-activkit/dodeactivation`).

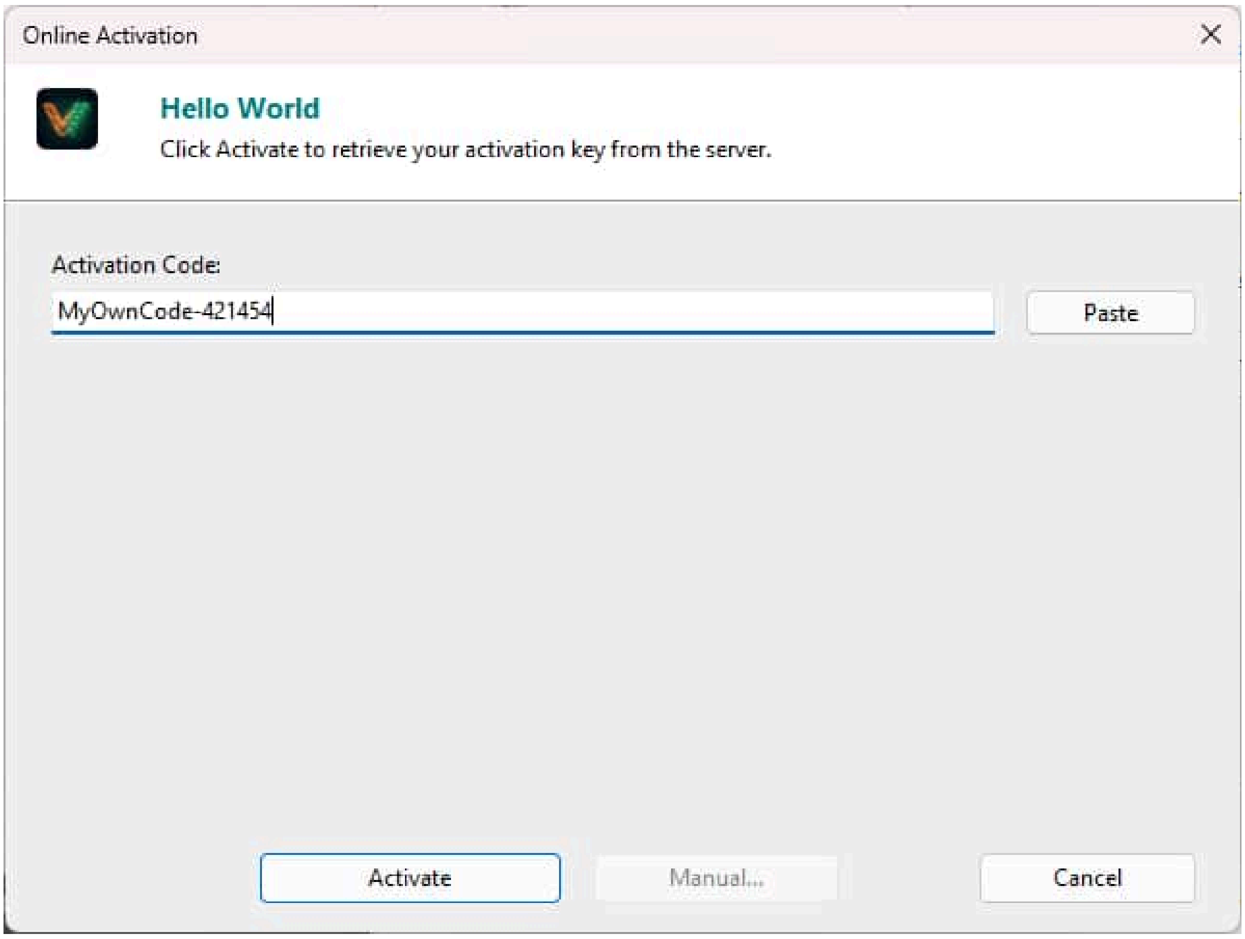
21.6.5 Online Validation

In the Online Validation dialog:

1. Check **Enable online validation**.
2. Set the **Base Validation URL** (e.g., `https://yourdomain.com/vbapadlock-activkit/dovalidation`).
3. Choose how often to validate (e.g., every launch, daily).

21.7 Step 6: Test the Full Flow

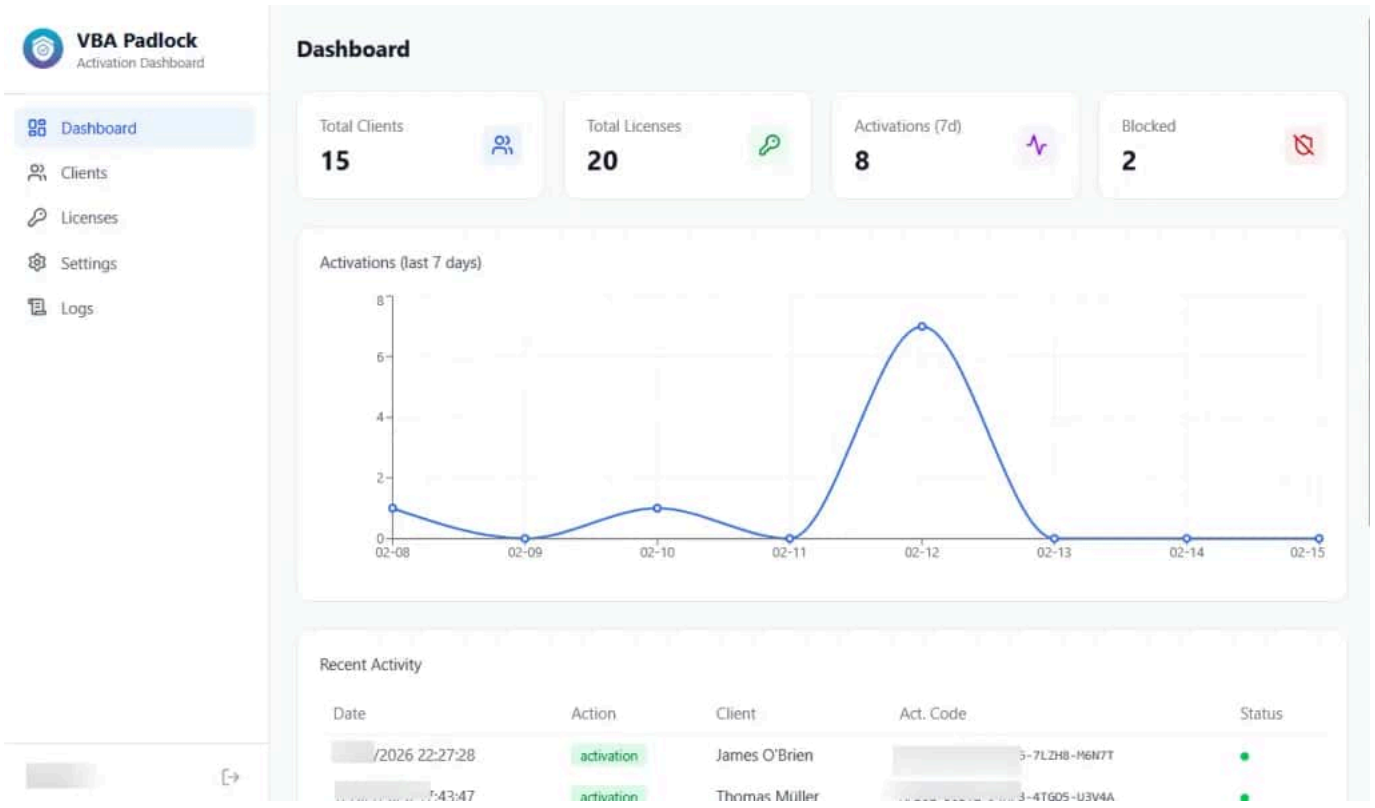
1. Compile and publish your project.
2. Open the protected Office file on a test machine.
3. The activation dialog appears with an online activation option.



4. Enter test user details and click Activate.
5. The server processes the request and returns a license key.
6. Verify the activation succeeded — check the **Logs** page in the dashboard.

21.8 Dashboard Overview

The web dashboard gives you full control over your activation system.



Recent Activity

Date	Action	Client	Act. Code	Status
[redacted]/2026 22:27:28	activation	James O'Brien	[redacted]5-7LZH8-M6N7T	●
[redacted]:43:47	activation	Thomas Müller	[redacted]3-4TGD5-U3V4A	●

21.8.6 Statistics

The home page displays four key metrics (total clients, total licenses, blocked licenses, activations this week) and a line chart of daily activations over 7, 30, or 90 days.

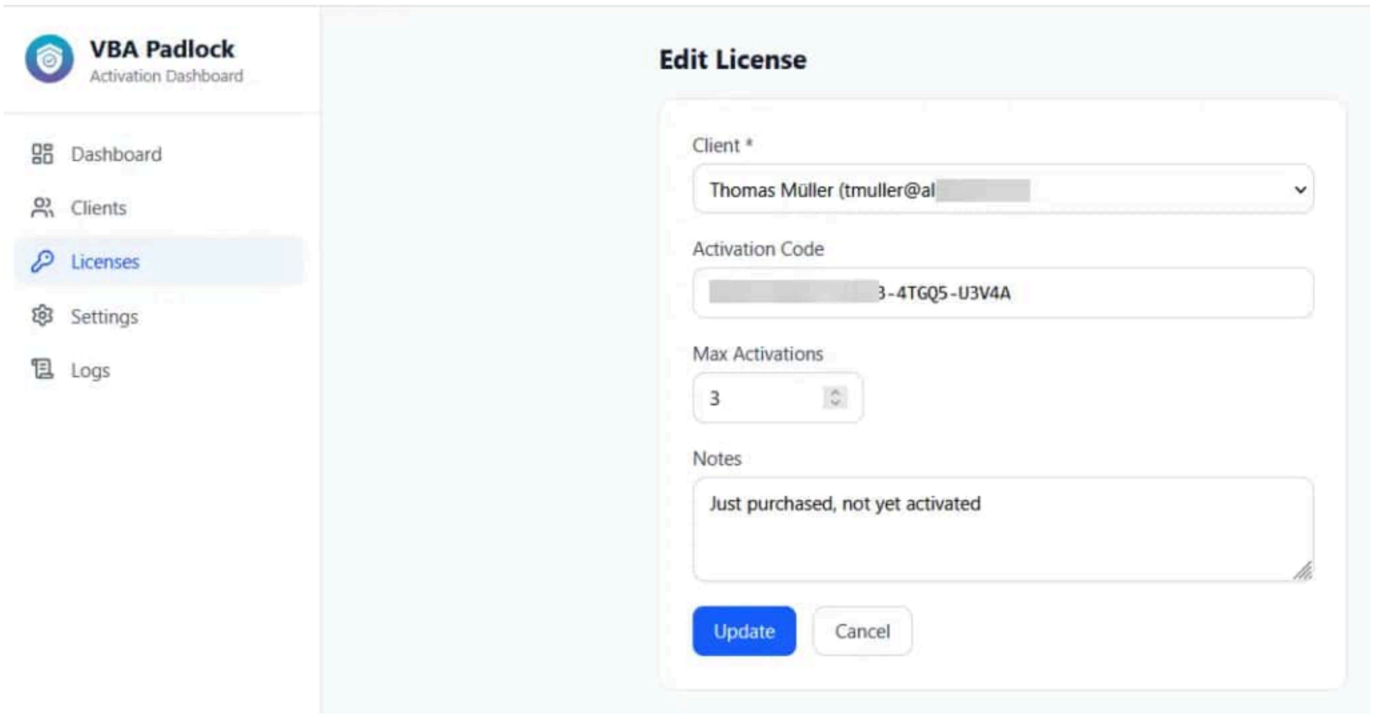
21.8.7 Client Management

Create, edit, search, and delete customers. Each client can have multiple licenses.

21.8.8 License Management

Create licenses with custom or auto-generated activation codes, set max activations, and use quick actions:

- **Block / Unblock** — Prevent or restore activation for a license
- **Reset Activations** — Restore activation count to maximum (for re-activation on new hardware)
- **Edit** — View activation details (last system ID, last IP, generated key)



Edit License

Client *
Thomas Müller (tmuller@al[redacted]

Activation Code
[redacted]3-4TGQ5-U3V4A

Max Activations
3

Notes
Just purchased, not yet activated

21.8.9 Activation Logs

A complete audit trail of every activation, validation, and deactivation event, with filtering by action type, status, system ID, and IP address.

21.9 Server Endpoints

The Activation Kit provides three public endpoints used by VBA Padlock DLLs:

Endpoint	Method	Purpose
`/getactivation`	POST	Process activation requests — returns a license key and token
`/dovalidation`	POST	Validate an active license — checks for revocation via challenge-response
`/dodeactivation`	POST	Deactivate a license — frees up an activation slot
`/` or `/ping`	GET	Health check — returns server status and version

21.9.10 Activation Response Format

```json
{
    "header": "VBA Padlock Activation",
    "status": 1,
    "key": "XXXXX-XXXXX-XXXXX-XXXXX-XXXXX",
    "token": "sha256_token_for_validation",
    "username": "John Doe"
}
```

Field	Required	Description
`header`	Yes	Must contain `"VBA Padlock"`
`status`	Yes	`1` = success, `2` = error
`key`	Yes (on success)	The generated license key
`token`	Yes (on success)	Unique token for future validation and deactivation
`username`	No	Registered user name. When provided, it is stored locally and returned by `VBAPL_GetRegisteredName()`.

21.9.11 Error Response Format

```json
{
    "header": "VBA Padlock Activation",
    "status": 2,
    "error": "Maximum number of activations reached"
}
```

Tip

The `username` field is optional but recommended. When provided, it allows your application to greet the user by name (e.g., "Welcome, John Doe") after online activation. Without it, `VBAPL_GetRegisteredName()` returns an empty string.

21.10 Activation Lifecycle

Understanding the full lifecycle helps you support your customers effectively.

21.10.12 Activation

1. The user enters their activation code in your protected application.
2. The DLL sends a `POST /getactivation` request with the code, hardware ID, and a SHA256 security checksum.
3. The server verifies the checksum, looks up the license, checks it's not blocked, and verifies remaining activations.
4. The server generates an activation key and returns it with a unique token.
5. Each activation from a **different machine** (different system ID) consumes one activation slot.

21.10.13 Reactivation (Same Machine)

If a user re-activates on the **same hardware** (same system ID), the server recognizes it and does **not** consume an additional activation slot. This handles cases like Windows reinstalls or application updates gracefully.

21.10.14 Validation

The DLL periodically sends `POST /dovalidation` requests to verify the license is still valid. The server checks the license exists and is not blocked, then returns a challenge-response to authenticate the exchange.

21.10.15 Deactivation

When a user wants to transfer their license to a new machine:

1. They click "Deactivate" in the application (or you provide a button).
2. The DLL sends a `POST /dodeactivation` request.
3. The server increments the remaining activation count.
4. The user can now activate on a different machine.

21.10.16 Backward Compatibility

If an activation code is **not found in the database**, the server still generates an activation key. This ensures the API works even before you create licenses in the dashboard. In this mode, there is no activation limit enforcement. To enable full tracking, create a license in the dashboard matching the activation code you distribute to customers.

21.11 PayPal Store (Optional)

The kit includes a public store page where customers can purchase licenses via PayPal.

21.11.17 Setup

1. In the dashboard, go to **Settings** → **PayPal**.
2. Set **Mode** to `sandbox` (for testing) or `live` (for production).
3. Enter your **PayPal Client ID** and **Client Secret** (from the PayPal Developer Dashboard).
4. Set the **Price** and **Currency** (e.g., `49.99` / `USD`).
5. Toggle **Store Enabled** to on.
6. The store is available at:

```
https://yourdomain.com/vbapadlock-activkit/store
```

21.11.18 Purchase Flow

1. Customer visits the store page and clicks the PayPal button.
2. After payment, a client record is created (or found by email), a license is generated, and the activation code is displayed.
3. Buyer and seller email notifications are sent (if configured in Settings → Email).

Tip

Always test the full purchase flow with PayPal **sandbox** accounts before switching to live mode.

21.12 Security Considerations

21.12.19 HTTPS is Mandatory

The activation process transmits license keys, hardware IDs, and security checksums. Always serve the kit over HTTPS.

21.12.20 File Protection

The `.htaccess` file blocks access to sensitive paths:

- `/inc/` — PHP source code, `config.ini`, vendor directory
- `/dashboard-src/` — React source code
- `*.ini`, `*.log`, `*.lock`, `*.md` files

Verify after deployment: try to access `https://yourdomain.com/vbapadlock-activkit/inc/config.ini` in your browser — it should return **403 Forbidden**.

21.12.21 Request Authentication

All activation requests are verified with **SHA256 checksums** using salts unique to VBA Padlock. Validation and deactivation use a **challenge-response mechanism** that prevents replay attacks.

21.12.22 Backups

Your license database is a **critical business asset**. Losing it means losing all customer activation records. Set up automated daily backups:

```
# Example cron job: daily backup at 2:00 AM, keep 30 days
0 2 * * * mysqldump -u dbuser -p'password' vbapadlock_db | gzip >
/backups/vbapadlock_$(date +\%Y\%m\%d).sql.gz
```

Also back up `inc/config.ini` — it contains your JWT secret, ECC keys, and database credentials.

21.13 Deployment Checklist

Before going live, verify:

- ☐ `vbapadlockmasterkey` matches VBA Padlock Studio project
- ☐ `vbapadlockpkey` matches VBA Padlock Studio (including braces)
- ☐ `usehardwarelocking` set correctly (0 or 1)
- ☐ ECC keys configured (if using long key format)
- ☐ `DEBUG = 0` in config.ini
- ☐ HTTPS enabled
- ☐ `.htaccess` active (test: `/inc/config.ini` returns 403)

- ☐ Setup endpoints locked (test: `POST /api/setup/database` returns 403)
- ☐ `config.ini` set to read-only
- ☐ Database backups scheduled
- ☐ Test Connection from VBA Padlock Studio succeeds
- ☐ Full activation test completed and visible in dashboard logs

21.14 Troubleshooting

21.14.23 Activation Errors

Error	Cause	Solution
"Security validation failed"	`vbapadlockpkey` mismatch between `config.ini` and VBA Padlock Studio	Copy the exact GUID from the Studio, including braces `{...}`
"Maximum number of activations reached"	License has 0 remaining activations	In the dashboard, click **Reset Activations** on the license
"License is blocked"	License was manually blocked	In the dashboard, click **Unblock** on the license
Activation succeeds but not tracked	Activation code not in database	Create a license in the dashboard with the same key

21.14.24 Server Errors

Problem	Cause	Solution
HTTP 404 on endpoints	`mod_rewrite` not enabled	Enable `mod_rewrite` in Apache and ensure `AllowOverride All` is set
HTTP 500	PHP dependencies not installed	Run `cd inc && composer install`
Empty response	PHP version too old	Requires PHP 8.1+ — check with `php -v`
Dashboard blank page	Build output missing	Ensure the `dashboard/` directory contains the pre-built SPA files
CORS errors in browser	Mixed HTTP/HTTPS	Ensure the dashboard URL and API URL both use HTTPS

21.14.25 Debug Mode

To enable detailed error logging, set `DEBUG = 1` in `config.ini`. Check your PHP error log for:

- Security checksum calculations (computed vs. received)
- ECC key operations

- Database query errors

Caution

Always set `DEBUG = 0` in production. Debug mode logs sensitive information.

21.15 What's Next?

Key Generation

Generate offline keys for users without Internet.

Read the guide →

Online Activation

Detailed configuration reference.

View settings →

Batch Compilation

Automate builds with CI/CD.

Read the guide →

Troubleshooting

Common issues and solutions.

View reference →

21.16 See Also

- Online Activation Feature — Architecture overview and feature description
- ☐ Online Activation Settings — Studio configuration reference
- Online Deactivation Settings — Deactivation configuration
- Online Validation Settings — Validation configuration
- Key Generation Guide — Manual and automated key generation (includes PHP SDK reference)
- License Return Codes — Server response codes

22. Batch Compilation and CI/CD Integration

You manage multiple VBA Padlock projects and want to automate the build process. This guide shows you how to compile projects from the command line, integrate with batch scripts, and set up CI/CD pipelines.

What you'll learn:

- How to build VBA Padlock projects from the command line
- How to create batch scripts for multiple projects
- How to integrate with CI/CD systems
- How to verify build artifacts

Time needed: 15 minutes

Tip

Before you start, make sure you have:

- VBA Padlock 2026 installed
- One or more VBA Padlock projects configured and tested in VBA Padlock Studio
- Familiarity with Windows batch scripting

22.1 Command-Line Compilation

22.1.1 Synopsis

```
VBAPadlock.exe "<office-file>" [switches]
```

The first argument is the path to the Office source file (`.xlsm`, `.docm`, `.accdb`, `.pptm`, etc.). The associated `.vbapadlock` project folder must be in the same directory.

22.1.2 Switches

Switch	Type	Description
`-c` or `--compile`	Flag	Compile the project and build all DLLs (32-bit, 64-bit, and satellite).
`-q` or `--quit`	Flag	Quit the application after compilation.
`-s` or `--silent`	Flag	Hide the main window (headless mode). Implies `--quit` when combined with `--compile`.

Switch	Type	Description
`--publish`	Flag	Perform a final/publish build. Disables HotLoading for production distribution.
`--log:<path>`	String	Save the compilation log to the specified file path.

Caution

String switch values use `:` or `=` as delimiter — **not** a space. Write `--log:build.log`, not `--log build.log`.

22.1.3 Exit Codes

Exit Code	Constant	Meaning
`0`	`EXIT_SUCCESS`	Compilation completed successfully. All DLLs have been built.
`1`	`EXIT_INVALID_ARGUMENTS`	Invalid command-line arguments or missing source file path.
`2`	`EXIT_FILE_NOT_FOUND`	The specified Office source file was not found.
`3`	`EXIT_PROJECT_LOAD_ERROR`	The project could not be loaded (missing `.vbapadlock` folder, corrupt `project.xml`, etc.).
`4`	`EXIT_COMPILE_ERROR`	VBA compilation failed (syntax error) or DLL build failed.

Tip

In batch scripts and CI/CD pipelines, always check `%ERRORLEVEL%` after calling VBA Padlock. A non-zero exit code means the build should be considered failed.

22.1.4 Opening a Project (GUI Mode)

Without any switches, passing just a file path opens the project in VBA Padlock Studio — no compilation is triggered:

```
VBAPadlock.exe "C:\Projects\MyApp\MyWorkbook.xlsm"
```

22.2 Step 1: Single Project Build

Create a batch script that builds one project and checks the result:

```
@echo off
setlocal

set VBAPADLOCK="C:\Program Files\VBA Padlock\VBAPadlock.exe"
set PROJECT="C:\Projects\MyApp\MyWorkbook.xlsm"

echo Building %PROJECT%...
%VBAPADLOCK% %PROJECT% --compile --silent --log:build.log

if %ERRORLEVEL% NEQ 0 (
    echo BUILD FAILED with exit code %ERRORLEVEL%
    type build.log
    exit /b %ERRORLEVEL%
)

echo Build successful.
endlocal
```

22.3 Step 2: Multi-Project Build

Build several projects in sequence:

```
@echo off
setlocal enabledelayedexpansion

set VBAPADLOCK="C:\Program Files\VBA Padlock\VBAPadlock.exe"
set PROJECTS_DIR=C:\Projects

set FAILED=0

for %%P in (
    "%PROJECTS_DIR%\AppA\Workbook.xlsm"
    "%PROJECTS_DIR%\AppB\Report.docm"
    "%PROJECTS_DIR%\AppC\Database.accdb"
) do (
    echo.
    echo === Building %%~nxP ===
    %VBAPADLOCK% %%P --compile --silent --log:%%~nP_build.log
    if !ERRORLEVEL! NEQ 0 (
        echo FAILED: %%~nxP
        set /a FAILED+=1
    ) else (
        echo OK: %%~nxP
    )
)

echo.
if %FAILED% GTR 0 (
    echo %FAILED% project(s) failed to build.
    exit /b 1
) else (
    echo All projects built successfully.
)

endlocal
```

22.4 Step 3: Verify Build Artifacts

After building, verify the output DLLs exist:

```
@echo off
set PROJECT_DIR=C:\Projects\MyApp

:: Check all three DLLs exist
if not exist "%PROJECT_DIR%\bin\MyWorkbookrun32.dll" (
    echo ERROR: Missing 32-bit runtime DLL
    exit /b 1
)
if not exist "%PROJECT_DIR%\bin\MyWorkbookrun64.dll" (
    echo ERROR: Missing 64-bit runtime DLL
    exit /b 1
)
if not exist "%PROJECT_DIR%\bin\MyWorkbook.dll" (
    echo ERROR: Missing satellite DLL
    exit /b 1
)

echo All build artifacts verified.
```

22.5 Step 4: CI/CD Integration

22.5.5 Requirements

- VBA Padlock must be installed on the build machine.
- The Office file and its `.vbapadlock` project folder must be in the repository.
- The `bin\` output directory should be in `.gitignore`.

22.5.6 Example: GitHub Actions (Self-Hosted Runner on Windows)

```
name: Build VBA Padlock Projects
on: [push]

jobs:
  build:
    runs-on: self-hosted
    steps:
      - uses: actions/checkout@v4

      - name: Build Project
        shell: cmd
        run: |
          "C:\Program Files\VBA Padlock\VBAPadlock.exe"
"%GITHUB_WORKSPACE%\MyWorkbook.xlsm" --compile --silent --publish --
log:build.log
          if %ERRORLEVEL% NEQ 0 (
            type build.log
            exit /b %ERRORLEVEL%
          )

      - name: Upload artifacts
        uses: actions/upload-artifact@v4
        with:
          name: protected-app
          path: |
            MyWorkbook.xlsm
            bin/

      - name: Upload build log
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: build-log
          path: build.log
```

Note

VBA Padlock requires a Windows environment. Cloud-based Linux runners cannot be used. Use a self-hosted Windows runner with VBA Padlock installed.

22.6 Project Structure for Version Control

```
MyProject/
├── MyWorkbook.xlsm              ← Office file
├── MyWorkbook.vbapadlock/       ← Project folder (commit this)
│   ├── project.xml
│   ├── Main.bas
│   └── Helpers.bas
├── build.bat                    ← Build script
├── .gitignore                   ← Excludes bin/
└── bin/                         ← Output (generated, do not commit)
```

22.7 Log File Format

When `--log:<path>` is specified, the log file is a plain UTF-8 text file containing all compilation messages. Each line is prefixed with its severity level:

```
VBA Padlock Compilation Log - 2026-02-12 14:30:00
------------------------------------------------------------
[Info] Validating project settings...
[Info] Compiling scripts...
[Info] Built 32-bit runtime DLL: C:\Projects\bin\MyWorkbookrun32.dll
[Info] Built 64-bit runtime DLL: C:\Projects\bin\MyWorkbookrun64.dll
[Info] Built satellite DLL: C:\Projects\bin\MyWorkbook.dll
[Success] Compilation completed successfully.
```

The log file is created in all exit paths — including error scenarios — so it can always be inspected after a build.

22.8 Environment Notes

- The Office document and its `.vbapadlock` project folder must be in the same directory.
- VBA Padlock must have access to its `dll_templates\` subdirectory (part of the installation).

- All VBA script modules (`.bas` files) must be saved as **UTF-8**. The compiler reads scripts using UTF-8 encoding. Using a different encoding may cause compilation errors or incorrect string handling.

22.9 What's Next?

Project Format

Understand the .vbapadlock folder structure.

View reference →

Protect Excel

Step-by-step project creation.

Read the guide →

Troubleshooting

Common build issues and solutions.

Read more →

22.10 See Also

- Project Format Reference — Project directory structure
- Troubleshooting — Common issues and solutions

23. UI Reference

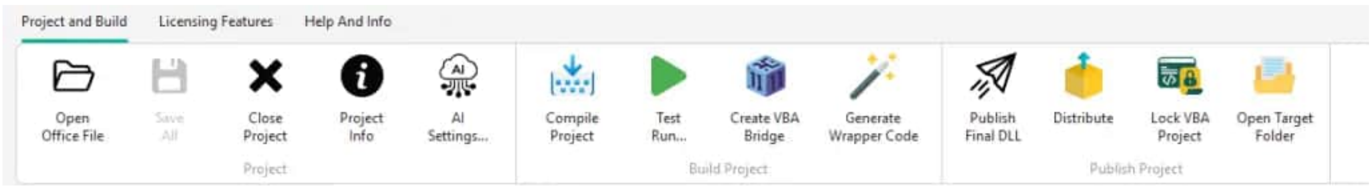
Welcome to the comprehensive UI Reference for **VBA Padlock Studio**. This section provides a detailed breakdown of every button, checkbox, and configuration field within the application.

Tip

Context-Sensitive Help: You can access these pages directly from within the software. Click the **Help** button in any dialog to open the corresponding documentation page.

23.1 🏗 Project and Build

The **Project and Build** tab is where you manage your source files, write protected scripts, and compile your final DLLs.



Workspace Essentials

From the **Welcome Screen** to the **Main Window**, learn how to navigate your projects.

- Welcome Screen
- Main Window
- Project Information

Coding & Testing

Write and debug your protected logic in a secure environment.

- Code Editor
- Script Templates
- Test Runner

The VBA Bridge

Connect your DLL to Microsoft Office with ease.

- VBA Bridge Injection
- Wrapper Generator
- Lock VBA Project

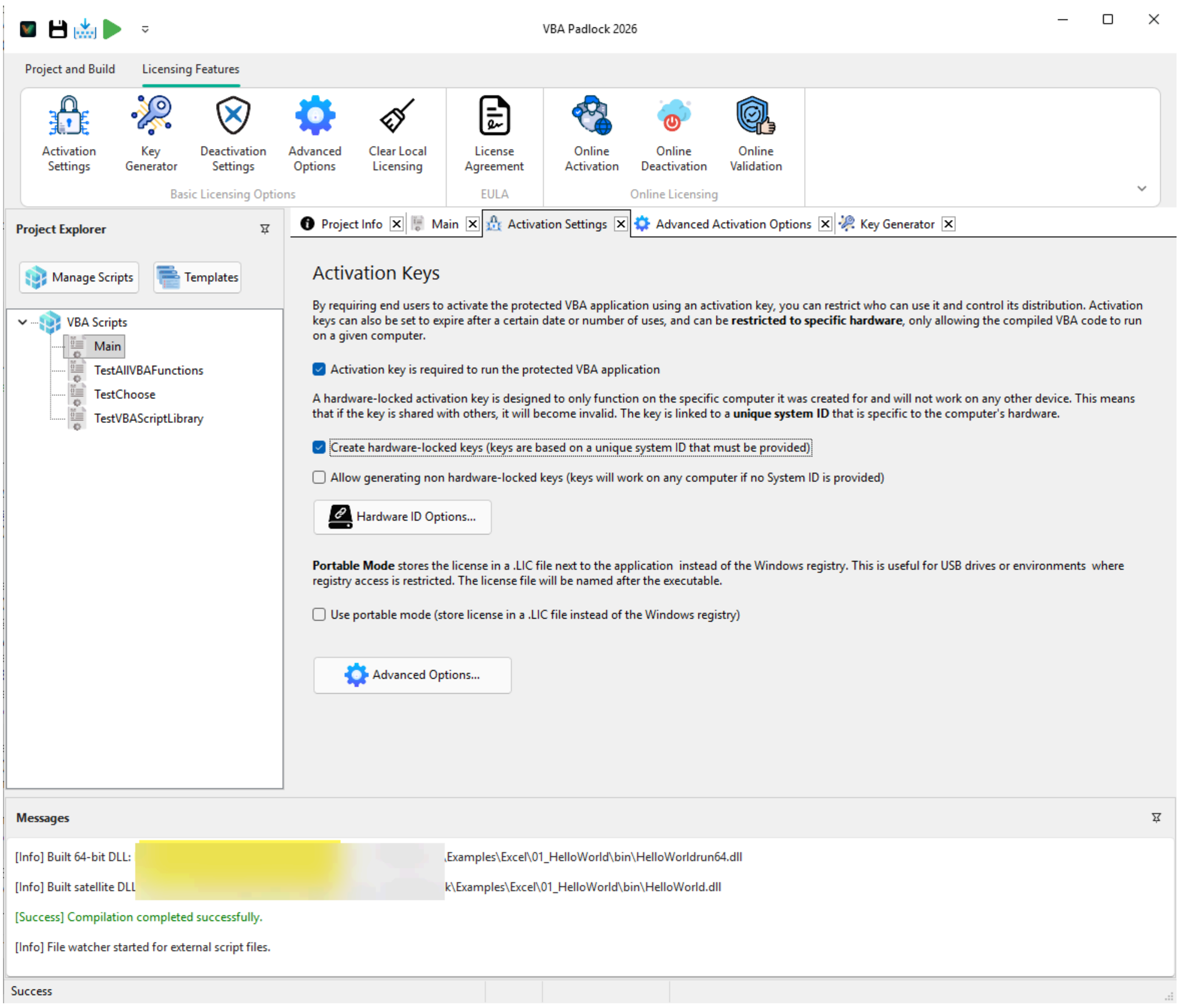
Finalization

Prepare your application for the real world.

- Compile Project
- Publish Final DLL
- Distribution & ZIP
- Create Installer

23.2 Licensing Features

The **Licensing Features** tab allows you to transform your protected code into a commercial product with powerful DRM and activation options.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

☑ Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

☑ Create hardware-locked keys (keys are based on a unique system ID that must be provided)

☐ Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

☐ Use portable mode (store license in a .LIC file instead of the Windows registry)

```
[Info] Built 64-bit DLL: [redacted]\Examples\Excel\01_HelloWorld\bin\HelloWorldrun64.dll
[Info] Built satellite DLL[redacted]k\Examples\Excel\01_HelloWorld\bin\HelloWorld.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

Basic Licensing

Set up the core rules for your application access.

- Activation Settings
- Advanced Activation
- Hardware ID Options

Key Management

Generate and control license keys for your users.

- Key Generator
- Deactivation Control
- EULA Settings

Online Ecosystem

Automate your licensing with server-side validation.

- Online Activation
- Online Deactivation
- Online Validation

23.3 Need More Help?

Features Overview

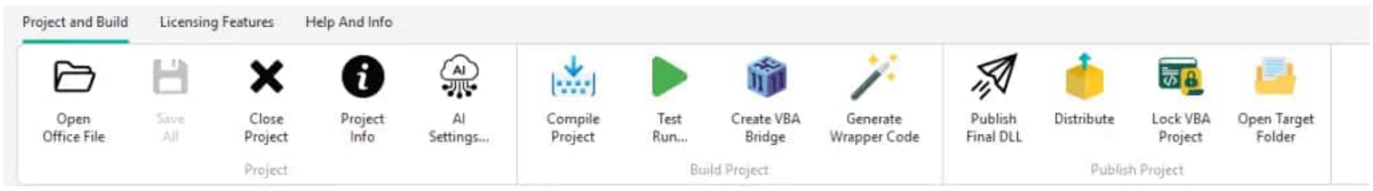
Understand the capabilities of VBA Padlock. View Features →

API Reference

Documentation for the `VBAPL_*` functions. View API →

24. Project and Build

The **Project and Build** ribbon tab is the heart of VBA Padlock Studio. It serves as your primary command center for the entire development lifecycle, providing all the tools necessary to transform raw VBA code into a professional, protected software product.



24.1 🔄 The Standard Workflow

Every successful project follows a logical path from initial setup to end-user delivery.

1. **Initialize:** Open an Office file and set your Project Information.
2. **Develop:** Write your sensitive logic in the Code Editor using native VBA syntax.
3. **Validate:** Compile your scripts and use the Test Runner to ensure everything works as expected.
4. **Connect:** Generate and inject the VBA Bridge to link your DLL with Microsoft Office.
5. **Publish:** Build your Final DLLs and package them for Distribution.

24.2 🛠️ Workspace & Tools

📖 Project Foundation

Configure the identity and structure of your application.

- Welcome Screen
- Main Window
- Project Info

Scripting Engine

Write protected code with full access to the Office Object Model.

- Code Editor
- Script Templates
- AI Assistant Settings

Build & Debug

Turn source code into secure bytecode and verify performance.

- Compilation Panel
- Test Runner
- VBA Bridge Injection

Deployment

Prepare your application for end-user installation.

- Publish Final DLL
- Distribution Options
- VBA Project Locking
- Setup Creator

24.3 See Also

Licensing Setup

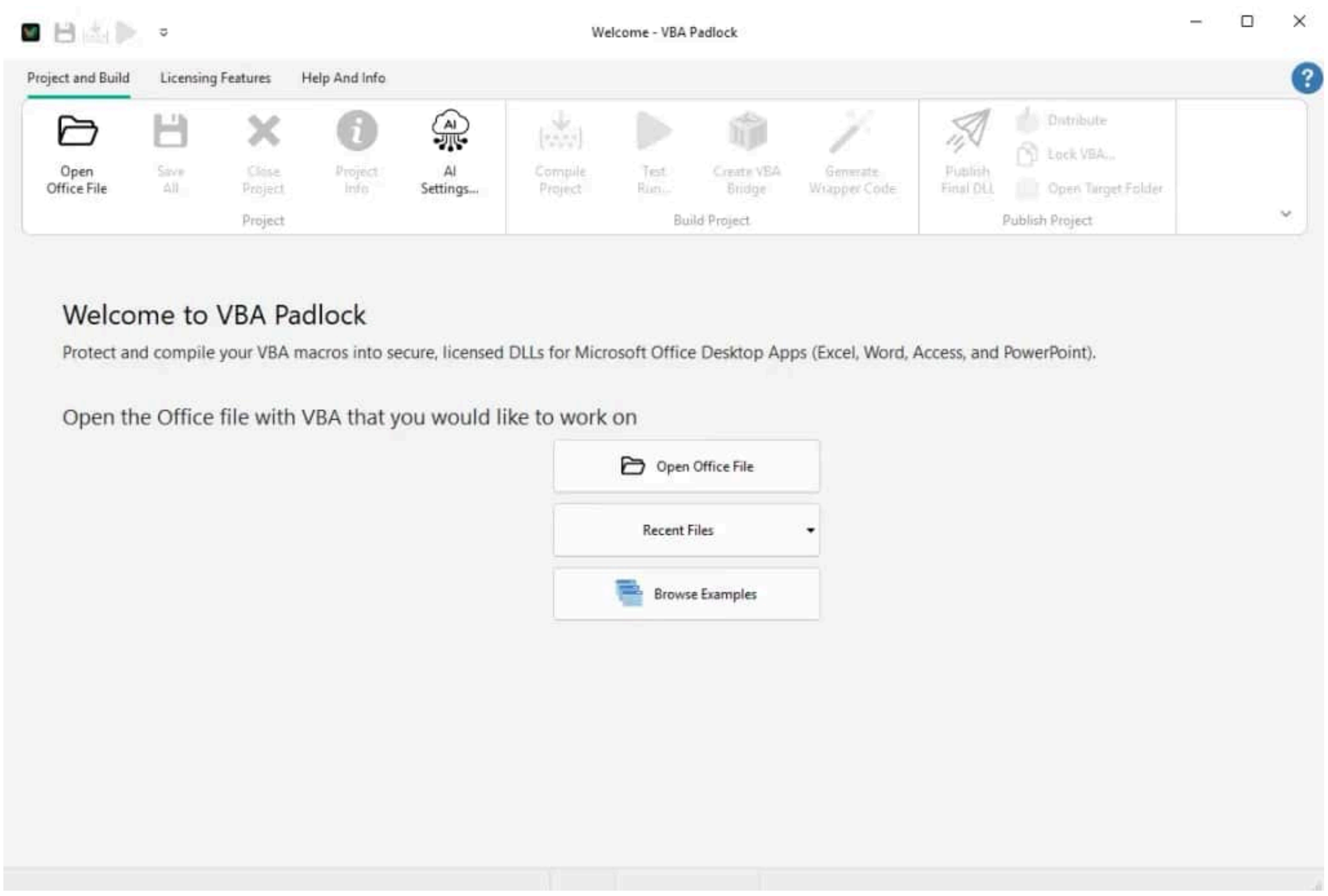
Learn how to add activation keys and hardware locking. View Licensing →

VBA Bridge API

Full reference for calling your DLL from Office. View API →

25. Welcome Screen

The **Welcome Screen** is the first thing you see when launching VBA Padlock Studio. It is designed to be your launchpad, getting you into your workspace as quickly as possible.



Welcome to VBA Padlock

Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

25.1 🚀 Jumpstart Your Project

Whether you're starting from scratch or returning to a masterpiece, the Welcome Screen provides direct routes to your goal:

- **Open Office File:** The heart of the workflow. Select your Excel, Word, Access, or PowerPoint file to instantly initialize a protection project.
- **Recent History:** Resume work on your active projects. The interface keeps a curated list of your most recently accessed files for one-click access.
- **Ready-made Examples:** Master the art of protection by exploring our built-in templates and sample projects, covering every major Office application.

Tip

Power User Hint: You can **drag and drop** any macro-enabled Office file (`.xlsm`, `.docm`, etc.) directly onto the Welcome Screen to skip the file picker and open the project immediately.

25.2 The Initialization Workflow

1. **Launch:** Start VBA Padlock Studio from your desktop or start menu.
2. **Import:** Select or drop your target Office document onto the workspace.
3. **Detect:** The Studio automatically scans for existing project metadata or initializes a new `.vbapadlock` project.
4. **Enter:** You are instantly transported to the Main Window with your environment ready for development.

25.3 See Also

Main Window

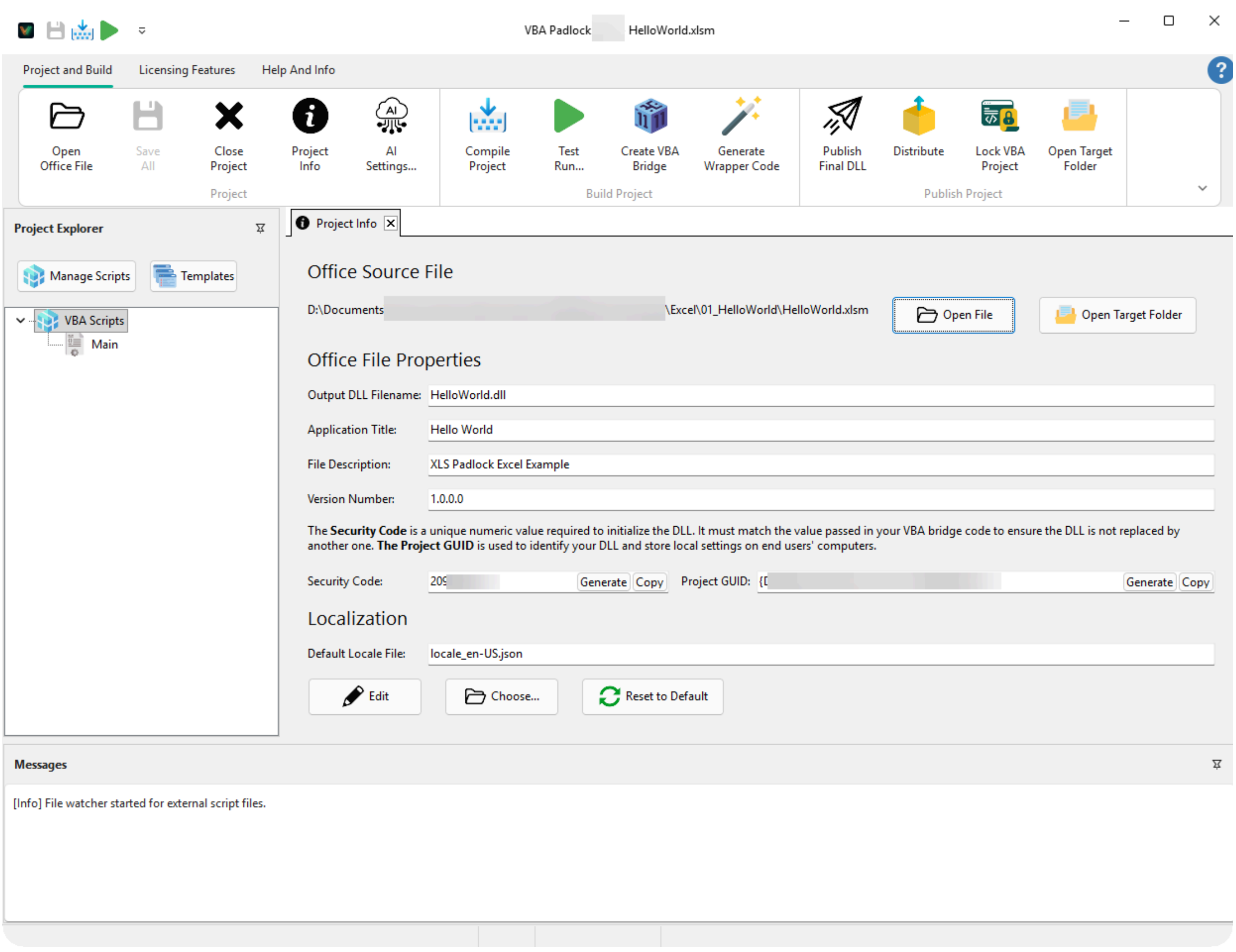
Explore the central cockpit. View Reference →

Quick Start

Follow the first-run guide. Follow Guide →

26. Main Window

The **Main Window** is the central cockpit of VBA Padlock Studio. From here, you manage your project's structure, write your protected code, and access all licensing and compilation tools in a single, unified environment.



Office Source File

D:\Documents[redacted]\Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description: XLS Padlock Excel Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 20[redacted] Project GUID: {[redacted]

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

26.1 🏗 Workspace Layout

The interface is optimized for a high-intensity professional workflow, divided into functional zones:

- **Intelligent Ribbon (Top):** Discover tools through context-aware tabs (**Project and Build**, **Licensing Features**, and **Edit Script**) that adapt to your current task.
- **Project Explorer (Left):** Maintain absolute control over your project's structure. Organize your VBA script modules and satellite resources with ease.
- **Code Editor (Center):** The heart of the Studio. A powerful MDI workspace where you build and refactor your protected intellectual property.

- **Dynamic Help (Right):** Documentation at your fingertips. This panel automatically updates as you navigate, providing instant reference for every button and field.
- **Message Log (Bottom):** Your real-time feedback loop. View detailed compilation logs, security warnings, and runtime reports.

26.2 ⚡ Productivity Shortcuts

Master these essential keys to maintain your development momentum:

Shortcut	Action
`Ctrl + O`	**Open Office File** — Quickly switch between protection projects.
`F5`	**Compile Project** — Trigger the security pipeline and bytecode generation.
`Ctrl+F6`	**Run Test** — Launch your project in the isolated Test Runner.
`Ctrl + S`	**Save All** — Secure your progress by saving all open scripts and settings.

26.3 🛠️ Ribbon Command Center

The Ribbon organizes VBA Padlock Studio's deep feature set into accessible, logical hubs:

Project & Build

Manage the core lifecycle: code editing, compilation, VBA Bridge injection, and final publishing.

Licensing Features

Engineer your business rules: activation logic, hardware locking, and Online Automation.

Edit Tools

Standardize your coding with advanced editor tools (Find, Replace, Indent) that activate whenever code is in focus.

Support & Info

Verify your version, check for updates, and access direct developer support.

26.4 See Also

Project Info

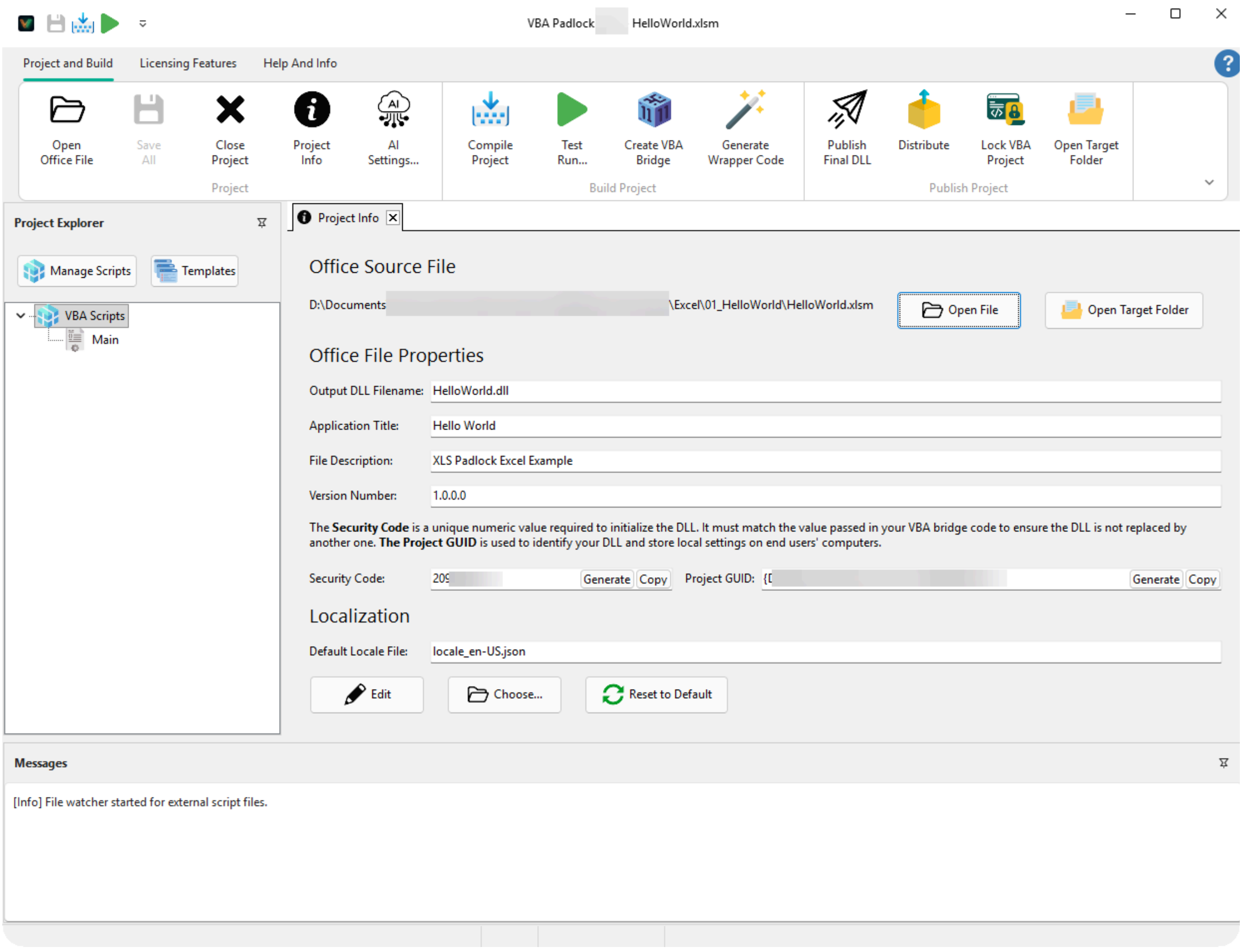
Configure application metadata. View Reference →

Code Editor

Master the scripting workspace. View Reference →

27. Project Info

The **Project Info** panel is the configuration hub for your application. This is where you define its public identity, security parameters, and version information.



Office Source File

D:\Documents[redacted]\Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description: XLS Padlock Excel Example

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code: 20[redacted]

Project GUID: {[redacted]

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

27.1 🏗️ Application Identity

These fields define the metadata that will be embedded into your final DLL.

- **Application Title:** The name of your software as it will appear in activation dialogs and licensing windows.
- **Version Number:** Essential for tracking updates. We recommend the `Major.Minor.Build.Revision` format.
- **File Description:** A brief description visible in the Windows file properties (Right-click DLL → Details).
- **Output DLL Filename:** The physical name of the `.dll` file that will contain your compiled code.

> ⚠ Caution
>
> **VBA Bridge Dependency:** If you change the filename of the DLL, you **must re-inject the VBA Bridge** into your Office file. The name is encoded within the bridge logic to ensure a secure, high-performance connection at runtime.

27.2 🛡 Security & Unique Identity

These settings are critical for the secure "handshake" between your Office file and your protected code.

Security Code

The **Security Code** is the cryptographic link between the VBA Bridge and your DLL. It is a fundamental part of the multi-layered security model.

> ⚠ Caution
>
> If you generate a new code, you **must re-inject the VBA Bridge** into your Office file, otherwise application initialization will fail.

Project GUID

The **Project GUID** is the unique fingerprint of your project. It ensures that a license key generated for this project can never be used for another.

> Danger
>
> **Never change the GUID** after you have started distributing licenses. Any modification will instantly invalidate all keys already sold to your customers.

27.3 📁 File Management

- **Office Source File:** The path to the Excel, Word, or Access file associated with this project. Click the **Open File** button to instantly launch the document in its native Office application—a highly efficient way to verify your VBA Bridge or licensing logic in real-time.
- **Target Folder:** Click the folder button to instantly open your project directory in Windows Explorer.

27.4 🌍 Localization

Customize the language and text for all interfaces displayed to your end users. For a deep dive into how translation works, see the Localization Feature.

Locale File

By default, VBA Padlock uses English. You can provide your own JSON file to translate dialogs (Activation, EULA, Errors).

- **Choose:** Select a custom `.json` locale file.
- **Edit:** Open the built-in translator.
- **Reset:** Revert to the original English texts.

27.5 See Also

VBA Bridge

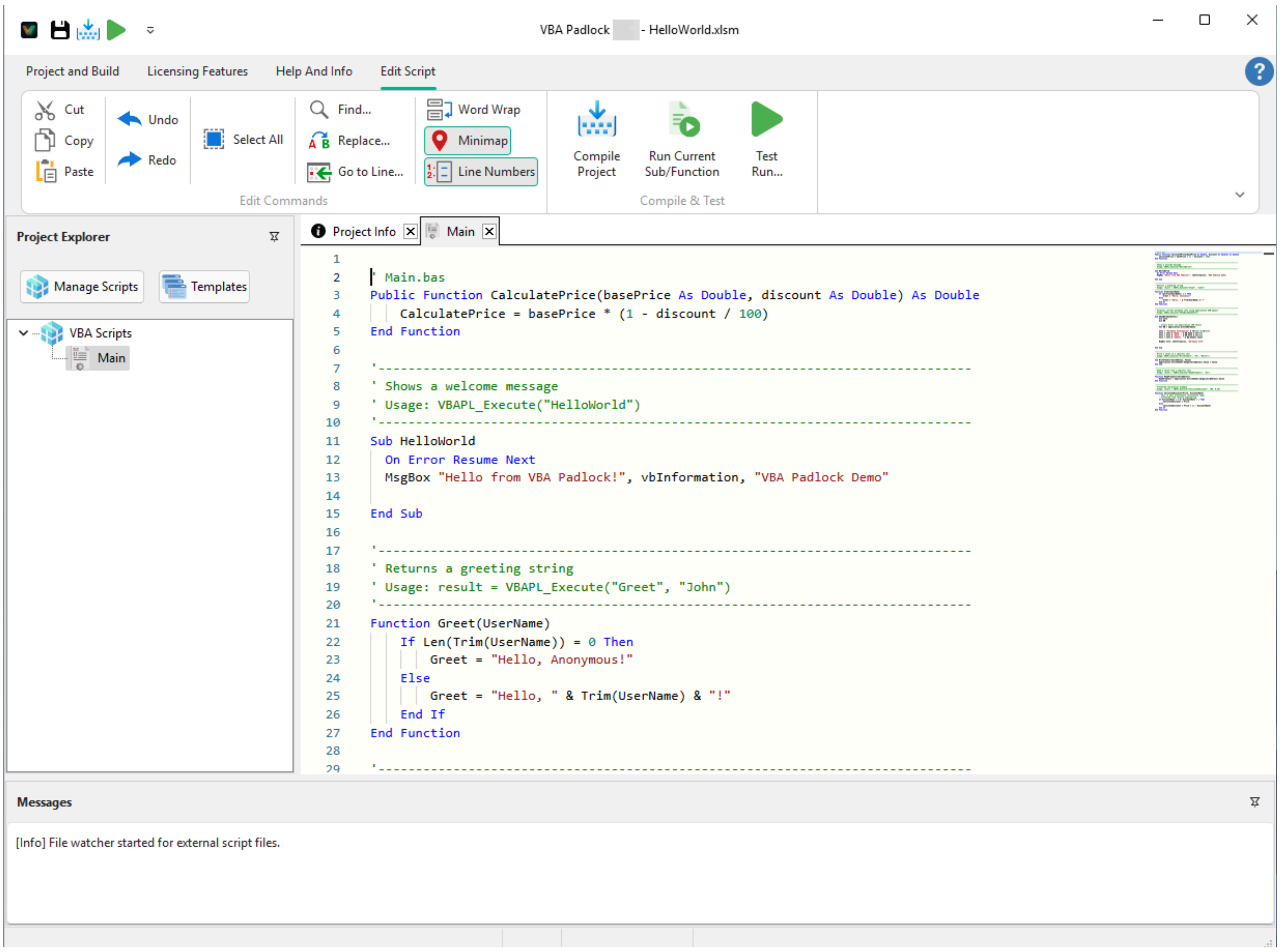
Connect your DLL to Office. View Help →

Publication

Compile and distribute. View Help →

28. Code Editor

The **Code Editor** is where your intellectual property takes shape. It is a full-featured MDI (Multiple Document Interface) workspace designed specifically for writing VBA code that runs within the secure environment of a compiled DLL.



```
' Main.bas
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function

'------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'------------------------------------------------------------------------------
```

[Info] File watcher started for external script files.

28.1 💻 Native Scripting Experience

Writing code in VBA Padlock Studio feels natural for any Office developer, but with the added benefit of compiled security.

- **VBA Syntax Highlighting:** Keywords, strings, comments, and logic operators are color-coded for maximum readability.
- **Real-time IntelliSense:** As you type, the editor suggests built-in VBA functions (`DateAdd`, `Trim`, etc.), constants, and project-specific variables.
- **Gutter Reference:** Line numbers and error markers provide instant visual feedback on your code's structure and health.

Note

DLL Execution Environment: While the syntax is identical to Microsoft Office VBA, your code executes inside the protected DLL. This makes it invisible to end-users and highly resistant to reverse-engineering.

28.2 📦 Module Management

Your project is organized into `.bas` script files, which you can manage from the Project Explorer or the Ribbon.

1. **Main.bas:** The mandatory entry point for every project. This is where your core logic usually resides.
2. **Modular Logic:** You can add as many additional modules as needed to keep helper functions, constants, and business rules organized.
3. **Command Palette (`F1`):** Access advanced editor commands, formatting tools, and quick search from a single search bar.

28.3 ⚡ Productivity Shortcuts

Shortcut	Action
`Ctrl + S`	**Save** — Save all open modules to the project folder.
`Ctrl + F` / `H`	**Find / Replace** — Standard search tools within the active module.
`Ctrl + Z` / `Y`	**Undo / Redo** — Standard history operations.
`F1`	**Command Palette** — Access all IDE commands.
`Ctrl + Space`	**IntelliSense** — Force the code completion window to appear.

28.4 ⚙️ Advanced Features

- **Error Line Highlighting:** When a compilation error is detected, the editor automatically highlights the offending line in red so you can locate the problem at a glance. To edit the highlighted line or dismiss the marker after fixing the issue, right-click anywhere in the editor and select **Clear Error Highlight** from the context menu.
- **Extension Libraries:** Full access to the built-in License Library for checking activation status and Hardware IDs directly from your code.

28.5 See Also

Script Functions

View the full API reference. Browse Reference →

VBA Compatibility

What's supported in the DLL? View Compatibility →

Script Libraries

Office constants & utilities. Browse Libraries →

29. Script Templates

The **Script Templates** workspace is designed to accelerate your development by providing tested, ready-made code blocks. Instead of writing complex security checks from scratch, you can insert high-quality patterns directly into your project.

29.1 ⚡ Accelerated Development

Templates provide a reliable foundation for the most frequent tasks in VBA Padlock Studio:

- **License Validation:** Standard boilerplate for checking if a license is valid and active.
- **Hardware Protection:** Pre-written routines for validating System IDs and hardware fingerprints.
- **Initialization & Cleanup:** Essential setup code for the handshake between Office and your DLL.
- **Data Processing:** Common patterns for handling strings, dates, and math within the protected environment.

29.2 📝 The Preview & Insert Workflow

The interface is split into two logical areas to help you choose the right pattern:

1. **Selection Gallery:** Browse through categorized templates on the left side of the window.
2. **Live Preview:** Click any template to see its source code and technical documentation in the central preview pane before committing.
3. **One-Click Injection:** Click **Add to Project** to instantly create a new `.bas` module containing the selected code.

> **Tip**
>
> **Customization:** After adding a template, you can immediately find it in your Code Editor. We recommend reviewing and customizing the logic to perfectly match your application's unique requirements.

29.3 See Also

Code Editor

Refactor your inserted templates. View Reference →

Script Functions

View the full API for templates. View Reference →

Script Libraries

Constants for Office automation. Browse Libraries →

30. AI Settings

The **AI Settings** panel is the bridge between your secure development environment and modern Artificial Intelligence. By enabling the **Model Context Protocol (MCP)**, you allow AI assistants like Claude, ChatGPT, and Gemini to understand, refactor, and help compile your protected VBA logic.

30.1 🤖 The MCP Hub

The MCP Server allows AI tools to "see" your project files and provide contextually aware code generation.

- **Server Management:** Control when the AI bridge is active. You can start/stop the server manually or set it to launch automatically with VBA Padlock Studio.
- **Port Configuration:** Defaulting to port `8765`, this setting defines the local communication channel for your AI agents.
- **Live Status:** Monitor whether your AI assistant is currently connected and ready to assist.

> 🚀 **Dual AI Advantage**
>
> **Direct Folder Access:** Because your source modules (`.bas`) are stored as plain text within your project folder, you can also point external tools like **Claude Code**, **Cursor**, or **VS Code** directly to your `.vbapadlock` directory for native editing without any complex configuration.

30.2 🔌 Provider Integration

VBA Padlock Studio provides pre-formatted configuration snippets for the major AI ecosystems. Copy these into your provider's settings to activate the integration:

Optimized for the Claude Desktop application. Copy the JSON into your `claude_desktop_config.json` file.

Standard configuration for OpenAI-compatible agents.

Reference configuration for integration with Gemini-based development tools.

30.3 🛠️ The Connection Workflow

1. **Open** the AI Settings dialog from the ribbon.
2. **Start** the MCP Server to open the communication channel.

3. **Copy** the configuration JSON specifically formatted for your chosen AI tool.
4. **Paste** the settings into your assistant’s configuration file.
5. **Develop** at lightning speed with an AI that understands your entire project context.

30.4 See Also

Code Editor

Write code with your AI partner. View Reference →

Compile Project

Validate AI-generated code. View Reference →

31. Compile Project

Compilation is the fundamental step in the VBA Padlock Studio security model. When you trigger the **Compile Project** command (`F5`), your human-readable VBA scripts are transformed into proprietary, encrypted bytecode. This ensures that even if your DLL were analyzed, your original source code remains hidden. Learn more about how this works in the VBA Compilation Feature guide.

31.1 The Security Pipeline

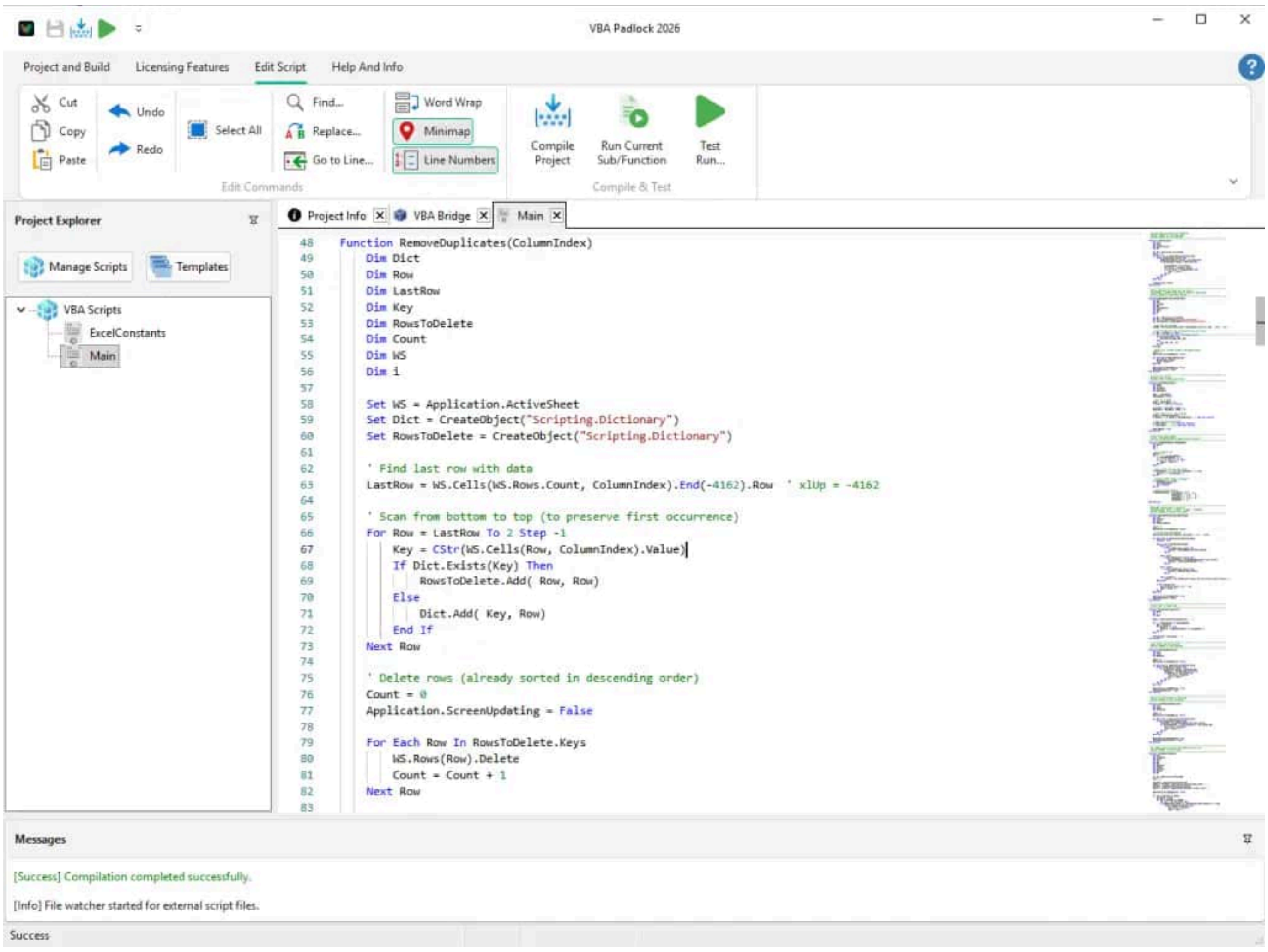
A successful compilation is the prerequisite for all subsequent steps in the build process.

1. **Parsing:** VBA Padlock Studio reads all `.bas` modules and checks for syntax compatibility.
2. **Analysis:** The engine performs semantic checks to ensure functions, variables, and libraries are correctly referenced.
3. **Bytecode Generation:** Valid code is transformed into highly secure, intermediate bytecode.
4. **Verification:** The resulting `.vbaplcode` file is verified to ensure it is ready for testing or final deployment.

31.2 Message Intelligence

The **Messages** panel at the bottom of the interface acts as your real-time diagnostic center. It provides instant feedback on the health of your project:

- **Success (Green):** Your project is secure and ready for testing.
- **Errors (Red):** Critical syntax or logic issues that prevent bytecode generation.
- **Warnings (Yellow):** Potential runtime issues or optimizations you should review.



```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Dim Row
    Dim LastRow
    Dim Key
    Dim RowsToDelete
    Dim Count
    Dim WS
    Dim i

    Set WS = Application.ActiveSheet
    Set Dict = CreateObject("Scripting.Dictionary")
    Set RowsToDelete = CreateObject("Scripting.Dictionary")

    ' Find last row with data
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp = -4162

    ' Scan from bottom to top (to preserve first occurrence)
    For Row = LastRow To 2 Step -1
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Dict.Exists(Key) Then
            RowsToDelete.Add( Row, Row)
        Else
            Dict.Add( Key, Row)
        End If
    Next Row

    ' Delete rows (already sorted in descending order)
    Count = 0
    Application.ScreenUpdating = False

    For Each Row In RowsToDelete.Keys
        WS.Rows(Row).Delete
        Count = Count + 1
    Next Row

```

```
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

Success

Pro Debugging

Instant Navigation: Don't waste time searching for code. **Double-click any error** in the Messages panel to automatically jump to the exact line and module in the Code Editor.

31.3 Quick Actions

Key	Action
F5	**Trigger Compilation** — Rebuild all modules from source.

31.4 See Also

Test Runner

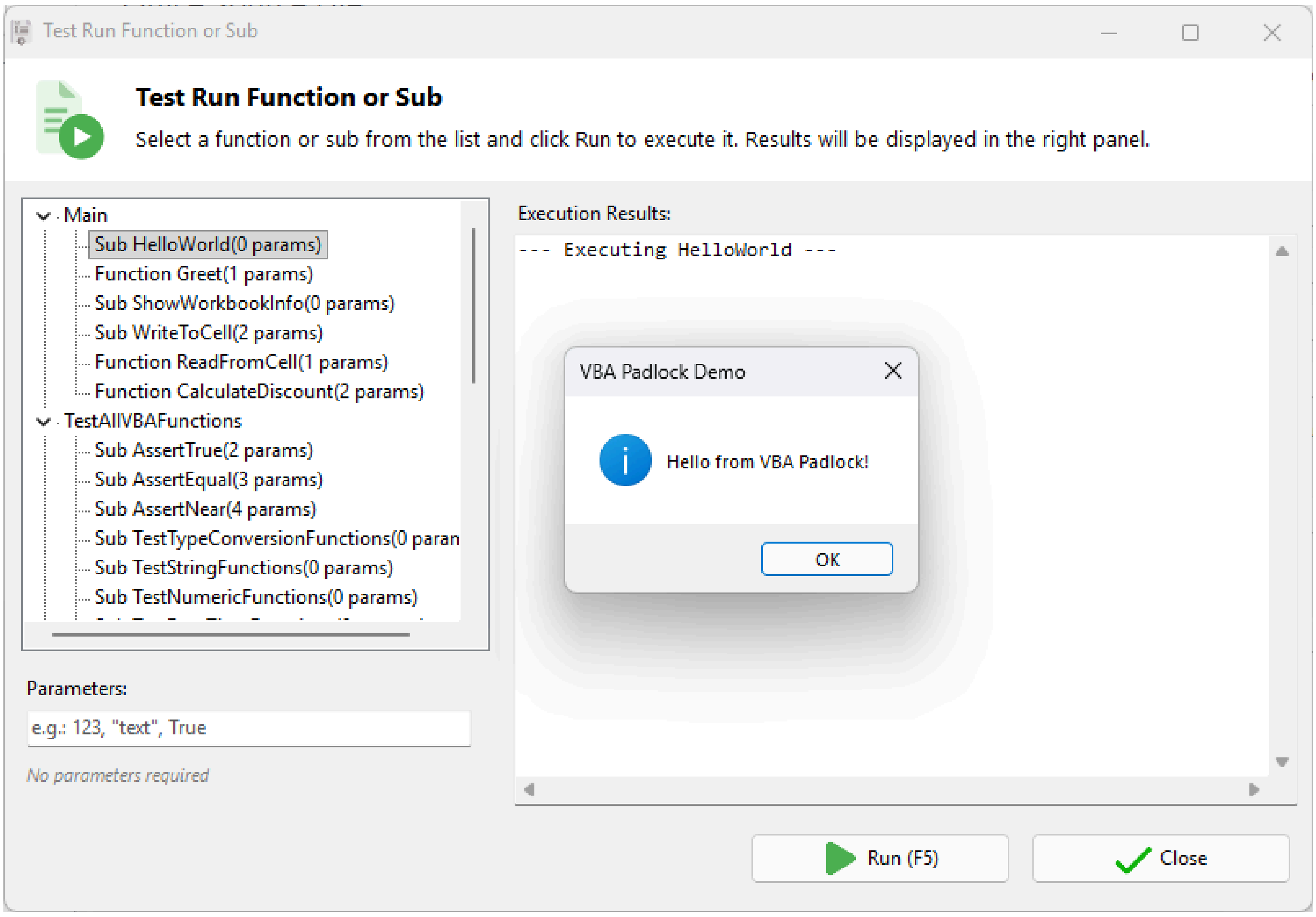
Execute your compiled bytecode. View Reference →

Publish Final DLL

Assembled the final package. View Reference →

32. Test Runner

The **Test Runner** is your rapid prototyping and debugging cockpit. It allows you to execute any function or subroutine from your compiled bytecode in a sterile environment, ensuring your logic is flawless before you ever touch a spreadsheet or document.


Test Run Function or Sub

Test Run Function or Sub

Select a function or sub from the list and click Run to execute it. Results will be displayed in the right panel.

- Main
  - Sub HelloWorld(0 params)
  - Function Greet(1 params)
  - Sub ShowWorkbookInfo(0 params)
  - Sub WriteToCell(2 params)
  - Function ReadFromCell(1 params)
  - Function CalculateDiscount(2 params)
- TestAllVBAFunctions
  - Sub AssertTrue(2 params)
  - Sub AssertEqual(3 params)
  - Sub AssertNear(4 params)
  - Sub TestTypeConversionFunctions(0 paran
  - Sub TestStringFunctions(0 params)
  - Sub TestNumericFunctions(0 params)

Parameters: e.g.: 123, "text", True

No parameters required

Execution Results:

```
--- Executing HelloWorld ---
```

VBA Padlock Demo

Hello from VBA Padlock!

32.1 🛠️ Isolated Execution

Testing in VBA Padlock Studio is significantly faster than testing in Office because it bypasses the application overhead.

- **Routine Browser:** A hierarchical list of every public and private function available in your compiled modules. Select any routine to target it for testing.
- **Live Parameter Entry:** Pass values to your functions in real-time. The Test Runner supports all standard VBA data types.
- **Result Persistence:** The right pane displays return values, execution time, and any console output in a high-contrast, developer-friendly terminal.

> **Note**
>
> **Auto-Compile:** You don't need to manually recompile before every test. The Test Runner detects code changes and triggers a background Compilation automatically if needed.

32.2 ⌨️ Parameter Syntax

Pass multiple arguments to your routines using a simple comma-separated format:

Data Type	Input Example	Notes
Numeric	`42`, `3.1415`	Supports integers and floating points.
String	`"VBA Padlock"`	Wrap text in double quotes.
Boolean	`True`, `False`	Case-insensitive logical values.
Mixed	`101, "Admin", True`	Separate multiple arguments with commas.

32.3 🔄 The Debugging Loop

1. **Launch:** Press `Ctrl+F6` to open the Test Runner.
2. **Select:** Pick the function you just finished writing in the Code Editor.
3. **Input:** Provide test data in the parameter field.
4. **Execute:** Click **Run** or press `F5` again.
5. **Verify:** Inspect the return value to ensure your business logic is correct.

32.4 See Also

Code Editor

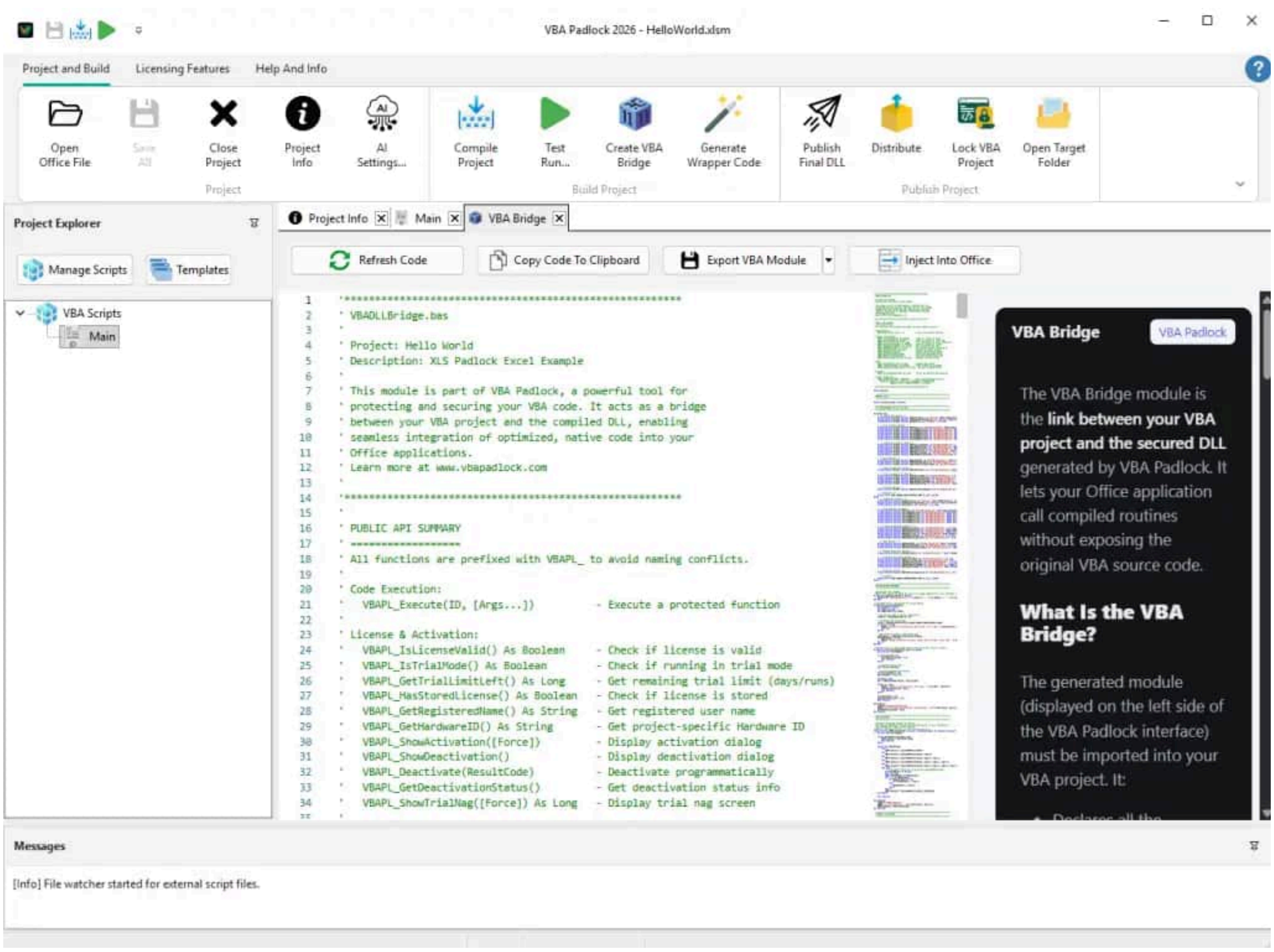
Refine your logic. View Reference →

Script Functions

Check return type requirements. View Reference →

33. VBA Bridge

The **VBA Bridge** is safe, robust, and automated. It generates the `VBAPadlockBridge` module, a specialized piece of VBA code that allows Microsoft Excel, Word, PowerPoint, or Access to securely load and interact with your protected DLL.



```
'*******************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*******************************************************
'
' PUBLIC API SUMMARY
' ------------------
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])         - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean    - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean       - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long    - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean  - Check if license is stored
'   VBAPL_GetRegisteredName() As String  - Get registered user name
'   VBAPL_GetHardwareID() As String      - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])        - Display activation dialog
'   VBAPL_ShowDeactivation()             - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)         - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()        - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long  - Display trial nag screen
'
```

VBA Bridge VBA Padlock

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

[Info] File watcher started for external script files.

33.1 🏗 The Communication Layer

The generated module is complex under the hood, but simple to use. It handles all technical plumbing so you can focus on your code:

- **Dual Architecture Support:** Automatically generates both 32-bit and 64-bit API declarations, with `PtrSafe` support for 64-bit Office, ensuring seamless compatibility across Office versions.
- **Secure Initialization:** Includes logic to safely locate and bind to your satellite DLLs, preventing “DLL not found” errors.
- **The VBAPL_* API:** Provides 20 high-level functions for core tasks like code execution, license validation, and online activation.

Developer Note

Hands Off: Never modify the generated `VBAPadlockBridge` code manually. If you change your project settings (like the Security Code), always use the **Refresh** and **Inject** tools to ensure consistency.

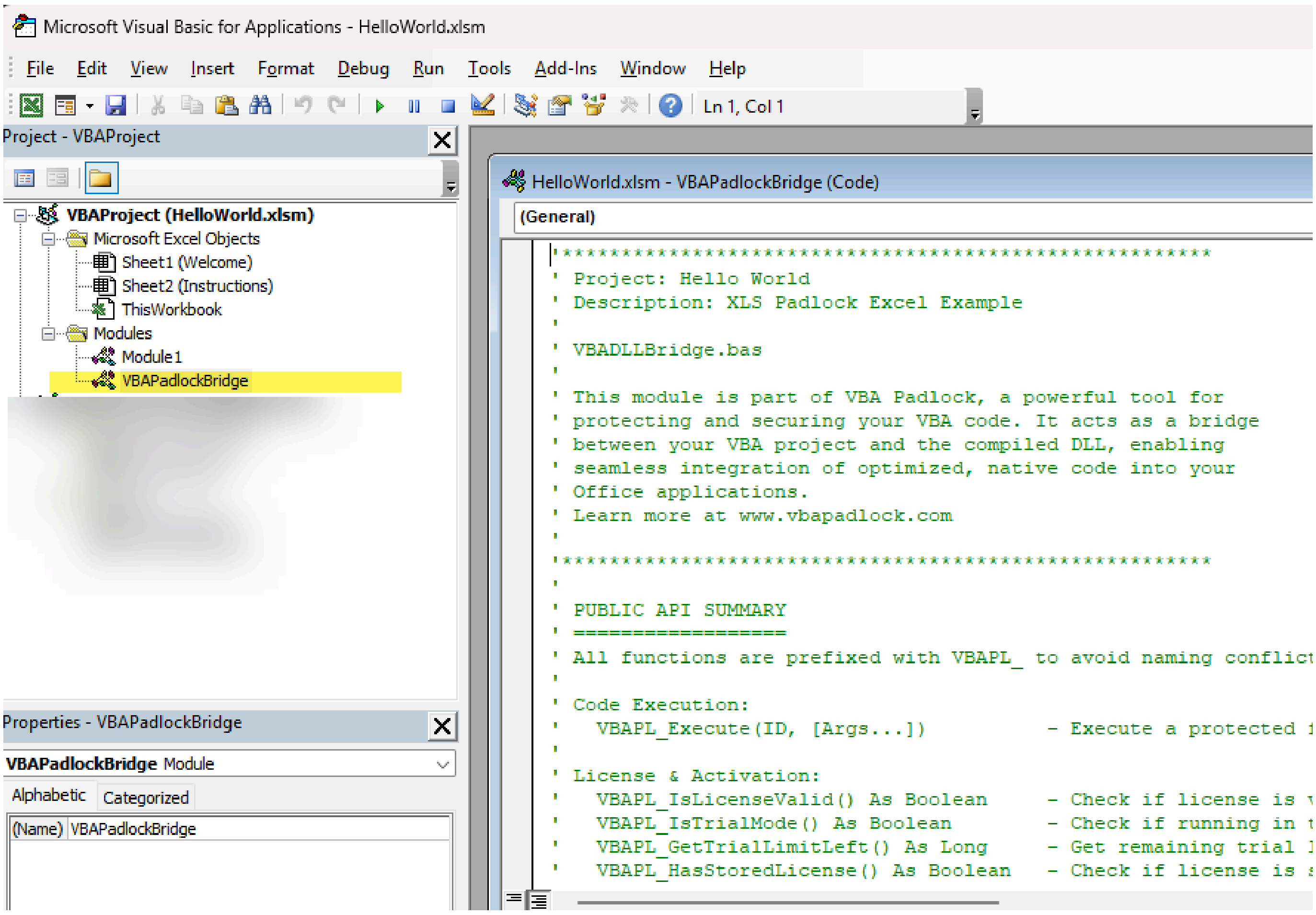
33.2 Injection & Portability

VBA Padlock Studio offers multiple ways to get the bridge code into your Office document:

1. **Direct Injection (Recommended):** Click **Inject Into Office** to have the Studio automatically open your document and insert/update the module for you.
2. **Module Export:** Export the code as a standard `.bas` file for manual import into complex projects.
3. **Clipboard Sync:** Quickly copy the entire module source to manually paste it into the VBA Editor (`Alt + F11`).

33.3 The Connection Workflow

1. **Configure:** Finalize your settings in Project Info, particularly the Security Code.
2. **Generate:** Open the VBA Bridge workspace to see the customized code for your project.
3. **Inject:** Click the **Inject Into Office** button for the most automated experience.
4. **Confirm:** Open your Office file's VBA Editor and verify that the `VBAPadlockBridge` module is present and valid.


Microsoft Visual Basic for Applications - HelloWorld.xlsm
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - VBAProject
VBAProject (HelloWorld.xlsm)
Microsoft Excel Objects
Sheet1 (Welcome)
Sheet2 (Instructions)
ThisWorkbook
Modules
Module1
VBAPadlockBridge
Properties - VBAPadlockBridge
VBAPadlockBridge Module
Alphabetic Categorized
HelloWorld.xlsm - VBAPadlockBridge (Code)
(General)

(Name) VBAPadlockBridge

```
'*******************************************************
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' VBADLLBridge.bas
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*******************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflict
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected f
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is v
'   VBAPL_IsTrialMode() As Boolean        - Check if running in t
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial l
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is s
```

33.4 See Also

VBA Bridge API

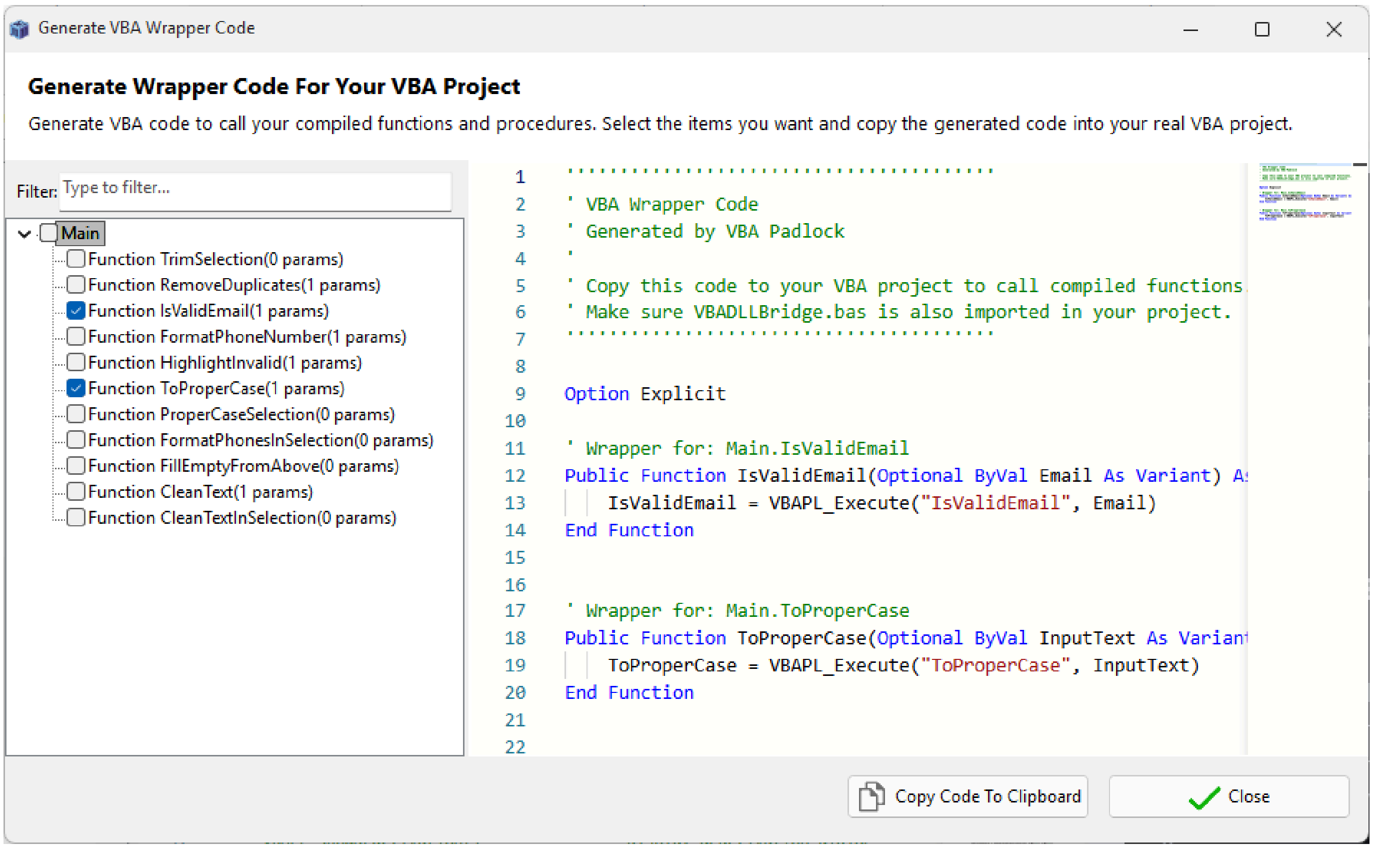
View the full function list. View Reference →

Wrapper Generator

Create typed calling functions. View Reference →

34. VBA Wrapper Generator

The **VBA Wrapper Generator** is a productivity tool that bridges the gap between generic API execution and elegant coding. Instead of calling your protected logic through complex string-based commands, this tool generates a native “facade” for your Office VBA project.



Generate Wrapper Code For Your VBA Project

Generate VBA code to call your compiled functions and procedures. Select the items you want and copy the generated code into your real VBA project.

Filter: Type to filter...

- ☐ Main
  - ☐ Function TrimSelection(0 params)
  - ☐ Function RemoveDuplicates(1 params)
  - ☑ Function IsValidEmail(1 params)
  - ☐ Function FormatPhoneNumber(1 params)
  - ☐ Function HighlightInvalid(1 params)
  - ☑ Function ToProperCase(1 params)
  - ☐ Function ProperCaseSelection(0 params)
  - ☐ Function FormatPhonesInSelection(0 params)
  - ☐ Function FillEmptyFromAbove(0 params)
  - ☐ Function CleanText(1 params)
  - ☐ Function CleanTextInSelection(0 params)

```
''''''''''''''''''''''''''''''''''''''''
' VBA Wrapper Code
' Generated by VBA Padlock
'
' Copy this code to your VBA project to call compiled functions
' Make sure VBADLLBridge.bas is also imported in your project.
''''''''''''''''''''''''''''''''''''''''

Option Explicit

' Wrapper for: Main.IsValidEmail
Public Function IsValidEmail(Optional ByVal Email As Variant) A
    IsValidEmail = VBAPL_Execute("IsValidEmail", Email)
End Function


' Wrapper for: Main.ToProperCase
Public Function ToProperCase(Optional ByVal InputText As Variant
    ToProperCase = VBAPL_Execute("ToProperCase", InputText)
End Function

```

34.1 💎 Type-Safe Automation

By default, the VBA Bridge uses a generic function called `VBAPL_Execute`. While powerful, it lacks IntelliSense support and type safety. The Wrapper Generator solves this:

- **IntelliSense Support:** Generated wrappers use the exact same function names and parameter types as your source code, enabling full auto-complete in the Office VBA Editor.
- **Automatic Selection:** The **Script Tree** allows you to toggle checkboxes for only the functions you want to expose to your Office UI.
- **Real-time Preview:** See the generated code update instantly as you filter and select your routines.

34.2 🔄 Before & After

Traditional (Manual) Call:

```
Result = VBAPL_Execute("Financials|CalculateTax", 5000, 0.2)
```

With Generated Wrapper:

```
' Clean, typed, and fully supported by IntelliSense
Result = CalculateTax(5000, 0.2)
```

34.3 Productivity Features

Smart Filtering

Quickly find specific routines in large projects by typing in the filter field.

One-Click Copy

Review the code in the built-in preview and copy it to your clipboard with a single click.

34.4 The Integration Workflow

1. **Compile:** Ensure your latest changes are built into bytecode.
2. **Select:** Open the Wrapper Generator and check the functions you wish to call from Office.
3. **Sync:** Copy the generated code and paste it into a standard module in your Office file.
4. **Develop:** Start calling your protected functions as if they were native VBA routines.

34.5 See Also

VBA Bridge

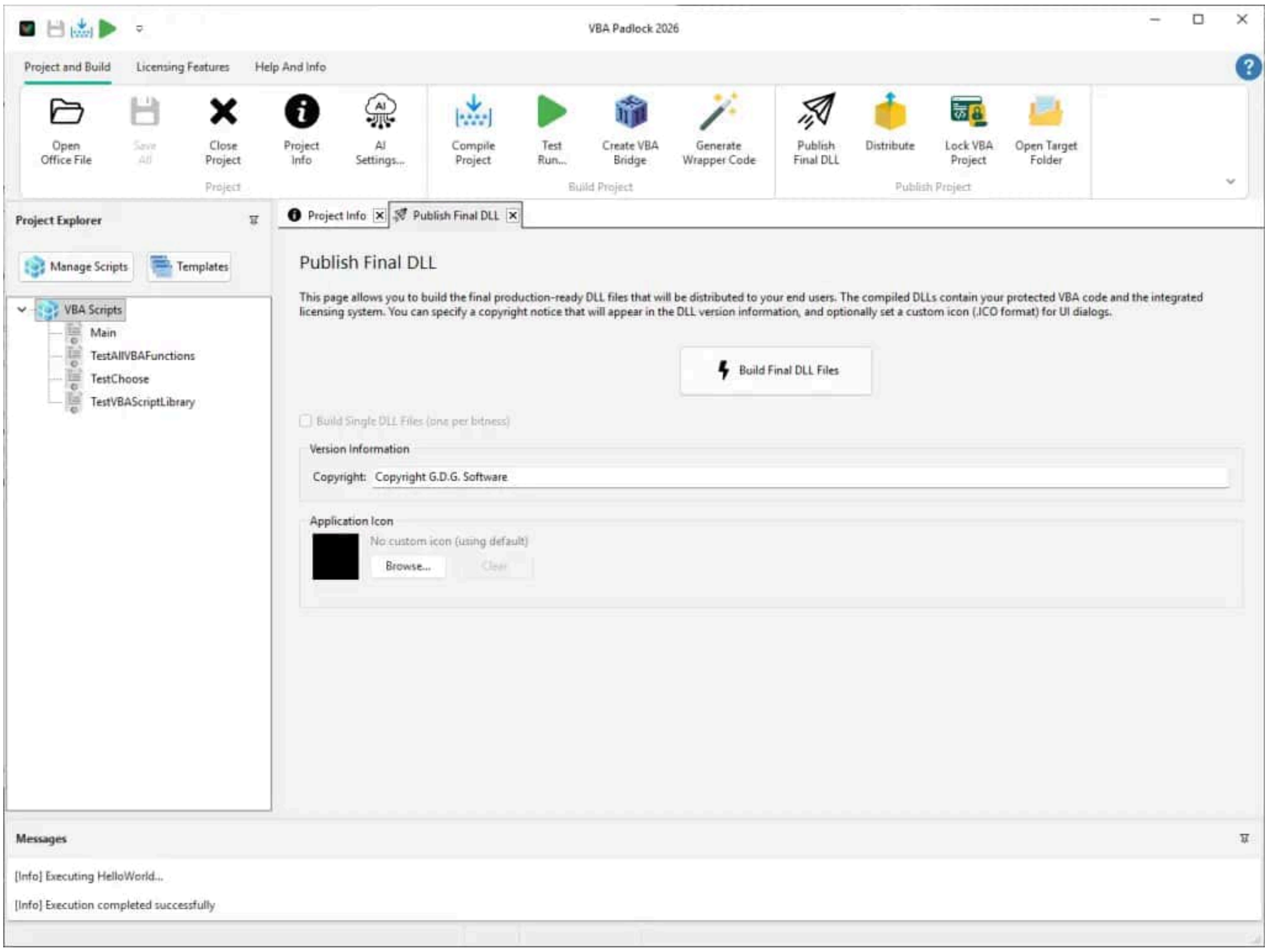
The engine behind the wrappers. View Reference →

Code Editor

Define your public functions. View Reference →

35. Publish Final DLL

The **Publish Final DLL** workspace is the finish line of your development cycle. This is where your compiled bytecode, project metadata, and security settings are fused into a professional software package.



Publish Final DLL

This page allows you to build the final production-ready DLL files that will be distributed to your end users. The compiled DLLs contain your protected VBA code and the integrated licensing system. You can specify a copyright notice that will appear in the DLL version information, and optionally set a custom icon (.ICO format) for UI dialogs.

☐ Build Single DLL Files (one per bitness)

Version Information

Copyright: Copyright G.D.G. Software

Application Icon

No custom icon (using default)

[Info] Executing HelloWorld...
[Info] Execution completed successfully

ⓘ Compile vs. Publish

Publish and **Compile** perform essentially the same compilation process. The key difference is that Publish **disables Hot Reloading**, which is the development feature that lets you recompile and instantly test changes without restarting Office. Publish is designed to generate the final, production-ready DLL for distribution to your end-users.

35.1 🏗️ The Final Assembly

Publishing is more than just saving files; it is a high-security procedure that ensures the integrity of your application:

- **Cryptographic Fusion:** Your bytecode is embedded into the satellite DLL with internal integrity checks.

- **Digital Trust:** The runtime components are digitally signed, helping to prevent antivirus false positives and ensuring the authenticity of your software.
- **Version Embedding:** Metadata like copyright and file descriptions are permanently baked into the DLL binary.

35.2 🎨 Professional Branding

Make your software stand out and look like a native Windows application.

- **Custom Application Icon:** Replace the default VBA Padlock icon with your company logo or project-specific artwork (supports standard `.ico` files).
- **Legal & Metadata:** Define the copyright string that will be visible in the **Windows File Properties → Details** tab. This provides a clear "owned by" signal to your end-users.

35.3 📂 Output Architecture

When you click **Build Final DLL Files**, VBA Padlock Studio generates a standard tripartite structure in your `bin\` directory:

Component	Responsibility
[Project]run32.dll	The 32-bit execution engine for users on older Office installations.
[Project]run64.dll	The high-performance 64-bit engine for modern Microsoft 365 and Office 2024 environments.
[Project].dll	The secure satellite containing your proprietary compiled code.

Tip

Distribution Tip: You should always distribute both the 32-bit and 64-bit runtimes along with your satellite DLL to ensure your software "just works" on any customer machine.

35.4 🔄 The Publishing Workflow

1. **Validate:** Ensure Compilation is successful.
2. **Brand:** Set your custom icon and confirm the copyright metadata.
3. **Build:** Click **Build Final DLL Files** to generate the binaries.
4. **Verify:** Check the `bin\` folder relative to your Office project to see your new production-ready files.

35.5 See Also

Distribution

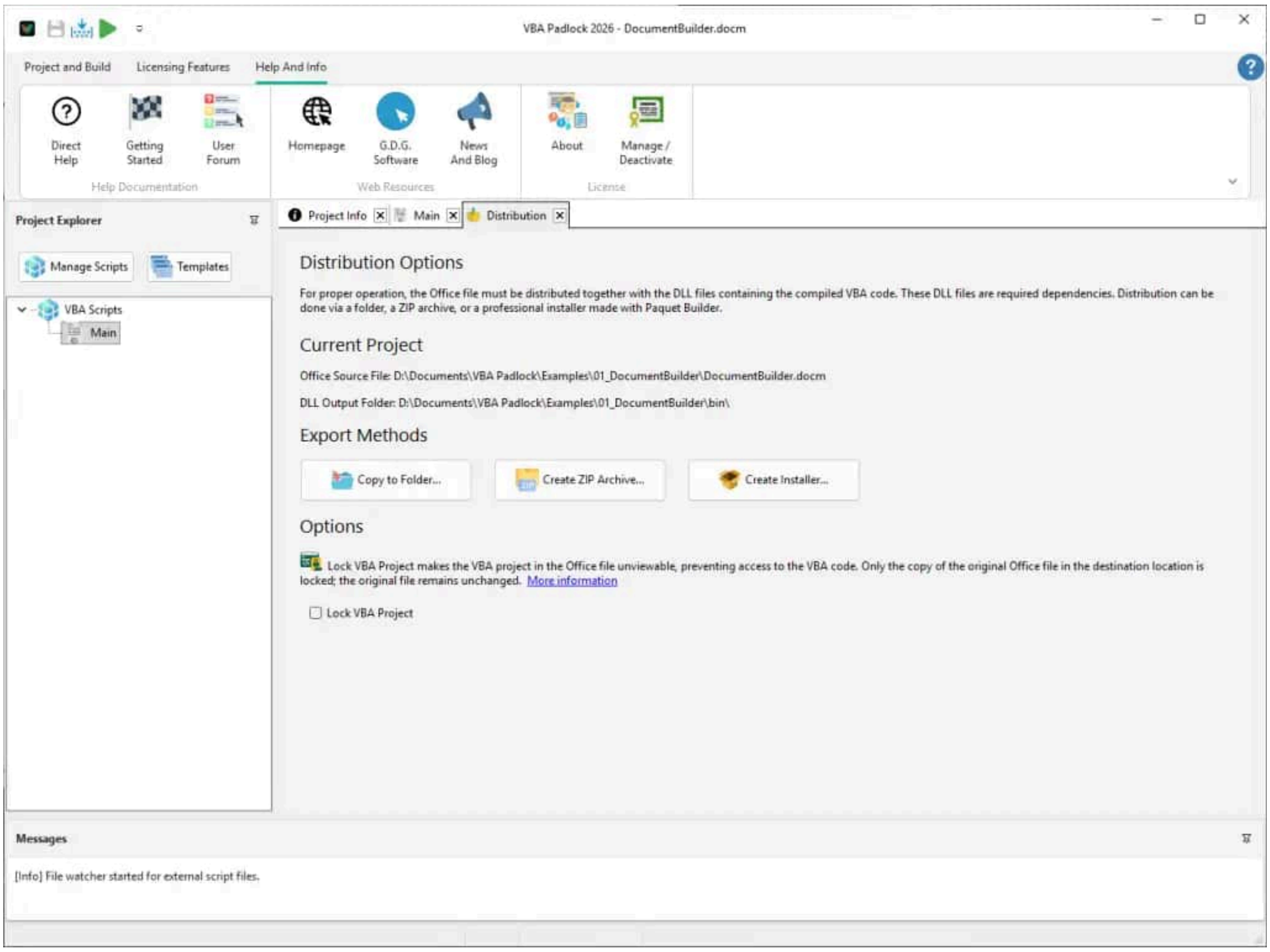
Package your files for users. View Reference →

Create Installer

Build a custom setup program. View Reference →

36. Distribution

The **Distribution** workspace is where you prepare your finalized software for the world. It automates the task of gathering your Office files, protected runtimes, and secure bytecode into a single, clean package ready for your end-users.



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

```
[Info] File watcher started for external script files.
```

36.1 📦 Packaging Strategies

Choose the distribution method that best fits your customer's infrastructure and technical profile:

- **ZIP Archive:** Perfect for lightweight, "green-software" distribution. It creates a single compressed file that users can extract anywhere on their machine.
- **Direct Folder Copy:** Ideal for manual integration into existing document management systems or local network drives.
- **Professional Setup:** Integrated access to the Installer Creator for a traditional Windows installation experience.

⚠ **Caution**

Integrity Warning: The generated `bin\` folder and its three DLLs are the heart of your application. They **must always** be distributed exactly as generated, in a subfolder relative to your Office document.

36.2 🔬 Anatomy of a Release

A standard distribution package follows a strict hierarchy to ensure the VBA Bridge can always locate its security engine:

```
Project_Package/
├── Financial_Model.xlsm     ← Your Office file (featuring the VBA Bridge)
└── bin/                     ← The logic powerhouse
    ├── ProjectName.dll      ← Your secure bytecode
    ├── ProjectNamerun32.dll ← 32-bit execution engine
    └── ProjectNamerun64.dll ← 64-bit execution engine
```

36.3 🛡️ Enhanced Protection

- **Lock VBA Project:** Ensure even your uncompiled VBA “glue” code is safe. When checked, VBA Padlock Studio will apply proprietary locking to the VBA project inside your Office file during the distribution process, making it inaccessible to curious users.

36.4 🔄 The Delivery Workflow

1. **Finalize:** Ensure you have clicked **Build Final DLL Files** in the Publishing tab.
2. **Toggle:** Choose whether to lock the VBA project for extra security.
3. **Bundle:** Select **Create ZIP Archive** for the most common distribution format.
4. **Confirm:** Open the output ZIP and verify that the Office file and the `bin\` folder are both present.

36.5 See Also

Lock VBA Project

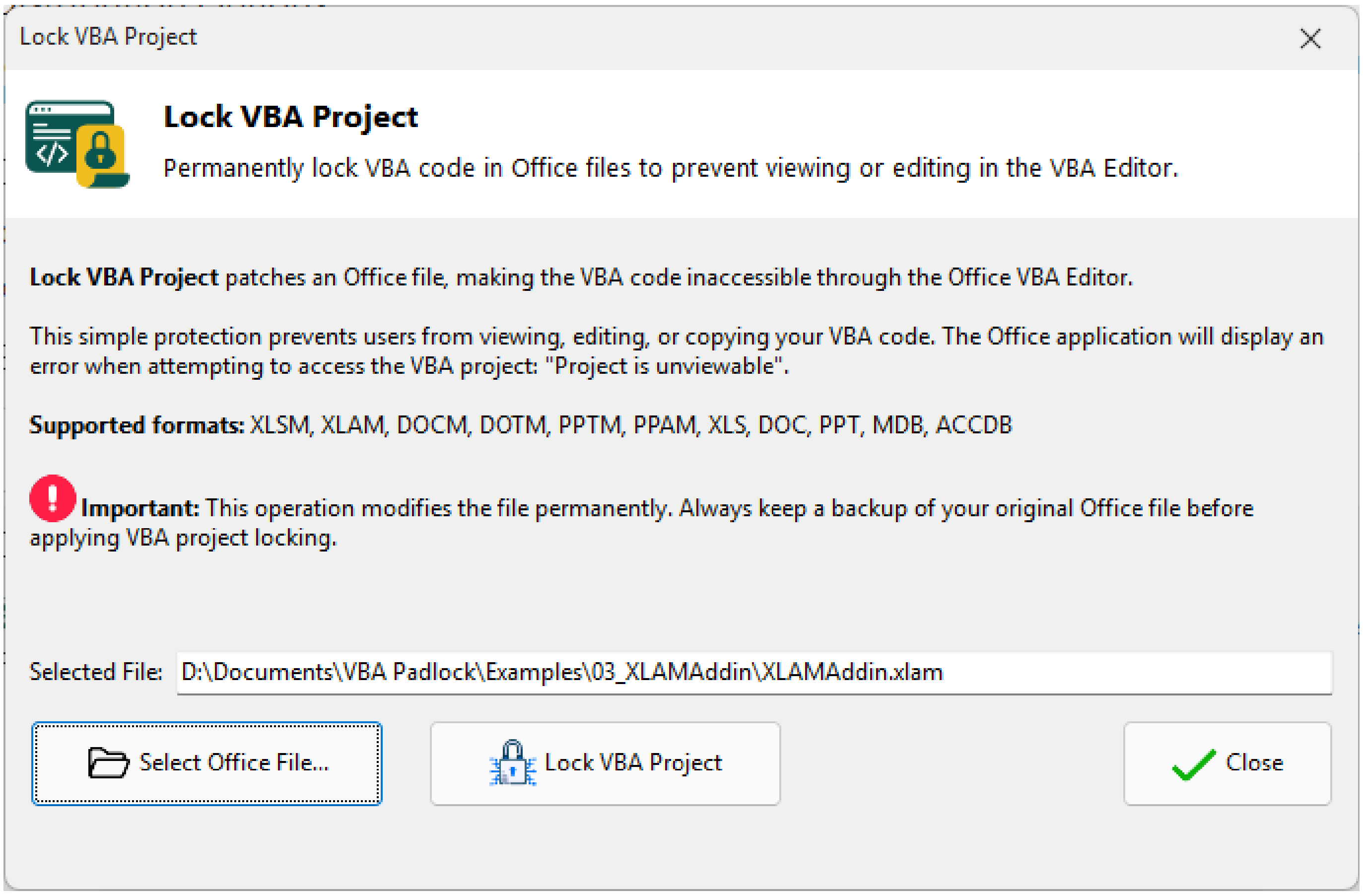
Learn about advanced locking. View Reference →

Create Installer

Build a full setup program. View Reference →

37. Lock VBA Project

While your core intellectual property is safely compiled into a DLL, the “VBA Bridge” code remains inside your Office file. The **Lock VBA Project** tool provides a crucial layer of defense-in-depth by making the VBA Editor interface inaccessible to end-users.



Permanently lock VBA code in Office files to prevent viewing or editing in the VBA Editor.

Lock VBA Project patches an Office file, making the VBA code inaccessible through the Office VBA Editor.

This simple protection prevents users from viewing, editing, or copying your VBA code. The Office application will display an error when attempting to access the VBA project: "Project is unviewable".

Supported formats: XLSM, XLAM, DOCM, DOTM, PPTM, PPAM, XLS, DOC, PPT, MDB, ACCDB

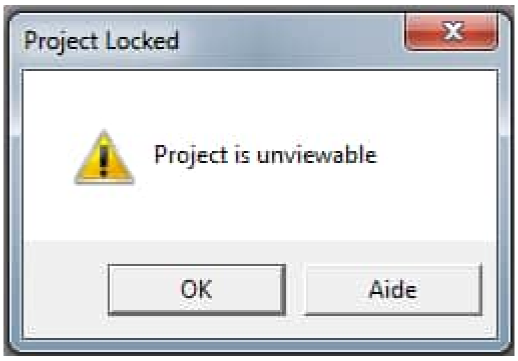
Important: This operation modifies the file permanently. Always keep a backup of your original Office file before applying VBA project locking.

Selected File: D:\Documents\VBA Padlock\Examples\03_XLAMAddin\XLAMAddin.xlam

37.1 🔒 Total Interface Lockdown

This tool is designed to stop curious users from viewing your bridge logic or trying to bypass licensing checks through the VBA Editor.

- **Proprietary Patching:** VBA Padlock Studio applies a specialized lock to the Office document’s metadata, far more reliable than standard Excel password protection.
- **Editor Suppression:** When a user tries to access the VBA Editor (`Alt + F11`), they will be met with an “unviewable project” message, completely blocking access to your code.


Project Locked

Project is unviewable

- **Invisible Glue:** By locking the project, your "glue" code—the part that connects the workbook to the secure DLL—becomes a black box.

> **Safety First**
>
> **Create a Backup:** Project locking modifies the internal structure of your Office file. Always keep an unlocked master copy in a safe location before applying this final layer of protection.

37.2 Format Compatibility

VBA Padlock Studio supports a wide array of legacy and modern Office formats for project locking:

Modern Office

`.xlsm`, `.xlam`, `.docm`, `.dotm`, `.pptm`, `.ppam`, `.accdb`

Legacy Support

`.xls`, `.doc`, `.ppt`, `.mdb`

37.3 The Lockdown Workflow

1. **Select:** Choose the finalized Office document you want to secure.
2. **Backup:** Ensure you have a non-locked version of the file saved elsewhere.
3. **Execute:** Click **Lock VBA Project** to apply the security patch.
4. **Test:** Open the resulting file and try to access the VBA Editor to confirm the lockdown is active.

Tip

Automate: You don't have to do this manually every time. Use the **Lock VBA Project** toggle in the Distribution dialog to include this step in your automatic build process.

37.4 See Also

Distribution

Automate locking during packaging. View Reference →

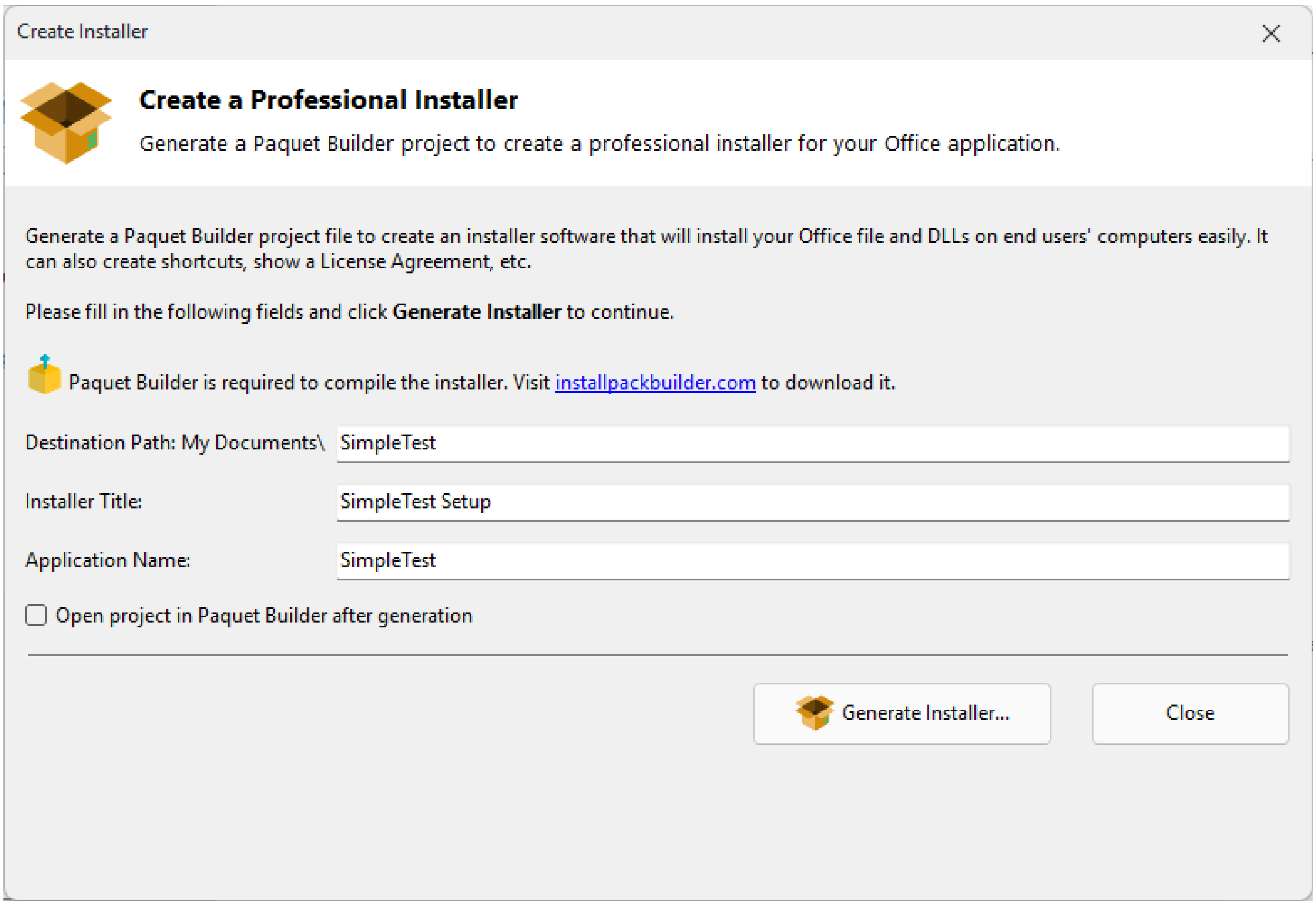
Security Overview

Learn about DLL-level security. View Features →

38. Create Installer

The **Create Installer** workspace is for developers who want to provide a polished, enterprise-grade experience to their customers. Instead of sending a ZIP file, you can generate a complete Windows setup project that handles file placement, shortcuts, and uninstallation.

To open this dialog, click the **Create Installer** button in the Distribution panel.



Create a Professional Installer

Generate a Paquet Builder project to create a professional installer for your Office application.

Generate a Paquet Builder project file to create an installer software that will install your Office file and DLLs on end users' computers easily. It can also create shortcuts, show a License Agreement, etc.

Please fill in the following fields and click **Generate Installer** to continue.

Paquet Builder is required to compile the installer. Visit installpackbuilder.com to download it.

Destination Path: My Documents\ SimpleTest

Installer Title: SimpleTest Setup

Application Name: SimpleTest

☐ Open project in Paquet Builder after generation

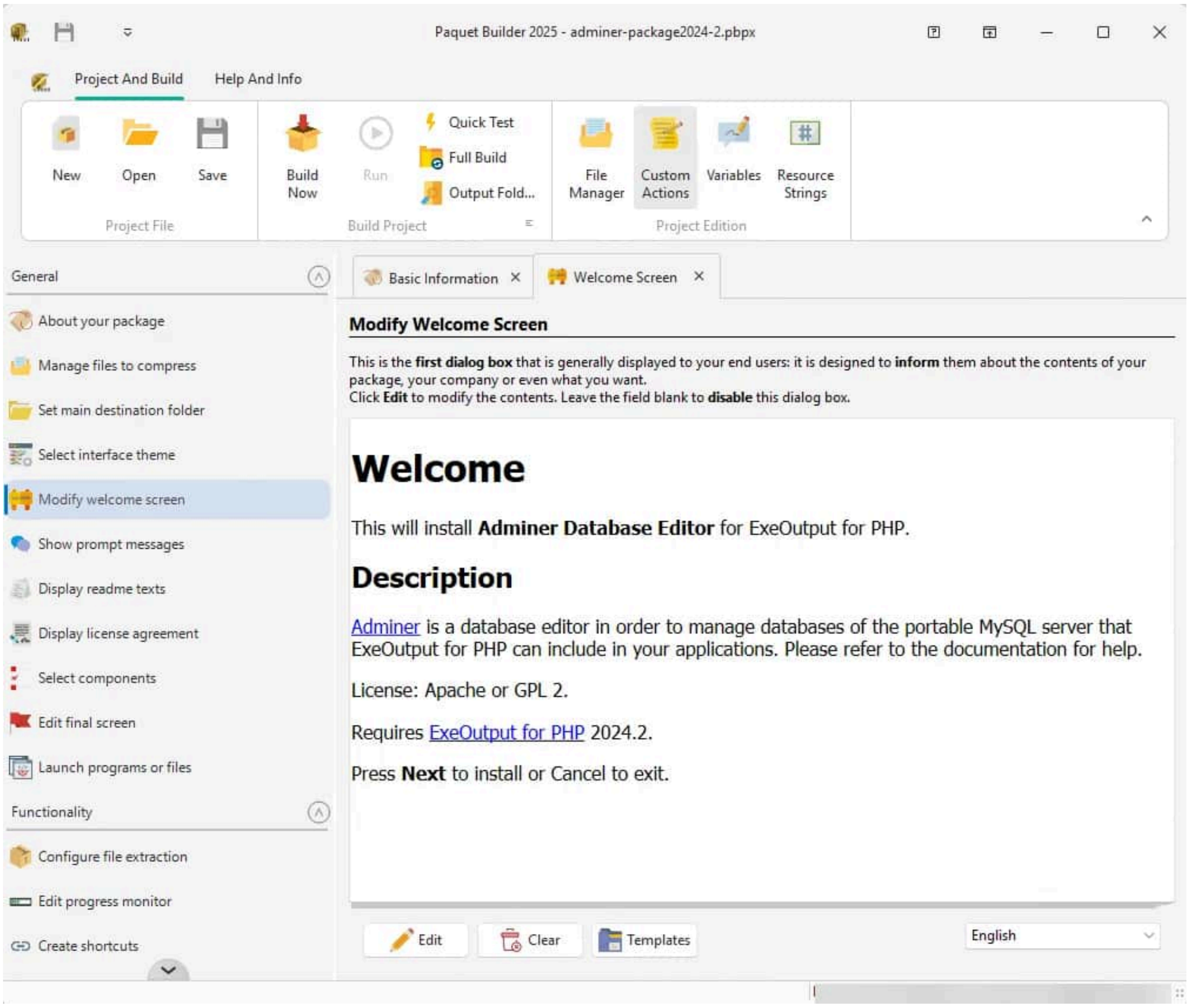
38.1 🏛 Enterprise-Grade Deployment

VBA Padlock Studio integrates with **Paquet Builder** to transform your project files into a standalone `.exe` installer.

- **Smart Directory Mapping:** Automatically organizes your Office file and its `bin\` folder into the correct hierarchy on the target machine.
- **System Integration:** Configure the installer to create Start Menu shortcuts and a professional entry in the Windows “Add/Remove Programs” list.
- **Custom Branding:** Define the installer’s window title and application name to match your corporate identity.

Prerequisite

This feature generates a project file (`.pbpx`) designed for **our installer creator software named Paquet Builder**. You will need Paquet Builder installed to compile the final `.exe` installer.



Modify Welcome Screen

This is the **first dialog box** that is generally displayed to your end users: it is designed to **inform** them about the contents of your package, your company or even what you want.
Click **Edit** to modify the contents. Leave the field blank to **disable** this dialog box.

Welcome

This will install **Adminer Database Editor** for ExeOutput for PHP.

Description

Adminer is a database editor in order to manage databases of the portable MySQL server that ExeOutput for PHP can include in your applications. Please refer to the documentation for help.

License: Apache or GPL 2.

Requires ExeOutput for PHP 2024.2.

Press **Next** to install or Cancel to exit.

38.2 🎨 Professional Presence

Configure how your application appears to the end-user during and after installation:

1. **Identity:** Set the **Installer Title** and **Application Name** to ensure consistent branding throughout the setup process.
2. **Destination:** Choose the default installation path (e.g., `My Documents` or `Program Files`) to ensure the user knows exactly where your software resides.
3. **Automation:** Use the **Open project in Paquet Builder** toggle to jump straight from configuration to final compilation.

38.3 The Setup Workflow

1. **Configure:** Input your application's identity and desired installation path.
2. **Generate:** Click **Generate Installer** to create the specialized `.pbpx` project file.
3. **Refine:** Open the project in Paquet Builder to add custom logos, license agreements (EULA), or additional files.
4. **Ship:** Compile your project into a single, digitally-weighted setup `.exe` for your customers.

Download Paquet Builder

38.4 See Also

Distribution

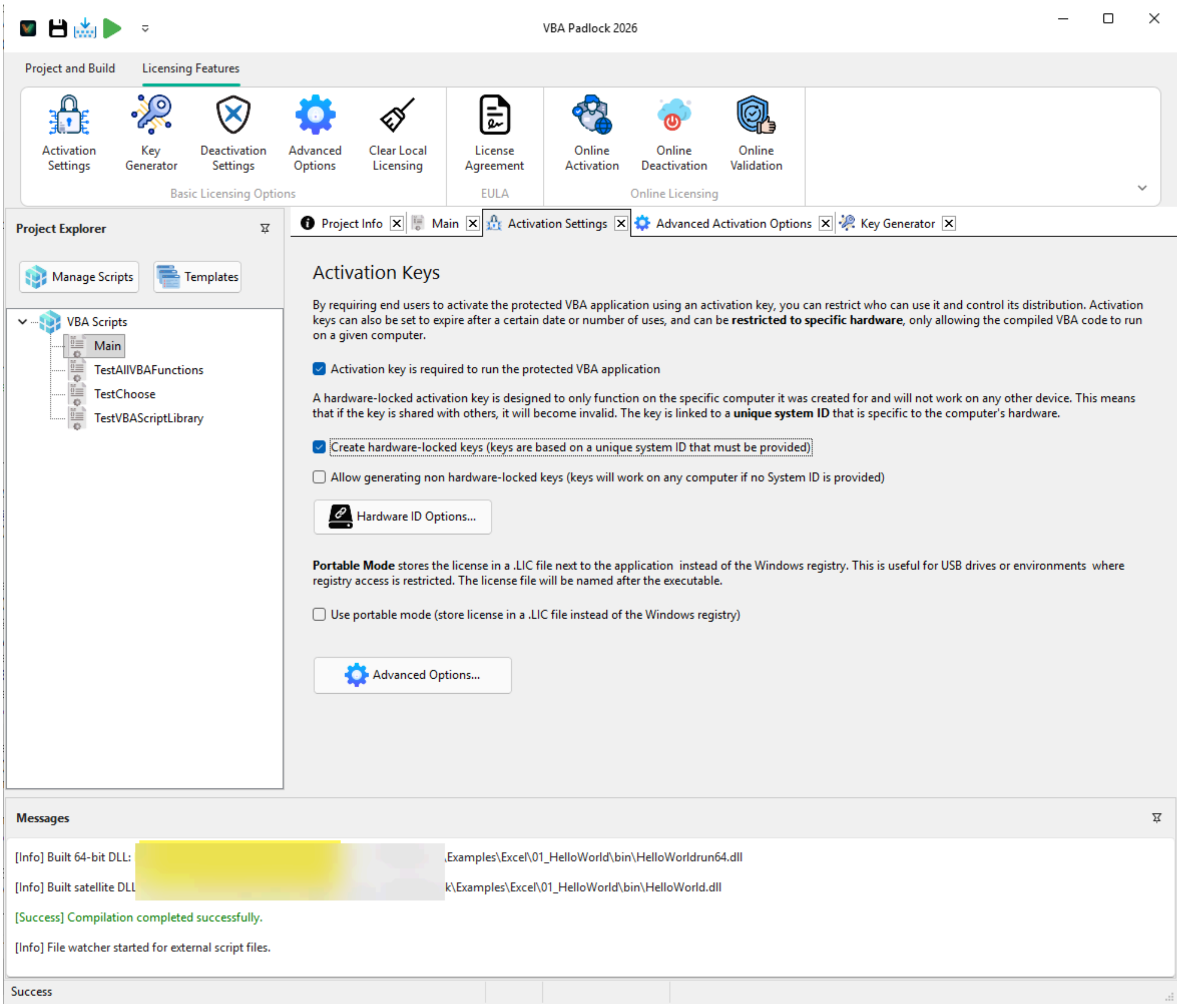
Compare other packaging methods. View Reference →

Publish Final DLL

Prepare your binaries for the installer. View Reference →

39. Licensing Features

The **Licensing Features** tab is where you define the business logic of your application. From simple trial modes to robust hardware locking and automated online activation, this section covers all aspects of protecting your intellectual property and managing your customers.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

☑ Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

☑ Create hardware-locked keys (keys are based on a unique system ID that must be provided)

☐ Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Hardware ID Options...

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

☐ Use portable mode (store license in a .LIC file instead of the Windows registry)

Advanced Options...

Messages

```
[Info] Built 64-bit DLL: [redacted]Examples\Excel\01_HelloWorld\bin\HelloWorldrun64.dll
[Info] Built satellite DLL k\Examples\Excel\01_HelloWorld\bin\HelloWorld.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

Success

39.1 Protection Strategy

A typical licensing setup involves choosing the right balance between security and user convenience.

1. **Basic Rules:** Enable Activation Settings and choose between silent execution or mandatory activation.
2. **Hardware Lock:** Decide which Hardware ID Components will tie the license to a specific PC.
3. **Governance:** Configure your EULA and Deactivation policies.

4 **Automation:** (Optional) Set up Online Activation to fulfill orders 24/7.

39.2 🛠️ Configuration Groups

Traditional Licensing

Core settings for offline activation and trial management.

- Activation Configuration
- Trial & Nag Screens
- Advanced Options

Identity & Hardware

Ensure your licenses aren't shared across multiple machines.

- Hardware ID Definition
- Hardware Locking Guide
- User Name Binding

Online Automation

Modern license management via a central PHP server.

- Online Activation
- Remote Validation
- Cloud Deactivation

Compliance & Keys

Manage the legal and technical aspects of license issuance.

- EULA Configuration
- Key Generator Tool
- Key Distribution

39.3 See Also

Project & Build

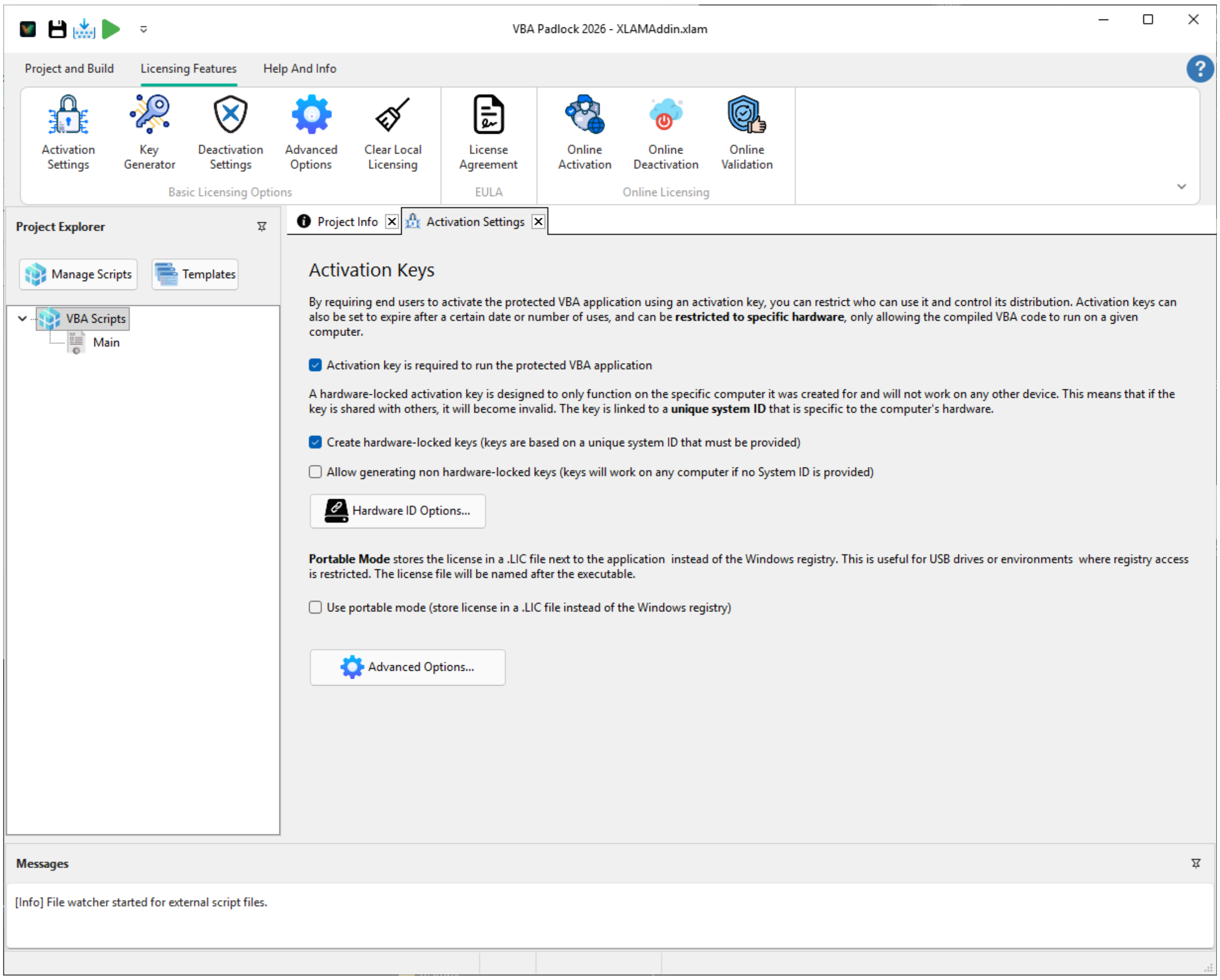
Review the source code and compilation settings. View Build Reference →

Activation Server

Learn how to deploy your own activation portal. Read Server Guide →

40. Activation Settings

The **Activation Settings** dialog is the central command center for your protection system. This is where you decide if your application requires a license to run and how that license is secured. For a comprehensive overview of licensing concepts, see the Licensing Feature guide.



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

- [✓] Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

- [✓] Create hardware-locked keys (keys are based on a unique system ID that must be provided)
- [ ] Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Hardware ID Options...

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

- [ ] Use portable mode (store license in a .LIC file instead of the Windows registry)

Advanced Options...

Messages

[Info] File watcher started for external script files.

40.1 🔑 The Master Switch

- **Activation key is required:** This is the most important option. When checked, your software will be locked until the user enters a valid license key.

ⓘ Note

If this box is unchecked, no activation window will be displayed. Your application will then run in “silent” or “unlicensed” mode (usable for free trials without restrictions).

40.2 🛡️ Hardware Locking Strategy

Hardware Locking allows you to bind a license to a specific computer to prevent key sharing. Learn more about this powerful anti-piracy technique in the Hardware Locking Feature guide.

- **Create hardware-locked keys:** When this option is active, a key generated for User "A" will never work on User "B's" computer. The customer must provide their unique **System ID**.
- **Allow generating non hardware-locked keys:** Useful if you want to sell "site licenses" or universal keys that work on any PC.
- **Hardware ID Options:** Click this button to choose which hardware components (CPU, Hard Drive, BIOS) will be used to identify the computer.

40.3 📂 Storage & Portability

By default, VBA Padlock stores the license in the Windows Registry.

- **Use portable mode:** By checking this option, the license is saved in a `.LIC` file located right next to your Office document.

> **Tip**
>
> Portable mode is ideal for applications distributed on **USB drives** or for users who do not have administrator rights on their machine.

40.4 ⚙️ Advanced Options

This is where you can customize the appearance of your customers' dialog boxes (logos, colors) or configure your online activation servers.

View Advanced Options →

40.5 See Also

Generate Keys

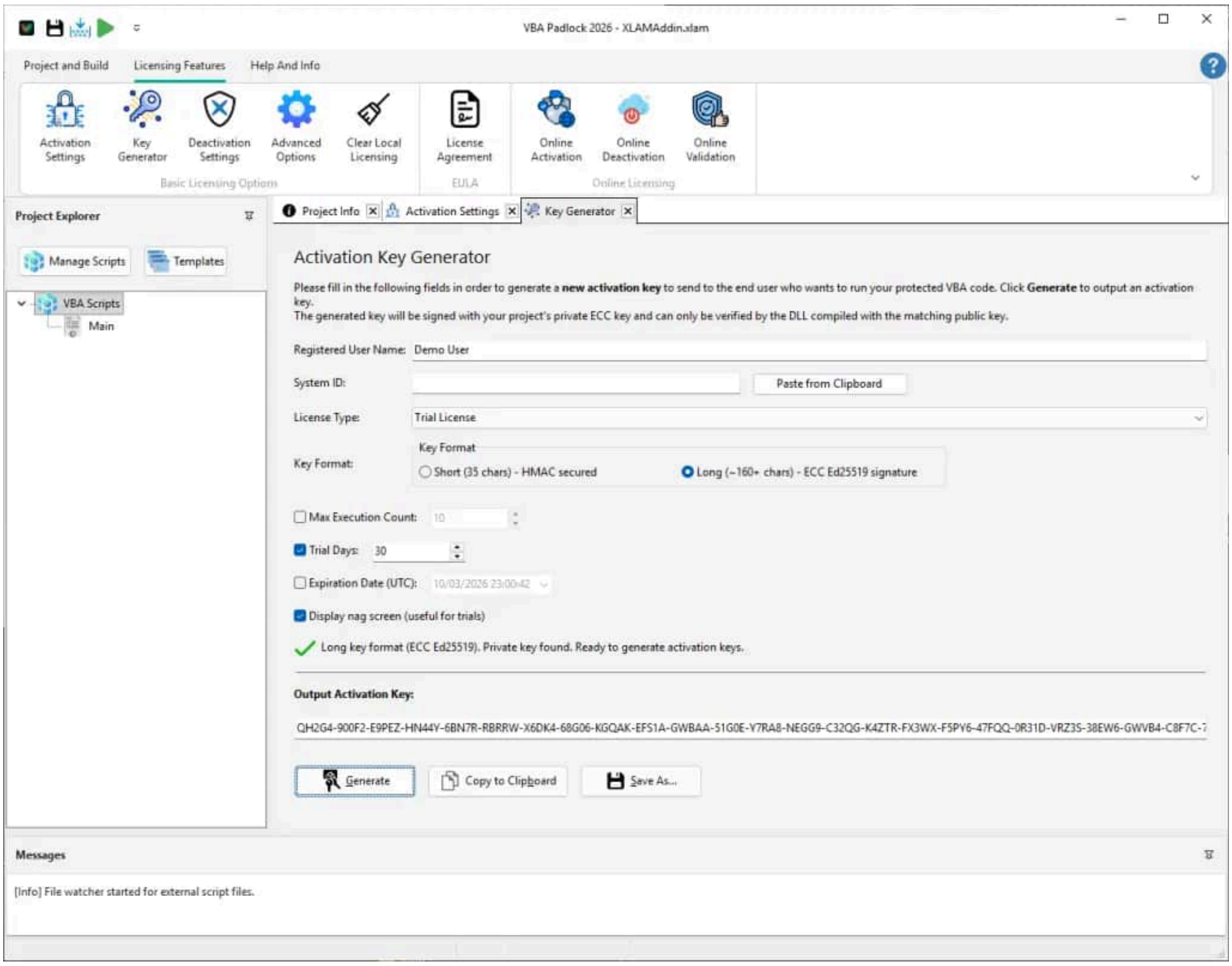
Create licenses for your customers. View Help →

Deactivation

Allow license transfers. View Help →

41. Key Generator

The **Key Generator** is the engine of your licensing business. It transforms project settings and customer identity into a secure, encrypted string—the Activation Key—that unlocks your application on the end-user's machine. Learn more about licensing strategies in the Licensing Feature guide.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format: Key Format — Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

[Info] File watcher started for external script files.

41.1 👤 Granting Access

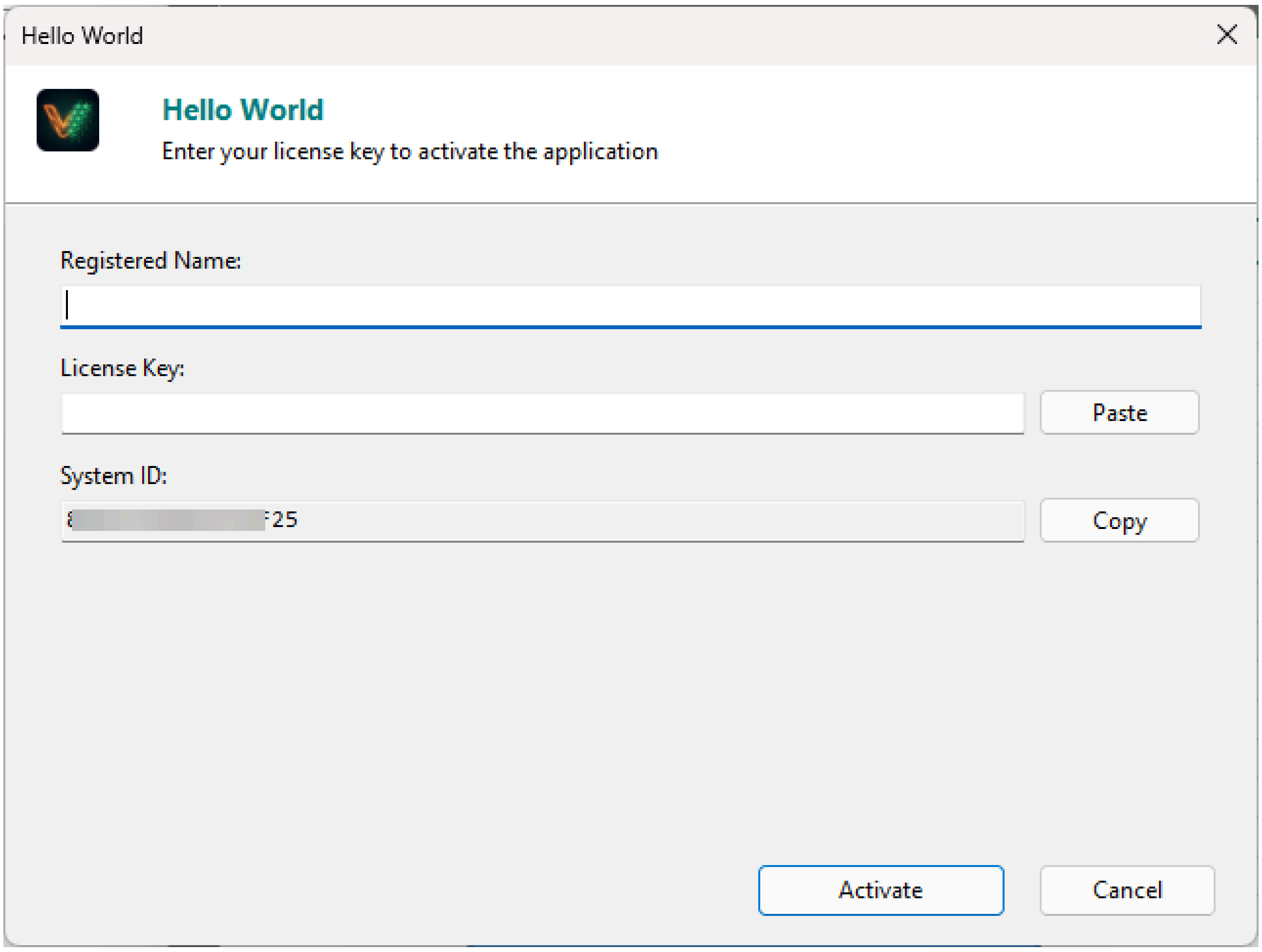
Every license key is bound to a registered identity, ensuring a professional "owned-by" experience for your customers.

- **Registered User Name:** The name of the licensee. This is embedded in the key and will be displayed in the application's "About" or "Activation" screens. You can also retrieve this name at runtime from your VBA code using `VBAPL_GetRegisteredName()` to personalize the user experience — for example, displaying the licensee name in a cell or on a slide:

```
' Display the licensee name in cell A1
Sub ShowLicensee()
    Dim name As String
    name = VBAPL_GetRegisteredName()
    If name <> "" Then
        Sheet1.Range("A1").Value = "Licensed to: " & name
    End If
End Sub
```

```
' Display the licensee name on the first slide
Sub ShowLicensee()
    Dim name As String
    name = VBAPL_GetRegisteredName()
    If name <> "" Then

ActivePresentation.Slides(1).Shapes("LicenseLabel").TextFrame.TextRange.Text
= "Licensed to: " & name
    End If
End Sub
```

- **System ID (Hardware Lock):** To prevent software piracy, paste the unique **Hardware ID** provided by your customer here. The resulting key will only function on that specific machine. Your end-users will find their System ID in the activation dialog that appears when they first open your protected application (see screenshot below). They need to copy this ID and send it to you so you can generate a hardware-locked key.


Hello World

Hello World

Enter your license key to activate the application

Registered Name:

License Key:

System ID: 8[redacted]F25

41.2 🔐 Formats & Security

VBA Padlock Studio supports two distinct cryptographic formats for your keys. Choose the one that matches your distribution strategy:

✓ Short Keys (HMAC)

35 Characters. Lightweight and easy for users to type. Ideal for standard activation, expiration dates, and basic hardware locking.

🚀 Long Keys (ECC)

160+ Characters. Ed25519 digital signatures for strong cryptographic security. Required for advanced features like **Max Execution Counts** and **Nag Screens**.

⚠ Caution

ECC Requirement: To generate Long keys, you must first create your private/public key pair in the Advanced Options workspace.

41.3 ⏳ License Lifecycle

Define the boundaries of your customer's access using these powerful limiters:

- **Trial Period:** Set a sliding window of usage (e.g., 30 days from the first launch).
- **Hard Expiration:** Set an absolute calendar date after which the software will cease to function.
- **Execution Count:** (Long Keys only) Limit the total number of times the application can be opened.
- **Nag Screens:** (Long Keys only) Display a custom reminder of trial status to encourage conversion.

Tip

When you set the **License Type** to Trial, a nag screen is automatically displayed to the end-user at application startup, reminding them of the remaining trial period and encouraging them to purchase a full license.

41.4 🔄 The Issuance Workflow

1. **Identify:** Enter the **Registered Name** and, if required, the **System ID** from your customer.
2. **Strategize:** Choose between **Short** (simplicity) or **Long** (advanced features) key formats.
3. **Define:** Select the **License Type** (Full, Trial, or Subscription) and apply any time or run limits.
4. **Generate:** Click **Generate** to create the encrypted activation string.
5. **Deliver:** Copy the key to your clipboard or save it to a text file to send to your customer.

41.5 See Also

Activation Settings

Configure the master lock. View Reference →

Hardware ID

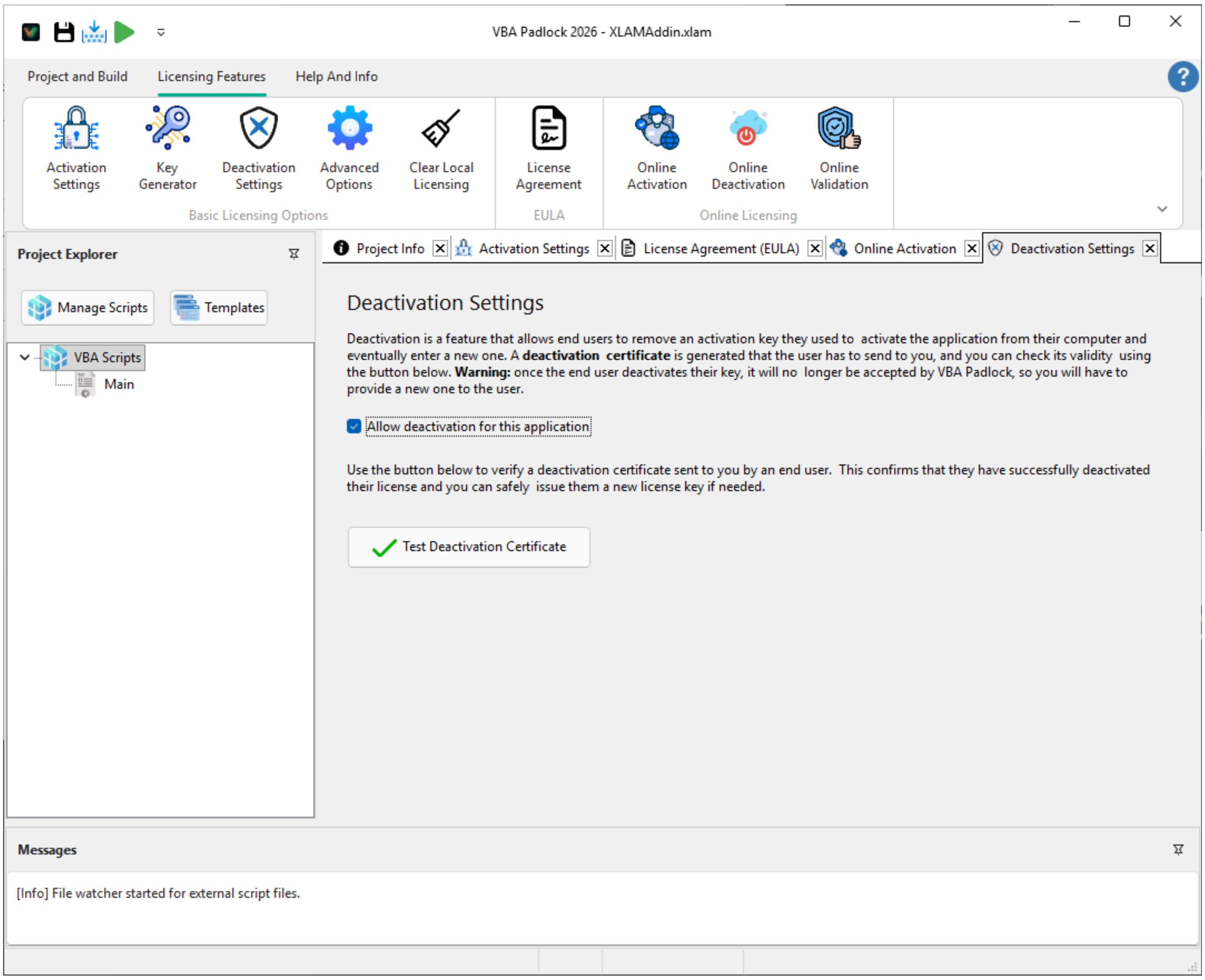
Identify specific computers. View Reference →

Key Generation Guide

Step-by-step tutorial for generating and distributing keys. Read the guide →

42. Deactivation Settings

The **Deactivation Settings** workspace is about flexibility and trust. By allowing users to deactivate their software, you enable a seamless license transfer experience when they upgrade their computers, significantly reducing your manual support burden. Learn more in the Deactivation Feature guide.



Deactivation Settings

Deactivation is a feature that allows end users to remove an activation key they used to activate the application from their computer and eventually enter a new one. A **deactivation certificate** is generated that the user has to send to you, and you can check its validity using the button below. **Warning:** once the end user deactivates their key, it will no longer be accepted by VBA Padlock, so you will have to provide a new one to the user.

☑ Allow deactivation for this application

Use the button below to verify a deactivation certificate sent to you by an end user. This confirms that they have successfully deactivated their license and you can safely issue them a new license key if needed.

Test Deactivation Certificate

[Info] File watcher started for external script files.

42.1 License Mobility

Modern users expect to move their software as they upgrade hardware. Enabling deactivation turns a rigid license into a mobile asset:

- **Self-Service Transfers:** When enabled, users can trigger a deactivation through your UI. This wipes the license from their local machine and generates a cryptographically signed **Deactivation Certificate**.
- **Proof of Removal:** The certificate serves as definitive proof that the software is no longer usable on the previous machine, allowing you to confidently issue a new key for their new hardware.
- **Security Integrity:** Once a license is deactivated, it is permanently neutralized on that specific computer.

Best Practice

Providing a clear “Deactivate License” button in your application’s “Help” or “About” menu is the most professional way to handle machine migrations and upgrades.

42.2 🔗 The Trust Chain

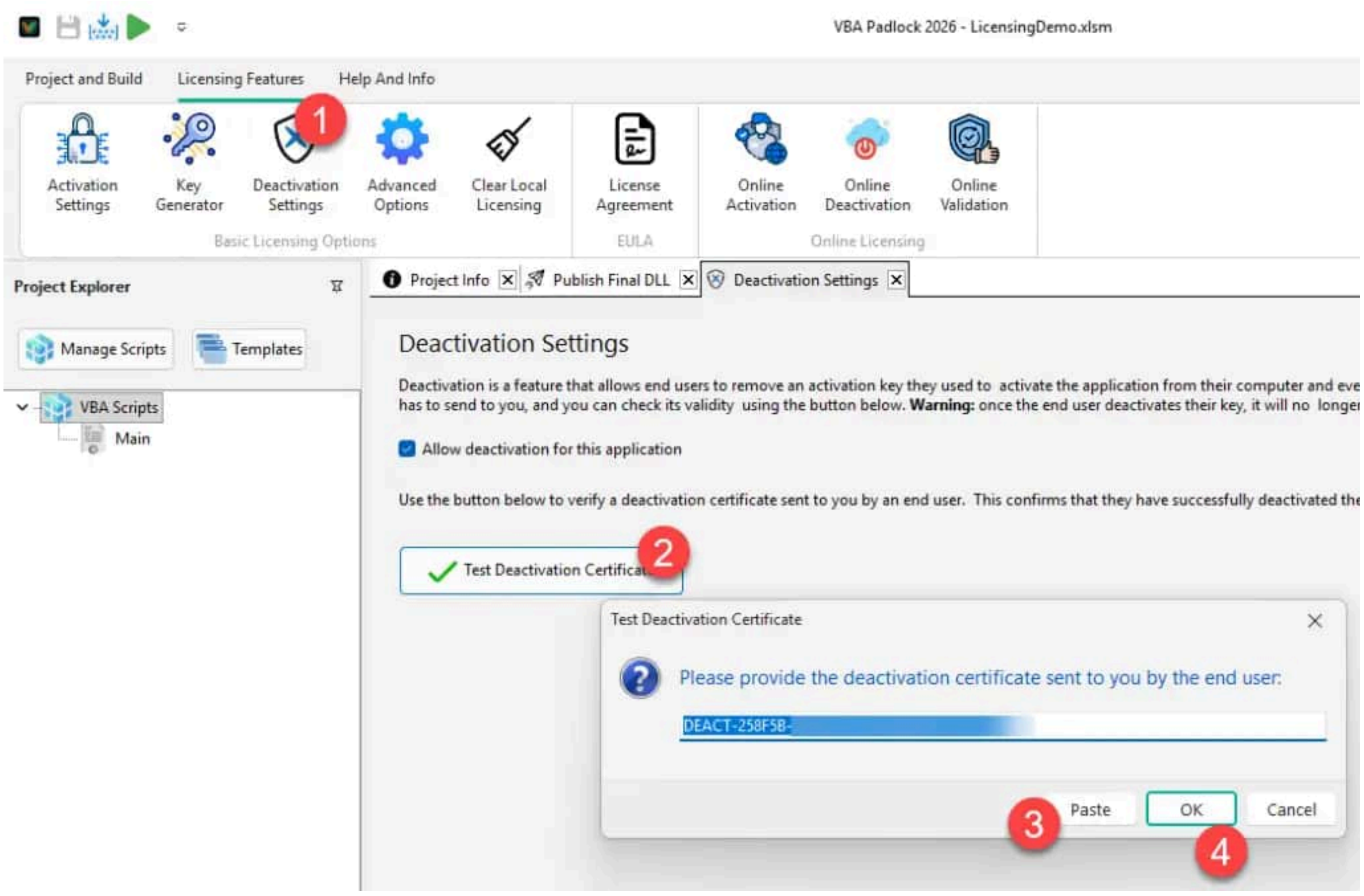
The deactivation process follows a secure, step-by-step lifecycle:

1. **Request:** The user initiates deactivation via the API (e.g., `VBAPL_ShowDeactivation`).
2. **Neutralize:** VBA Padlock scrubs the activation data from the registry or local license file.
3. **Certify:** The application generates a unique, signed string (the certificate).
4. **Verify:** You receive this string and use the **Test Deactivation Certificate** tool to confirm its authenticity.
5. **Re-Issue:** Once validated, you generate a new key for the user’s new Hardware ID.

42.3 🛠️ How to Test a Deactivation Certificate

VBA Padlock Studio includes a built-in validator to ensure certificates haven’t been tampered with.

- **Integrity Check:** Paste any certificate into the **Test Deactivation Certificate** tool.
- **Immediate Feedback:** The Studio will confirm whether the string is a valid, authentic proof of deactivation for your specific project.



Deactivation Settings

Deactivation is a feature that allows end users to remove an activation key they used to activate the application from their computer and eve
has to send to you, and you can check its validity using the button below. **Warning:** once the end user deactivates their key, it will no longer

Allow deactivation for this application

Use the button below to verify a deactivation certificate sent to you by an end user. This confirms that they have successfully deactivated the

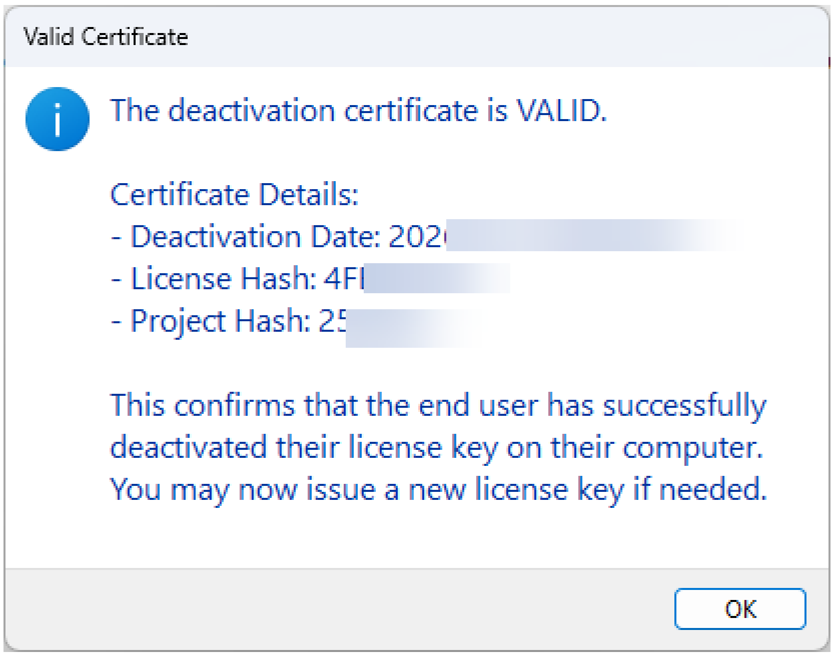
Test Deactivation Certificat

Test Deactivation Certificate

Please provide the deactivation certificate sent to you by the end user:

DEACT-258F5B-

When the certificate is valid, VBA Padlock Studio displays a confirmation message with the details of the deactivated license:


Valid Certificate

The deactivation certificate is VALID.

Certificate Details:
- Deactivation Date: 202
- License Hash: 4F
- Project Hash: 25

This confirms that the end user has successfully deactivated their license key on their computer. You may now issue a new license key if needed.

OK

42.4 🌐 Going Further with Online Deactivation

The manual certificate workflow described above works well for small-scale operations, but as your customer base grows, reviewing certificates by hand can become a bottleneck. Consider setting up Online Deactivation to automate the entire process:

- **24/7 Self-Service:** Your customers can deactivate their license at any time — no need to wait for you to be available.
- **Automatic Server Notification:** The deactivation is reported to your activation server in real time, so your license database is always up to date.
- **Instant Re-Activation:** Combined with Online Activation, users can deactivate on their old machine and re-activate on a new one without any manual intervention from you.

> **Tip**
>
> If you sell more than a handful of licenses, Online Deactivation is strongly recommended. It eliminates the back-and-forth of certificate emails and lets you focus on building your product instead of managing licenses.

42.5 See Also

Key Generator

Issue replacement keys. View Reference →

VBA Bridge API

Trigger deactivation via code. View Reference →

Online Deactivation

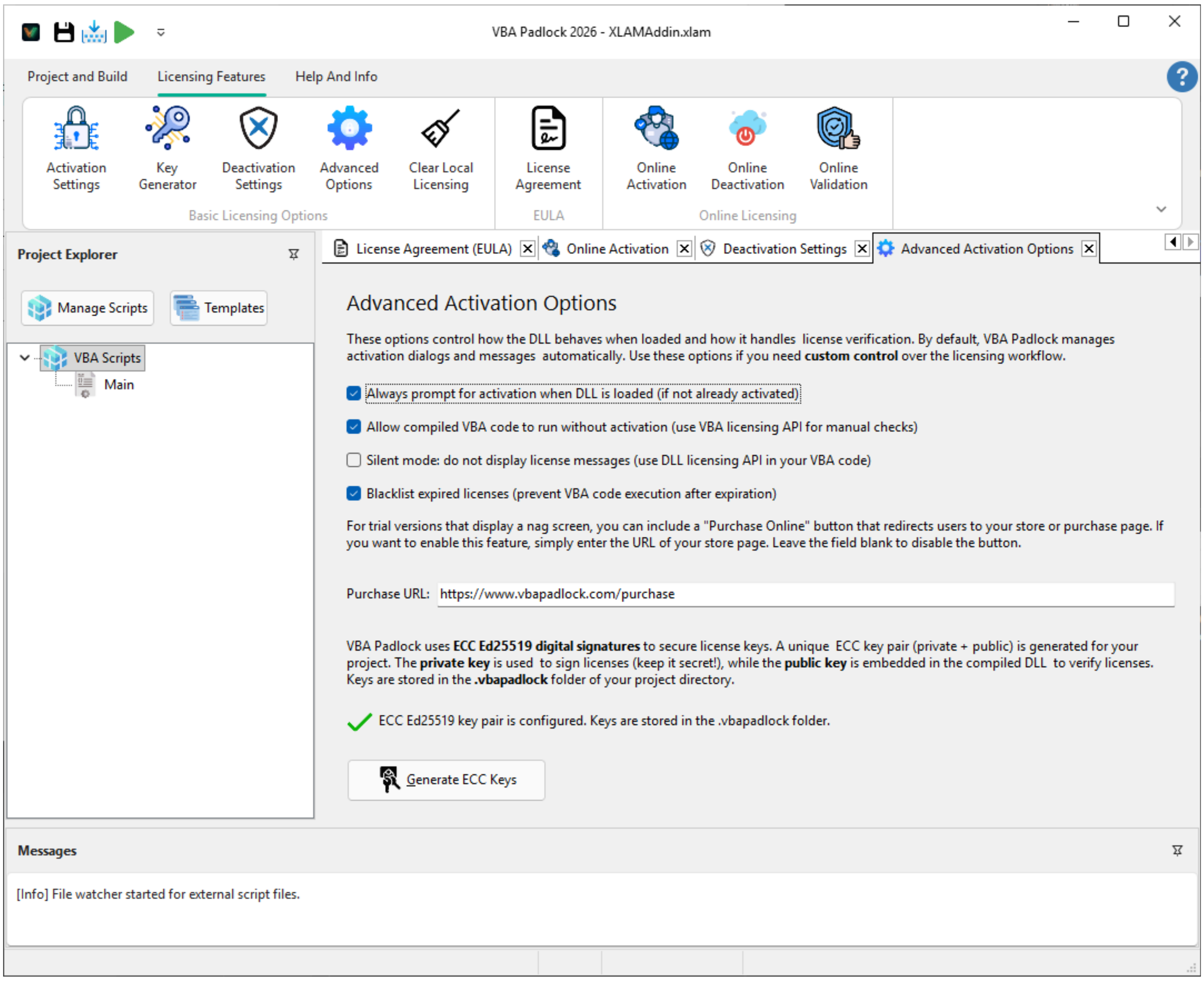
Automate license transfers with your server. View Reference →

Deactivation Feature

Full guide on deactivation workflows. Read the guide →

43. Advanced Activation Options

The **Advanced Activation** workspace is for developers who need specialized control over their software's security handshake. This is where you configure the cryptographic engines that power your activation keys and define how your application behaves when no valid license is present. For an in-depth look at the security architecture, see the Security Feature guide.



Advanced Activation Options

These options control how the DLL behaves when loaded and how it handles license verification. By default, VBA Padlock manages activation dialogs and messages automatically. Use these options if you need **custom control** over the licensing workflow.

- [✓] Always prompt for activation when DLL is loaded (if not already activated)
- [✓] Allow compiled VBA code to run without activation (use VBA licensing API for manual checks)
- [ ] Silent mode: do not display license messages (use DLL licensing API in your VBA code)
- [✓] Blacklist expired licenses (prevent VBA code execution after expiration)

For trial versions that display a nag screen, you can include a "Purchase Online" button that redirects users to your store or purchase page. If you want to enable this feature, simply enter the URL of your store page. Leave the field blank to disable the button.

Purchase URL: https://www.vbapadlock.com/purchase

VBA Padlock uses **ECC Ed25519 digital signatures** to secure license keys. A unique ECC key pair (private + public) is generated for your project. The **private key** is used to sign licenses (keep it secret!), while the **public key** is embedded in the compiled DLL to verify licenses. Keys are stored in the **.vbapadlock** folder of your project directory.

✓ ECC Ed25519 key pair is configured. Keys are stored in the .vbapadlock folder.

Generate ECC Keys

[Info] File watcher started for external script files.

43.1 🎭 Runtime Orchestration

Control the gravity of your security enforcement. Choose between a mandatory gatekeeper or a silent background monitor:

- **Enforcement Toggles:** Use **Always prompt for activation** to ensure the licensing dialog is the first thing a user sees.
- **Silent Execution:** Ideal for background services or advanced developers who want to build their own activation UI via the API. When checked, the DLL executes silently, and the developer is responsible for

checking the license status via code.
- **Expiration Blacklisting:** A critical safety feature that ensures code execution is immediately terminated the moment a license expires, preventing the one-day-late bypass.

43.2 🔐 Cryptographic Foundations

VBA Padlock Studio leverages industry-standard algorithms to ensure your licenses cannot be forged or tampered with.

Master Key

A secret random value that is the "DNA" of your project's licensing. The Master Key is **required for both Short and Long key formats** — it is the foundation of all license generation. Keep it safe and never change it after release.

Ed25519 ECC Keys

Modern Elliptic Curve Cryptography used **additionally** for **Long (ECC)** keys. Generate a unique key pair to digitally sign your licenses with strong cryptographic security. Only required if you use the Long key format.

⚠ Point of No Return

Warning: Never regenerate your ECC or Master keys once you have distributed software to customers. Changing these keys will instantly invalidate every license key you have ever issued, causing a total service disruption for your users.

43.3 🛒 Conversion & Sales

Drive revenue by making it easy for trial users to upgrade to a full license:

- **Purchase URL:** Define the global destination for your "Buy Now" links. When a user clicks the purchase button in a trial nag screen, they are redirected to this URL (e.g., your Stripe or PayPal checkout page).

43.4 🔄 The Security Workflow

1. **Initialize:** Click **Generate ECC Keys** to establish your cryptographic identity.

2. **Secure:** Confirm your **Master Key** is generated and valid.
3. **Behavior:** Select your enforcement strategy (Standard, Silent, or Permissive).
4. **Connect:** Provide your **Purchase URL** to bridge the gap between trial and sale.

43.5 See Also

Key Generator

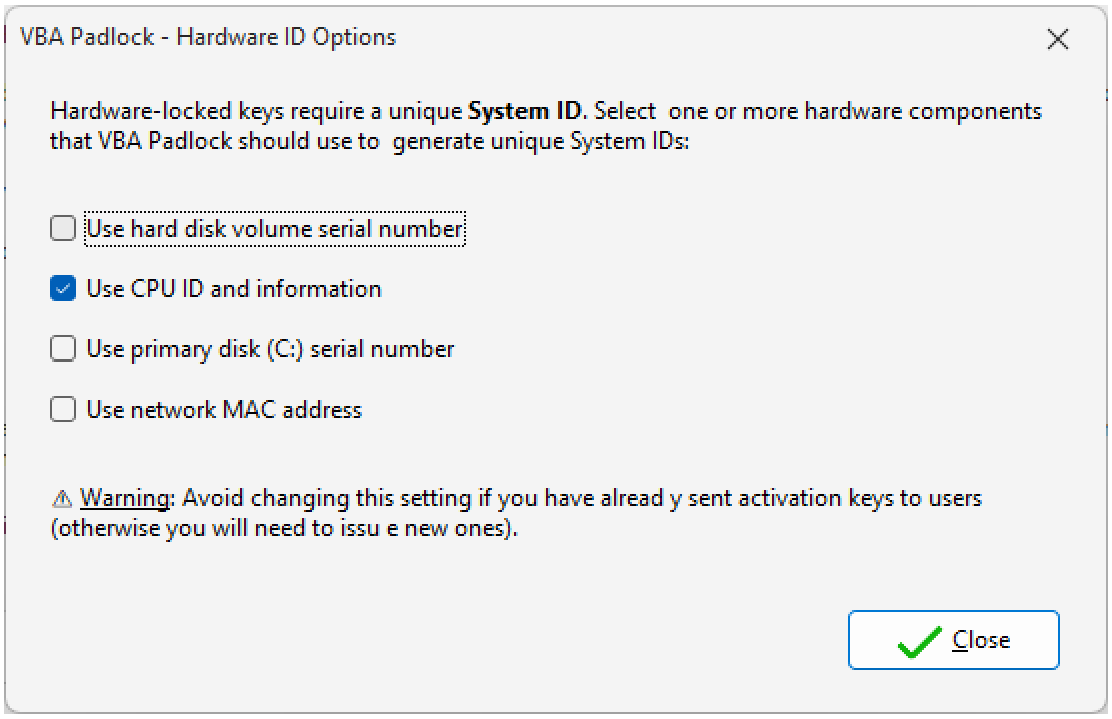
Create keys using these settings. View Reference →

Activation UI

Configure the user-facing dialog. View Reference →

44. Hardware ID Options

The **Hardware ID Options** workspace allows you to define the “DNA” of a user’s computer. By selecting specific physical components, you create a unique **System ID** that prevents the unauthorized sharing of license keys across different machines.


VBA Padlock - Hardware ID Options

Hardware-locked keys require a unique **System ID**. Select one or more hardware components that VBA Padlock should use to generate unique System IDs:

- [ ] Use hard disk volume serial number
- [x] Use CPU ID and information
- [ ] Use primary disk (C:) serial number
- [ ] Use network MAC address

⚠ Warning: Avoid changing this setting if you have alread y sent activation keys to users (otherwise you will need to issu e new ones).

44.1 🧬 Digital Fingerprinting

VBA Padlock Studio can sample several hardware layers to generate a resilient and unique identifier. choosing the right combination is key to balancing security with user convenience:

- **Processor (CPU ID):** High stability. This ID only changes if the user replaces their entire motherboard or CPU.
- **System Drive (Volume Serial):** Standard stability. This is tied to the software volume ID of the primary drive. It is resilient to hardware swaps but changes if the OS is reformatted.
- **Physical Disk Serial:** Maximum security. Tied to the factory-recorded serial number of the hard drive itself.
- **Network (MAC Address):** Lowest stability. While unique, MAC addresses can be volatile due to virtual adapters, VPNs, or network card changes.

⚠ System Integrity

Stability Warning: Do not change these components after you have started issuing keys. Changing the fingerprint formula will change the System ID of every customer machine, instantly invalidating all existing activations.

44.2 ⚖️ Strategy & Recommendations

How much “uniqueness” does your software require?

✓ Standard (Recommended)

CPU ID + Volume Serial. The best balance for commercial software. Stable during most upgrades but unique enough to prevent piracy.

High Security

CPU ID + Physical Disk Serial. Used for high-value enterprise tools where license sharing must be strictly eliminated at the cost of less flexibility.

44.3 🔄 The Fingerprinting Workflow

1. **Strategize:** Decide on your balance between stability and security.
2. **Select:** Check the desired components in the Hardware ID workspace.
3. **Deploy:** Compile your project to bake the new formula into the DLL.
4. **Confirm:** Ask a test user for their System ID and verify it remains stable across a machine reboot.

44.4 See Also

Activation Settings

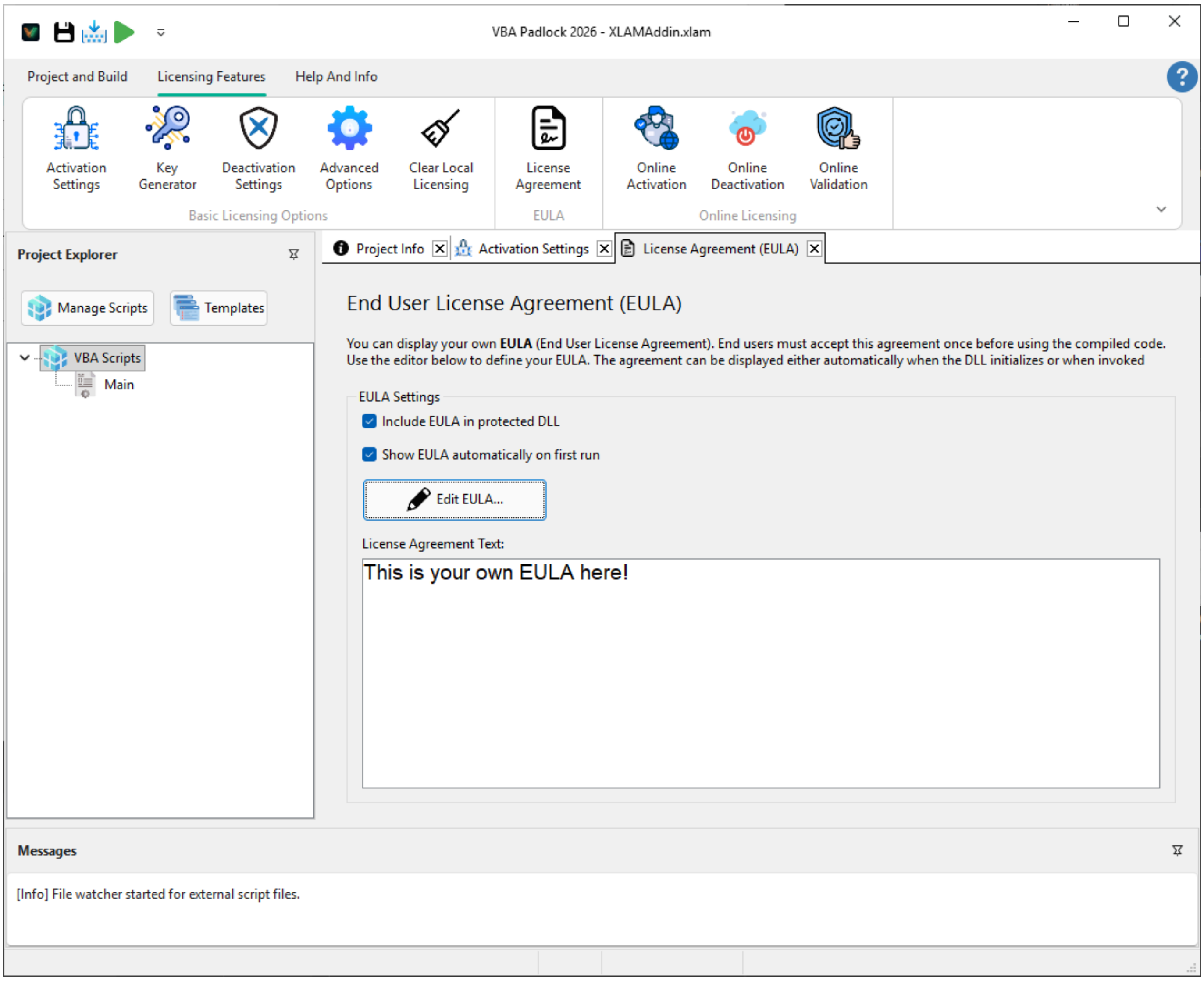
Enable hardware locking. View Reference →

Key Generator

Issue locked keys. View Reference →

45. License Agreement (EULA)

The **License Agreement** workspace (EULA) ensures that your customers are aware of and agree to your terms of service before they ever see your protected code. By embedding the agreement directly into the DLL, you create a tamper-proof legal gatekeeper for your application. Learn more in the EULA Feature guide.



End User License Agreement (EULA)

You can display your own **EULA** (End User License Agreement). End users must accept this agreement once before using the compiled code. Use the editor below to define your EULA. The agreement can be displayed either automatically when the DLL initializes or when invoked

EULA Settings

- [✓] Include EULA in protected DLL
- [✓] Show EULA automatically on first run

License Agreement Text:

This is your own EULA here!

[Info] File watcher started for external script files.

45.1 🏛 Governance & Compliance

Choosing how to present your EULA is critical for both legal validity and user experience.

- **Secure Embedding:** Checking **Include EULA in protected DLL** ensures your license text is cryptographically bound to your software, making it impossible to replace with a modified version.
- **Automated Enforcement:** Enable the **Show EULA automatically on first run** option to force a popup dialog during the application's initial handshake. The user must click "Accept" to proceed; otherwise, the application will terminate.

- **Rich Text Support:** Use the built-in editor to format your agreement with bold headers, lists, and clear paragraphs for better readability.

45.2 💻 Programmatic Control (API)

For developers who need more granular control, the VBA Bridge provides a suite of functions to manage the EULA state directly from your code.

```
' Check if the user has already accepted your terms
If VBAPL_IsEulaAccepted() = 0 Then
    ' Force the dialog if you've updated your EULA
    If VBAPL_ShowEULA() = 0 Then
        MsgBox "Agreement declined. Closing application."
        Exit Sub
    End If
End If
```

Tip

Compliance Tracking: Use `VBAPL_GetEulaAcceptanceDate()` to retrieve the exact timestamp when a user accepted the agreement, which can be stored in your logs for support or audit purposes.

45.3 🔄 The Legal Workflow

1. **Draft:** Use the **Edit EULA** tool to write or paste your primary license agreement.
2. **Toggle:** Activate the **Include EULA** checkbox to bake the text into your binaries.
3. **Deploy:** Choose “Automated Enforcement” for the simplest integration or use the API for custom logic.
4. **Validate:** Compile your project and run a Test to verify the dialog appears as expected.

45.4 See Also

Activation Settings

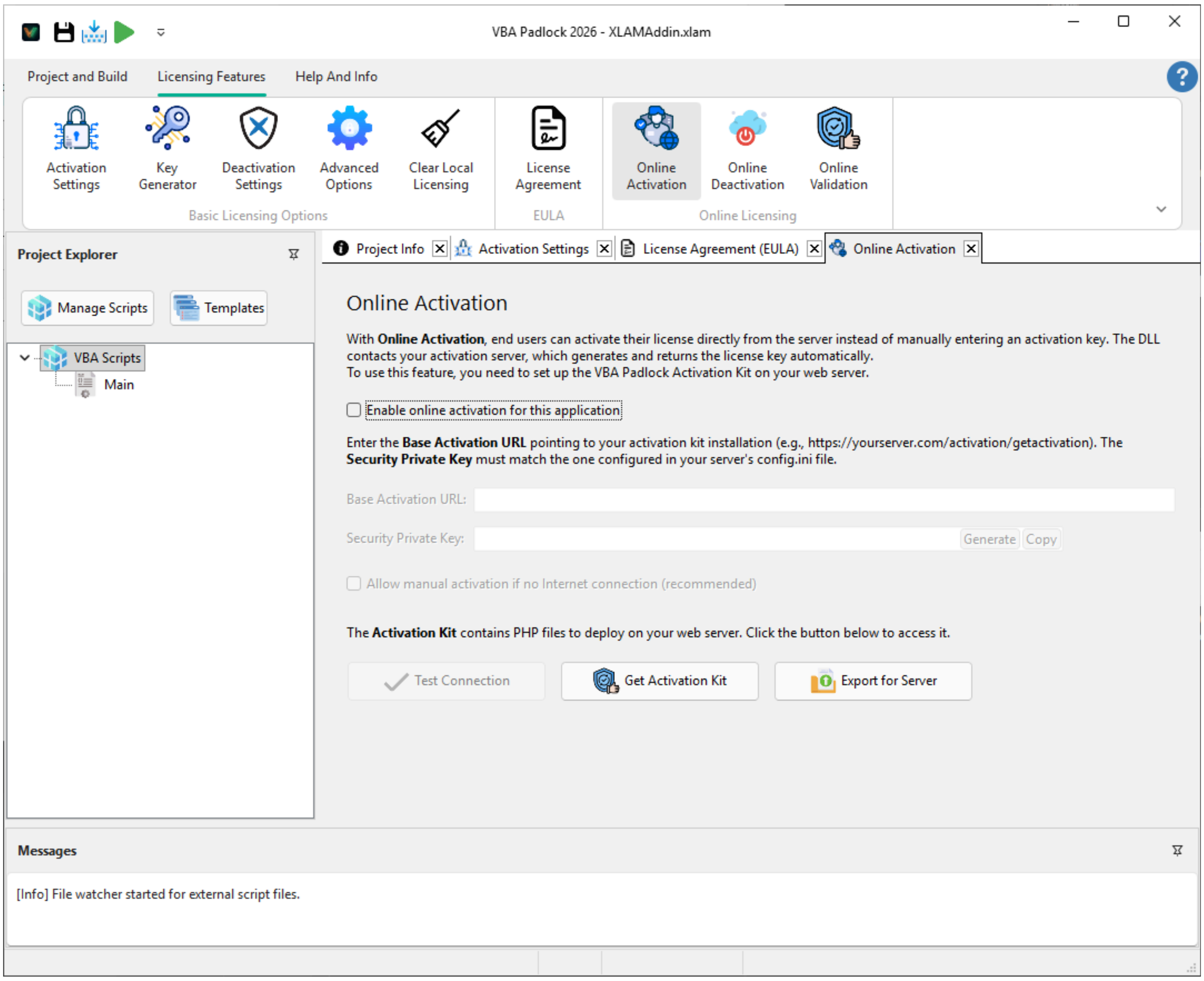
Combine EULA with activation keys. View Reference →

VBA Bridge API

View legal API documentation. View Reference →

46. Online Activation

The **Online Activation** workspace is where you transform your software into a scalable commercial product. Instead of manually generating keys for every customer, your DLL communicates directly with the VBA Padlock Activation Kit — a PHP server you install on your web hosting — to validate credentials and unlock itself in seconds. Learn more in the Online Activation Feature guide.



Online Activation

With **Online Activation**, end users can activate their license directly from the server instead of manually entering an activation key. The DLL contacts your activation server, which generates and returns the license key automatically.
To use this feature, you need to set up the VBA Padlock Activation Kit on your web server.

☐ Enable online activation for this application

Enter the **Base Activation URL** pointing to your activation kit installation (e.g., https://yourserver.com/activation/getactivation). The **Security Private Key** must match the one configured in your server's config.ini file.

Base Activation URL:

Security Private Key: Generate Copy

☐ Allow manual activation if no Internet connection (recommended)

The **Activation Kit** contains PHP files to deploy on your web server. Click the button below to access it.

Test Connection | Get Activation Kit | Export for Server

[Info] File watcher started for external script files.

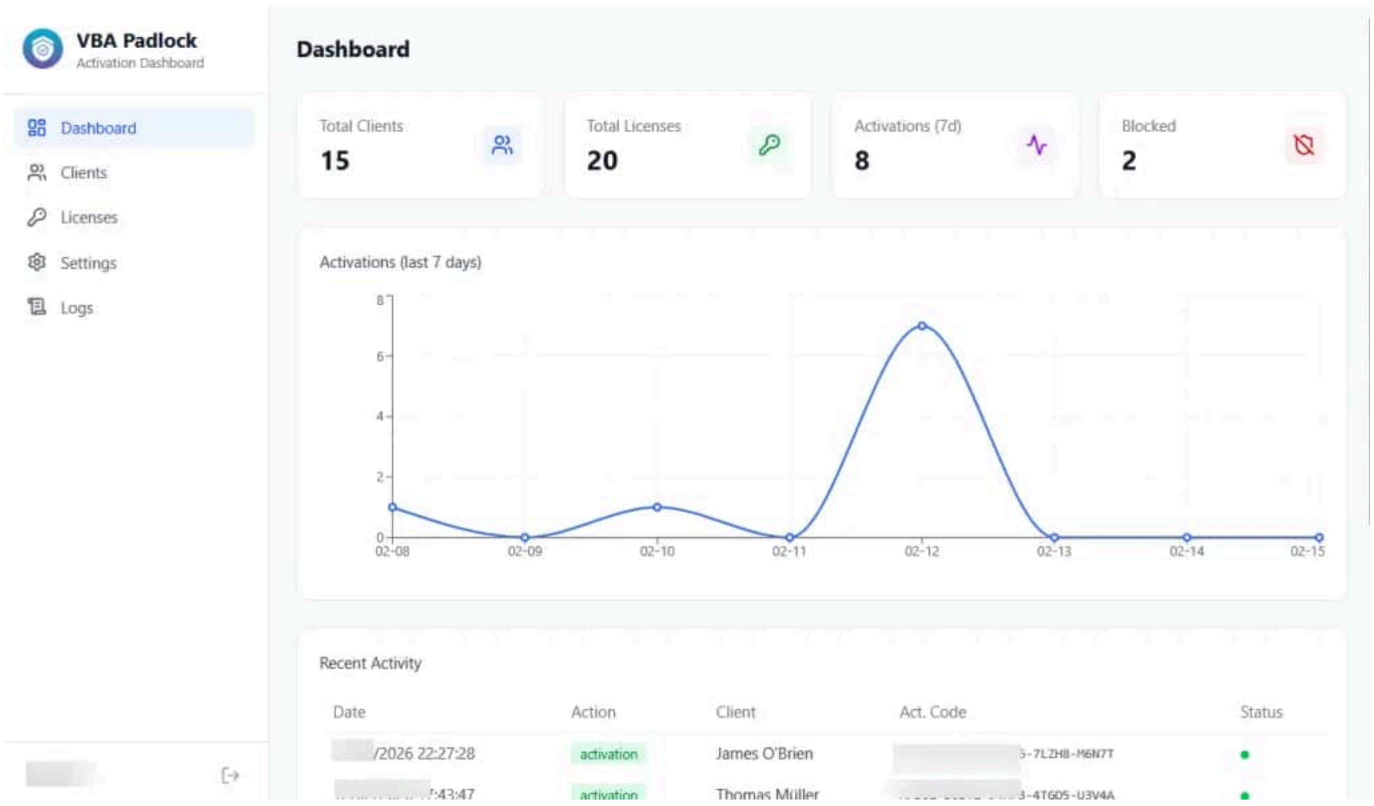
46.1 🚀 Automated Fulfillment

Online activation removes the friction from your sales process. When a user buys your software, they can activate it immediately, regardless of your time zone:

- **Real-time Validation:** The DLL sends the user's Hardware ID and activation code to your server. If valid, the server returns an encrypted key that the DLL installs automatically.
- **Offline Fallback:** Checking **Allow manual activation** ensures that users with restricted firewalls can still activate via email or phone, providing a fail-safe user experience.

- **Scalability:** Whether you have ten users or ten thousand, the PHP server handles the heavy lifting of key issuance and database management.

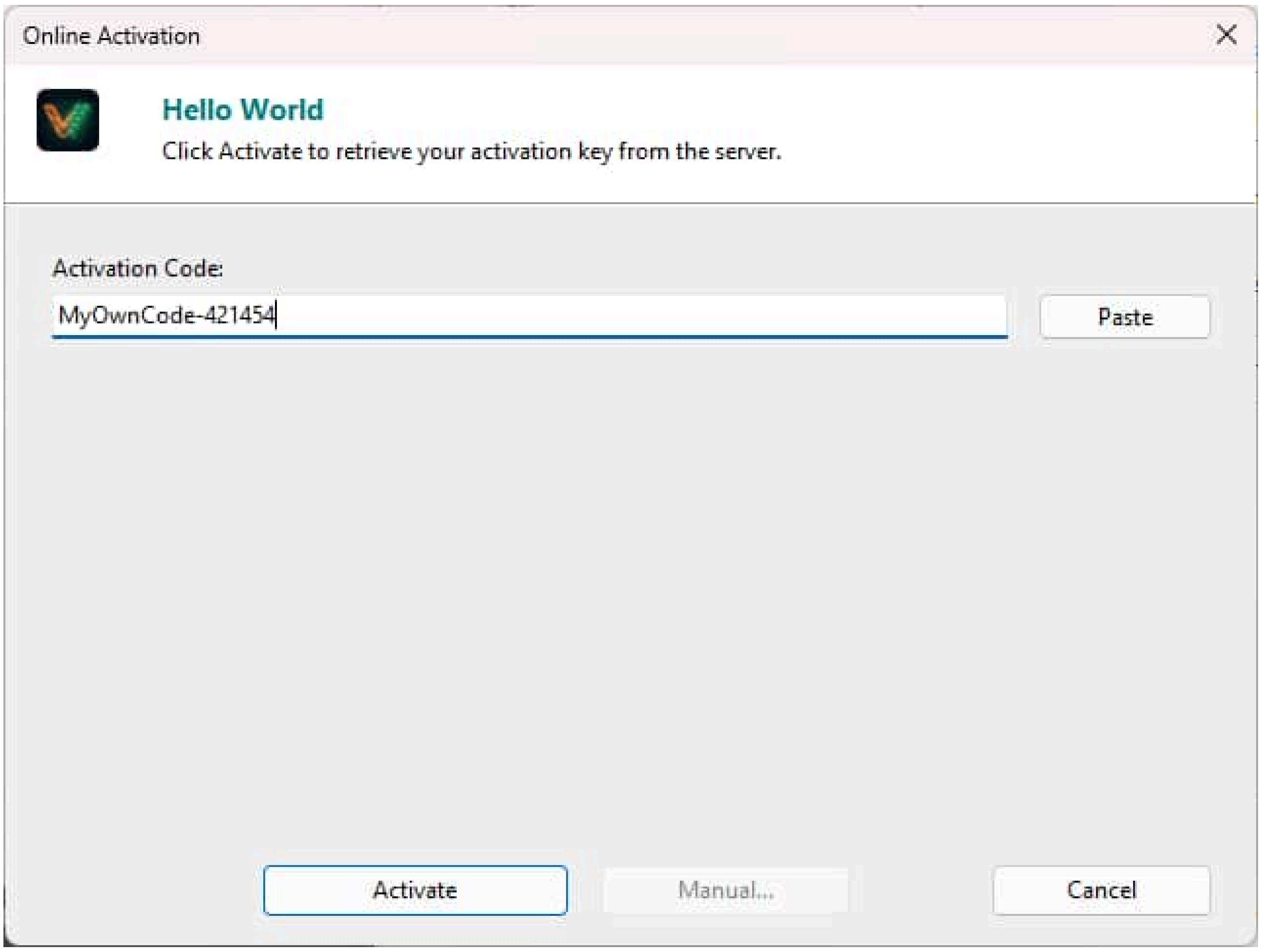
The activation codes are managed from the **Activation Kit Dashboard** on your server. Each license entry contains an **Activation Code** that you provide to your customer after purchase (e.g., via email or your e-commerce confirmation page):



Recent Activity

Date	Action	Client	Act. Code	Status
[redacted]/2026 22:27:28	activation	James O'Brien	[redacted]5-7LZH8-M6N7T	●
[redacted]:43:47	activation	Thomas Müller	[redacted]3-4TGD5-U3V4A	●

Your end-user then enters this activation code in the online activation dialog that appears in your application:


Online Activation

Hello World

Click Activate to retrieve your activation key from the server.

Activation Code:

MyOwnCode-421454

46.2 Security Infrastructure

The trust between your software and your server is anchored by a shared secret and a robust endpoint configuration.

Activation Endpoint

The URL of your hosted PHP script. This acts as the secure gatekeeper for all license requests.

Security Private Key

A unique shared secret used to sign and verify every message between the DLL and your server. This ensures that no third party can spoof an activation response.

Key Consistency

Match Requirement: The **Security Private Key** in your project must exactly match the key configured in your PHP server's `config.php` file. If they differ, all activation attempts will be rejected as insecure.

46.3 📋 Custom Registration

Turn your activation process into a lead generation tool. By enabling the **Custom Registration Form**, you can collect valuable data before the software unlocks:

1. **Lead Capture:** Require users to provide their Name, Email, and Company during activation.
2. **HTML Flexibility:** Use the **Custom Form HTML** editor to design a registration window that matches your brand's aesthetic.
3. **Server Storage:** All collected data is securely transmitted to your PHP server and stored alongside the license record.

46.4 🔄 The Automation Workflow

1. **Configure:** Enable online activation and provide your **Base Activation URL**.
2. **Sync:** Generate your **Security Private Key** and update your PHP server's configuration.
3. **Sync (Optional):** Define your custom registration form if market research or lead capture is required.
4. **Validate:** Use the **Test Connection** tool to ensure the Studio can communicate with your live server.
5. **Deploy:** Compile your project to bake the server connectivity into the DLL.

46.5 See Also

Online Deactivation

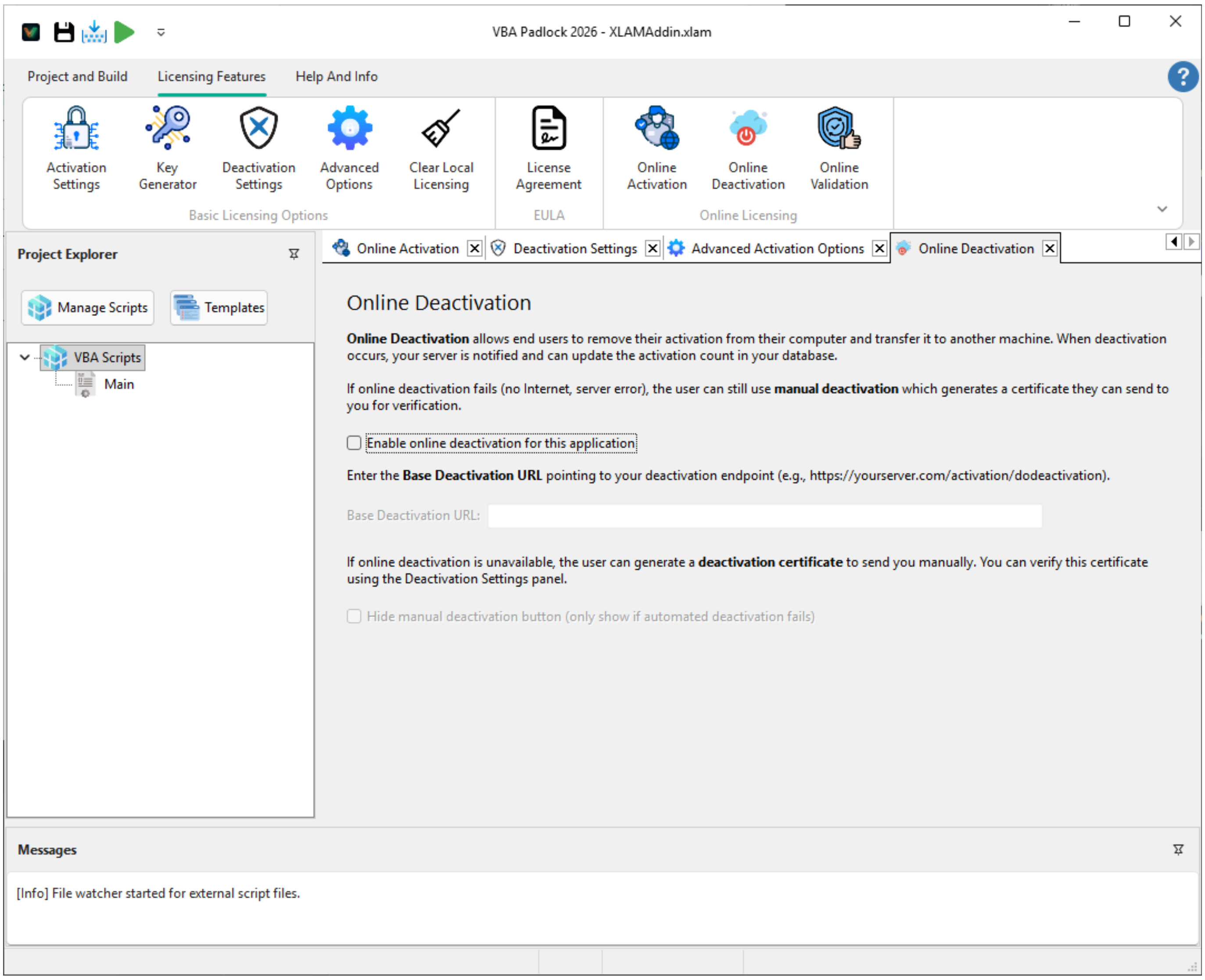
Manage remote license transfers. View Reference →

Online Validation

Ensure licenses stay active. View Reference →

47. Online Deactivation

The **Online Deactivation** workspace closes the loop on automated license management. Instead of manually verifying Deactivation Certificates, your server automatically records the revocation and frees the license slot, allowing the user to reactivate on a new machine immediately.



Online Deactivation

Online Deactivation allows end users to remove their activation from their computer and transfer it to another machine. When deactivation occurs, your server is notified and can update the activation count in your database.

If online deactivation fails (no Internet, server error), the user can still use **manual deactivation** which generates a certificate they can send to you for verification.

☐ Enable online deactivation for this application

Enter the **Base Deactivation URL** pointing to your deactivation endpoint (e.g., https://yourserver.com/activation/dodeactivation).

Base Deactivation URL:

If online deactivation is unavailable, the user can generate a **deactivation certificate** to send you manually. You can verify this certificate using the Deactivation Settings panel.

☐ Hide manual deactivation button (only show if automated deactivation fails)

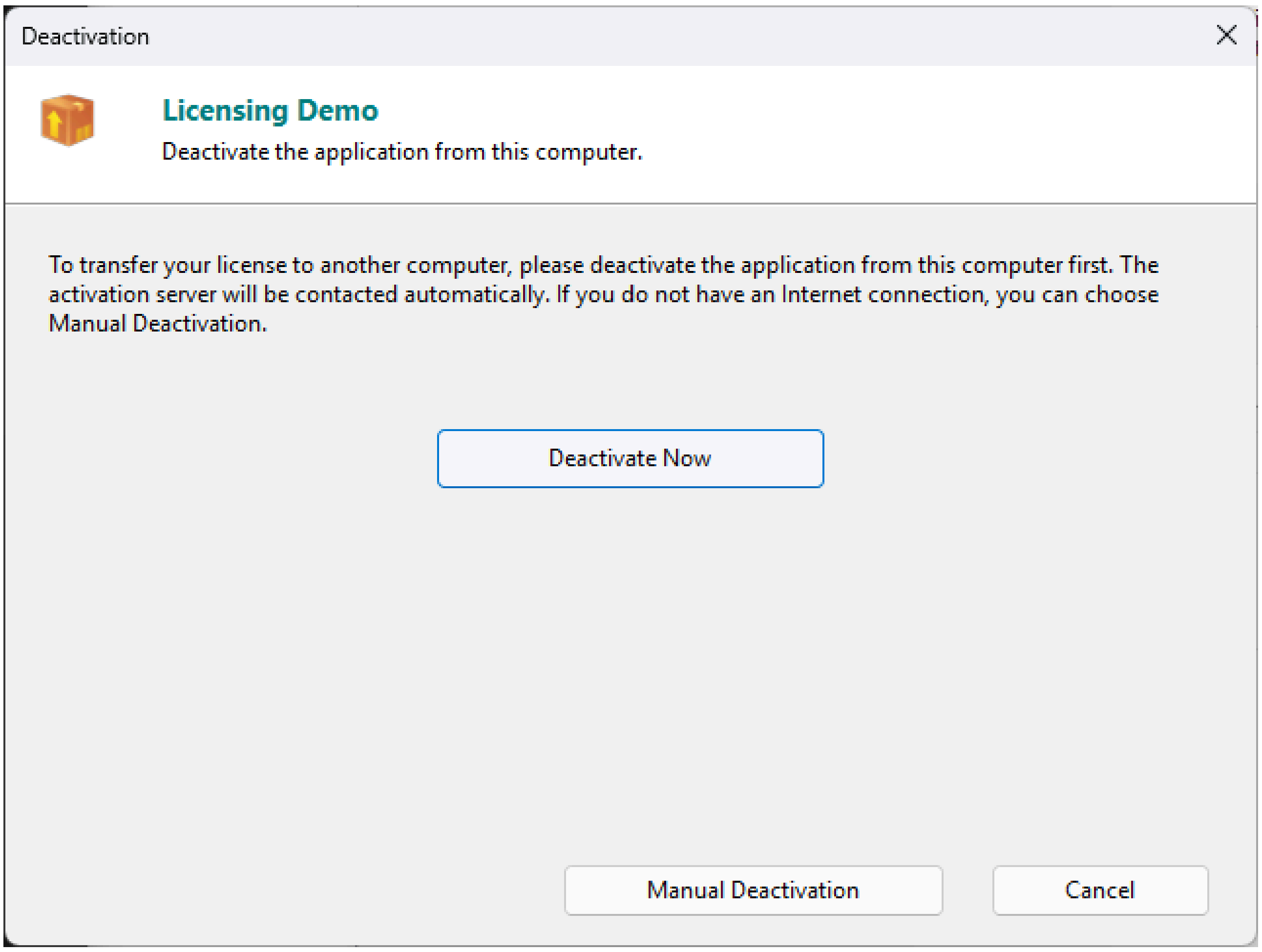
```
[Info] File watcher started for external script files.
```

47.1 🔒 Remote Revocation

Online deactivation transforms a manual, two-way support ticket into a single-click customer experience. When a user deactivates their software:

- **Server Handshake:** The DLL sends a cryptographically signed request to your **Base Deactivation URL**.
- **Asset Recovery:** The PHP server validates the request and marks that specific machine's activation as "Available," effectively returning the license to your pool.
- **Immediate Re-use:** Because the server handles the verification, the user can instantly activate the same key on a different computer without waiting for your approval.

Here is what your end-users see when they trigger an online deactivation from your application:


Deactivation

Licensing Demo

Deactivate the application from this computer.

To transfer your license to another computer, please deactivate the application from this computer first. The activation server will be contacted automatically. If you do not have an Internet connection, you can choose Manual Deactivation.

47.2 🛠️ Smart Fallbacks

Not every machine is online 24/7. VBA Padlock Studio includes intelligent logic to ensure deactivation is never "stuck":

Direct Revocation

The preferred method. If an internet connection is found, the license is revoked and the server is updated in one smooth step.

Certificate Fallback

If the server is unreachable, the system can automatically fall back to generating a Manual Certificate for offline verification.

Tip

Checking **Hide manual deactivation button** ensures a clean UI for your users. The manual option will only appear if the automated server handshake fails, maintaining a high-tech “it just works” aesthetic.

47.3 The Infrastructure Workflow

1. **Enable:** Activate **Online Deactivation** and provide your endpoint URL.
2. **Verify:** Ensure Basic Deactivation is also checked — this is a prerequisite for the feature.
3. **Sync:** Confirm your PHP server’s `/dodeactivation` endpoint is active and connected to your database.
4. **Deploy:** Compile and test a deactivation on a dev machine to watch the server record update in real-time.

47.4 See Also

Online Activation

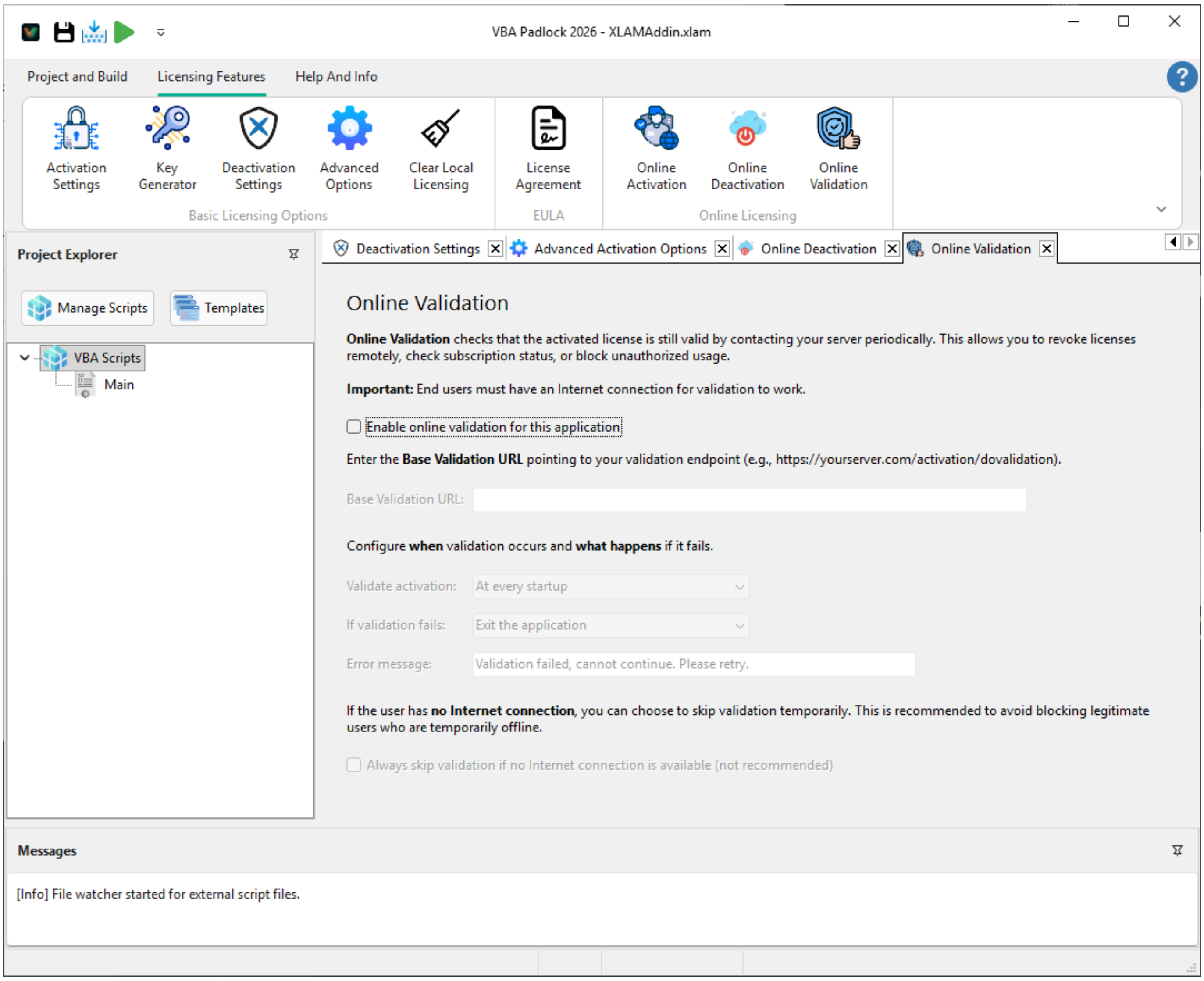
The other half of the server loop. View Reference →

Online Validation

Keep licenses in sync. View Reference →

48. Online Validation

The **Online Validation** workspace ensures that your software remains in sync with your business logic long after the initial activation. By establishing a periodic "heartbeat" with the VBA Padlock Activation Kit installed on your web hosting, you can remotely revoke access, enforce subscription renewals, and neutralize leaked keys without redeploying your code.



Online Validation

Online Validation checks that the activated license is still valid by contacting your server periodically. This allows you to revoke licenses remotely, check subscription status, or block unauthorized usage.

Important: End users must have an Internet connection for validation to work.

☐ Enable online validation for this application

Enter the **Base Validation URL** pointing to your validation endpoint (e.g., https://yourserver.com/activation/dovalidation).

Base Validation URL:

Configure **when** validation occurs and **what happens** if it fails.

Validate activation: At every startup

If validation fails: Exit the application

Error message: Validation failed, cannot continue. Please retry.

If the user has **no Internet connection**, you can choose to skip validation temporarily. This is recommended to avoid blocking legitimate users who are temporarily offline.

☐ Always skip validation if no Internet connection is available (not recommended)

[Info] File watcher started for external script files.

48.1 ❤️ Continuous Compliance

Online validation transforms a static license into a dynamic, server-managed asset. This is the cornerstone of any modern SaaS or subscription-based software model:

- **Remote Revocation:** If a customer cancels their subscription or a chargeback occurs, simply mark the license as "Invalid" on your server. The DLL will detect this during its next validation cycle and terminate access.

- **Tamper Proofing:** Every validation request includes a cryptographic challenge. This prevents "replay attacks" where a user might try to simulate a successful server response while offline.
- **Automatic Handshake:** The process is entirely silent and automatic. The DLL performs the check during its initialization before any protected code is even loaded.

48.2 ⚖️ Frequency & Enforcement

Balance user convenience with security by defining how often the "heartbeat" occurs:

✓ Check Frequency

Choose from **Every Startup**, **Daily**, or **Weekly** checks. For lower-security needs, utilize the **Random** setting to vary the check timing.

Enforcement Mode

Define the consequence of failure. Choose **Block Execution** for strict enforcement or **Show Warning** for a "soft-nag" approach.

⚠ Internet Dependency

Warning: Be wary of the "Skip if no Internet" option. While it prevents user frustration during outages, it also creates an offline bypass. A better strategy is to implement a grace period on your server side before the license is marked as failed.

48.3 💻 Transparent Validation with `VBAPL_IsLicenseValid()`

When **Validate activation at every startup** is enabled in this dialog, calling `VBAPL_IsLicenseValid()` in your VBA code automatically triggers an online validation check against your server — no additional code required. The DLL contacts the server, verifies the license status, and returns the result, all transparently behind a single function call.

This means your standard startup code already handles online validation:

```
Private Sub Workbook_Open()

   On Error GoTo CleanFail

   ' Reset first cell
   Dim ws As Worksheet
   Set ws = ThisWorkbook.Worksheets(1)

   ws.Range("A1").Value = "Hello World Excel Example"



   If VBAPL_IsLicenseValid() Then
       ws.Range("B2").Value = "Licensed To"
       ws.Range("B3").Value = VBAPL_GetRegisteredName()
       ws.Shapes("Button 1").TextFrame.Characters.Text = "Deactivate"
   ElseIf VBAPL_IsTrialMode() Then
       ws.Range("B2").Value = "Trial, please purchase"
       ws.Range("B3").Value = ""
       ws.Shapes("Button 1").TextFrame.Characters.Text = "Activate..."
   Else
       ' Deactivated or no valid license
       ws.Range("B2").Value = "No valid license"
       ws.Range("B3").Value = ""
       ws.Shapes("Button 1").TextFrame.Characters.Text = "Activate..."
   End If

   Exit Sub

CleanFail:
    MsgBox "Initialization Error in Compiled Code: " & Err.Number & " - " &
Err.Description, vbCritical, "Runtime Error"

End Sub
```

Zero Code Change

If you already check `VBAPL_IsLicenseValid()` in your application, enabling **Validate activation at every startup** is all you need. The online check happens behind the scenes — your existing VBA code works as-is, with server-side validation added transparently.

48.4 🔧 Manual Validation at Critical Moments

For finer control, you can also trigger an explicit validation check at specific points in your logic — for example, before starting a high-value export or data processing task:

```
' Force a validation check before starting a high-value process
If VBAPL_IsValidationRequired() <> 0 Then
    If VBAPL_ValidateOnline(1) = 0 Then
        MsgBox "Your subscription has expired. Please renew to continue."
        Exit Sub
    End If
End If
```

48.5 🔄 The Validation Workflow

1. **Configure:** Enable online validation and provide your **Base Validation URL**.
2. **Strategy:** Set your check frequency (e.g., Daily) and your enforcement mode (Block vs. Warn).
3. **Sync:** Ensure your PHP server's `/dovalidation` endpoint is active and connected to your license database.
4. **Validate:** Compile your project. The DLL will now automatically begin its heartbeat cycle on the next launch.

48.6 See Also

Online Activation

The initial server handshake. View Reference →

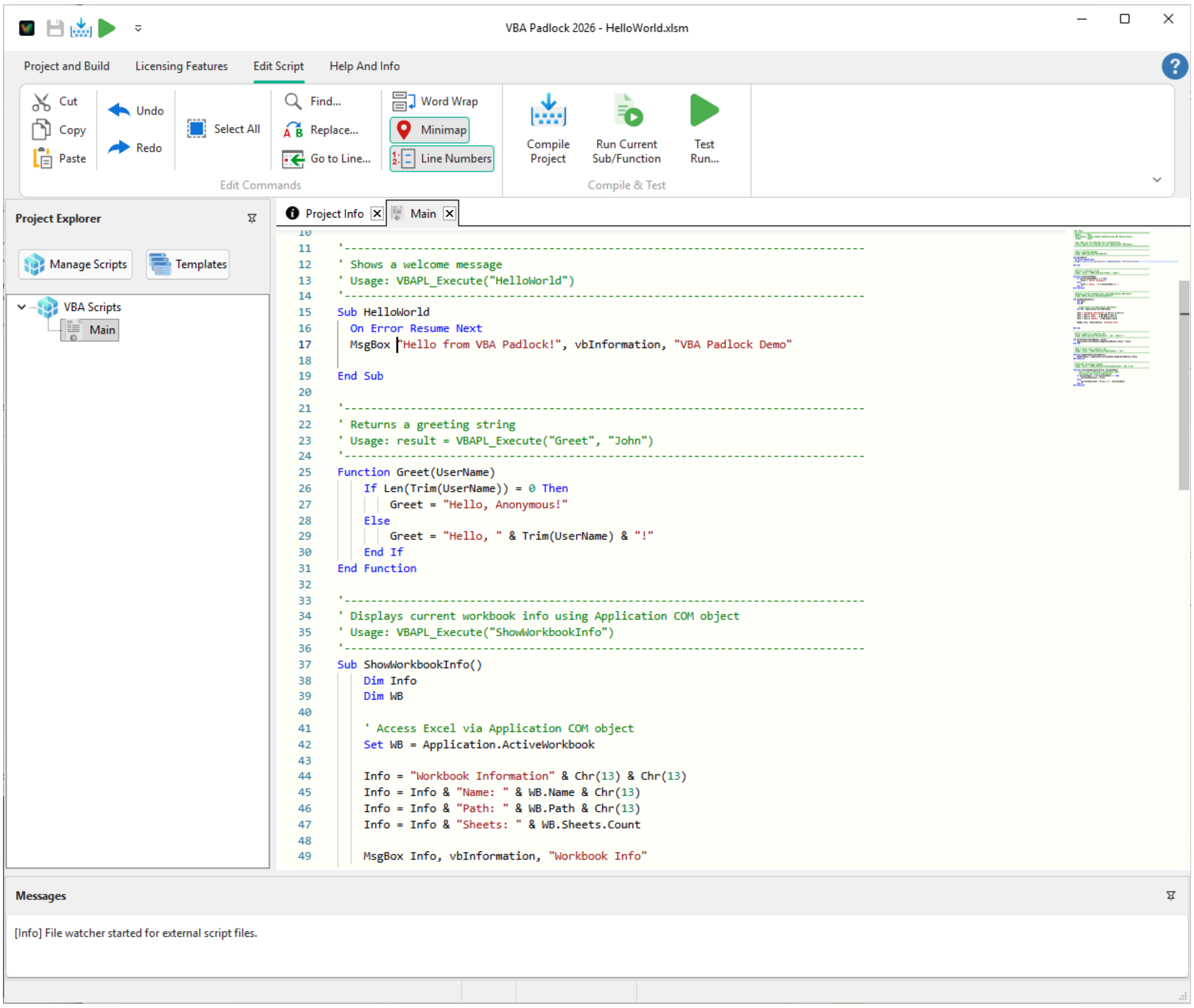
Script Functions

View API for manual validation. View Reference →

49. VBA Compatibility

VBA Padlock compiles VBA (Visual Basic for Applications) into protected bytecode executed inside your secured DLL.

Use this page as your **practical compatibility guide**: what works, what does not, and what to adapt when moving from standard Office VBA to compiled scripts.



```
'---------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'---------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'---------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'---------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'---------------------------------------------------------------------------
' Displays current workbook info using Application COM object
' Usage: VBAPL_Execute("ShowWorkbookInfo")
'---------------------------------------------------------------------------
Sub ShowWorkbookInfo()
    Dim Info
    Dim WB

    ' Access Excel via Application COM object
    Set WB = Application.ActiveWorkbook

    Info = "Workbook Information" & Chr(13) & Chr(13)
    Info = Info & "Name: " & WB.Name & Chr(13)
    Info = Info & "Path: " & WB.Path & Chr(13)
    Info = Info & "Sheets: " & WB.Sheets.Count

    MsgBox Info, vbInformation, "Workbook Info"
```

```
[Info] File watcher started for external script files.
```

49.1 Compatibility at a glance

Area	Status	Notes
Core VBA language (`Dim`, loops, procedures, arrays, `Enum`, `Type`)	✅ Fully supported	Includes dynamic arrays and recursion
Built-in VBA functions	✅ Broad coverage	See Script Functions Reference

Area	Status	Notes
Office object model (`Application`)	✅ Fully supported	Use explicit `Application.` prefix in protected code
COM automation (`CreateObject`)	✅ Supported	`Scripting.Dictionary`, `ADODB.*`, etc.
Windows API `Declare`	⚠️ Opt-in	Disabled by default; enabled on request — see DLL Calls
Class modules / UserForms	❌ Not supported	Use standard modules + Padlock dialogs

Quick migration mindset

If your code is mostly procedural VBA (functions, loops, Office automation), migration is usually straightforward. The biggest required adjustment is to always qualify host objects with `Application.` in protected code.

49.2 Supported VBA features

49.2.1 Language Fundamentals

Feature	Supported	Notes
Variables (`Dim`, `Private`, `Public`)	Yes	All standard data types
Constants (`Const`)	Yes	
Arrays (fixed and dynamic)	Yes	`ReDim`, `ReDim Preserve` supported
User-defined types (`Type`)	Yes	
Enumerations (`Enum`)	Yes	
`Option Explicit`	Yes	Recommended
`Option Compare`	Yes	`Binary` and `Text`
Comments (`'` and `Rem`)	Yes	

49.2.2 Data Types

Type	Supported	Notes
`Boolean`	Yes	
`Byte`	Yes	
`Integer`	Yes	
`Long`	Yes	
`LongLong`	Yes	64-bit integer

Type	Supported	Notes
`Single`	Yes	
`Double`	Yes	
`Currency`	Yes	
`String`	Yes	Variable-length strings
`Date`	Yes	Stored as `Double` internally
`Variant`	Yes	Full support
`Object`	Yes	COM objects via `Application` and `CreateObject()`

49.2.3 Control Flow

Feature	Supported
`If...Then...Else...End If`	Yes
`Select Case`	Yes
`For...Next`	Yes
`For Each...Next`	Yes
`Do...Loop` (While/Until)	Yes
`While...Wend`	Yes
`GoTo`	Yes
`GoSub...Return`	Yes
`On Error GoTo`	Yes
`On Error Resume Next`	Yes
`Exit Sub/Function/For/Do`	Yes
`End`	Yes

49.2.4 Procedures

Feature	Supported	Notes
`Sub`	Yes	
`Function`	Yes	
`Property Get/Let/Set`	Yes	
`Optional` parameters	Yes	
`ParamArray`	Yes	

Feature	Supported	Notes
`ByRef` / `ByVal`	Yes	`ByRef` is default
Recursive calls	Yes	
`Declare` (DLL calls)	Opt-in	Disabled by default; see DLL Calls
Module-level code	Yes	Executed on module initialization

49.2.5 Operators

All standard VBA operators are supported:

- **Arithmetic:** `+`, `-`, `*`, `/`, `\` (integer division), `Mod`, `^`
- **Comparison:** `=`, `<>`, `<`, `>`, `<=`, `>=`
- **Logical:** `And`, `Or`, `Not`, `Xor`, `Eqv`, `Imp`
- **String:** `&` (concatenation), `Like` (pattern matching)
- **Other:** `Is`, `IsNot`, `TypeOf...Is`

49.2.6 Built-in Functions

VBA Padlock includes a comprehensive set of built-in functions that mirror standard VBA functions. See the Script Functions Reference for the complete list, including:

- String functions (`Mid`, `Left`, `Right`, `InStr`, `InStrRev`, `Replace`, `Split`, `Join`, `Trim`, `LTrim`, `RTrim`, `UCase`, `LCase`, `StrComp`, `StrReverse`, `Chr`, `ChrW`, `Asc`, `AscW`, `Hex`, `Space`, `String`, `Format`)
- Math functions (`Abs`, `Sqr`, `Sin`, `Cos`, `Tan`, `Exp`, `Round`, `Int`, `Fix`, `Sgn`, `Rnd`)
- Date functions (`Now`, `Date`, `Time`, `DateAdd`, `DateDiff`, `Weekday`, `WeekdayName`, `MonthName`, `Year`, `Month`, `Day`, `Hour`, `Minute`, `Second`)
- Type functions (`TypeName`, `VarType`, `IsObject`, `IsMissing`, `IsArray`, `IsDate`, `IsEmpty`, `IsNull`, `IsNumeric`)
- Conversion functions (`CInt`, `CLng`, `CDbl`, `CSng`, `CStr`, `CBool`, `CByte`, `CDate`, `CCur`)
- Flow functions (`IIf`, `Choose`, `Switch`)
- Utility functions (`Array`, `DoEvents`, `RGB`, `Nz`, `LBound`, `UBound`, `Randomize`)

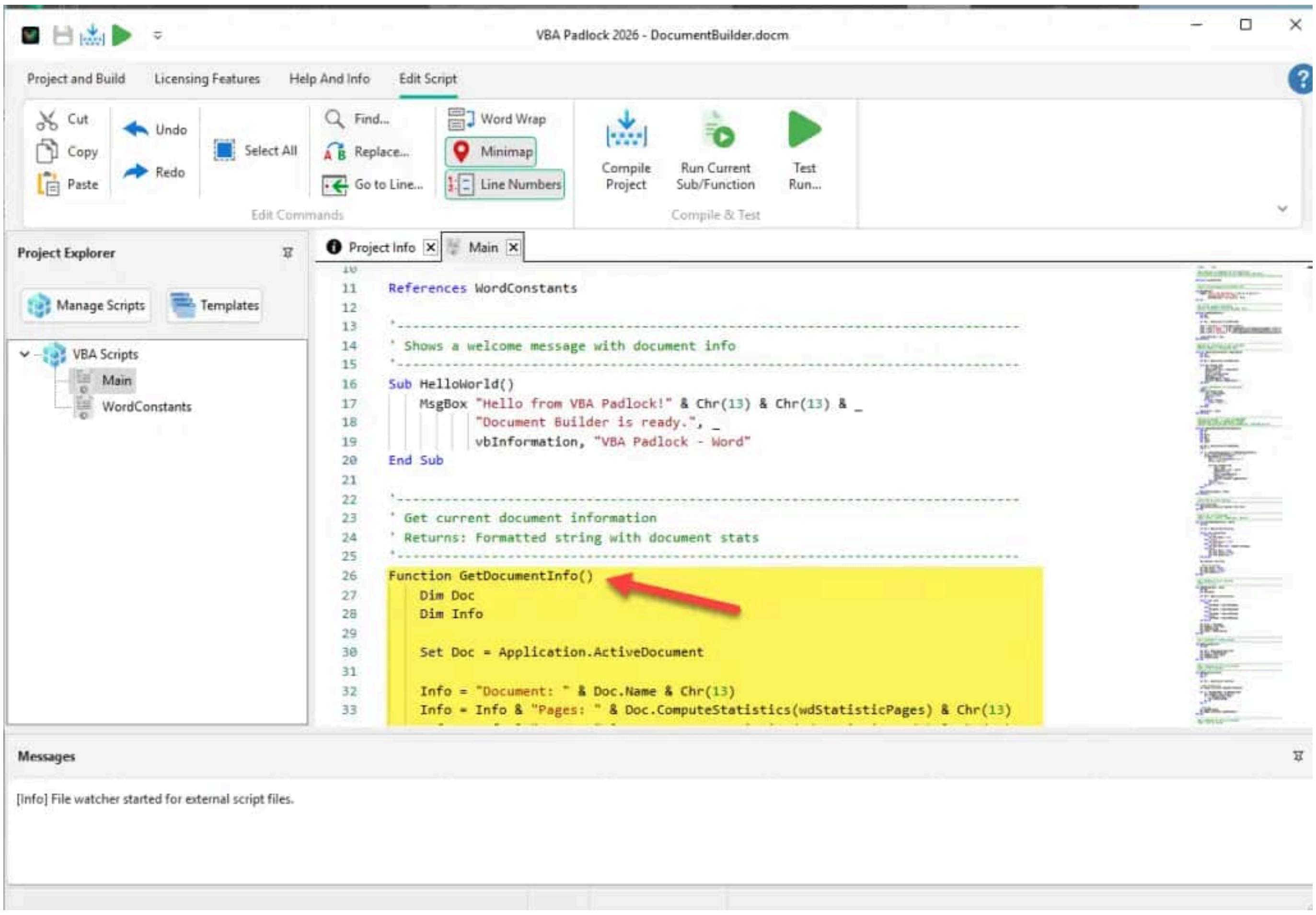
49.2.7 Constants templates and script references

VBA Padlock ships with a rich set of **ready-to-use script libraries** containing thousands of Office constants and utility functions. These `.bas` template files can be added to your project via the **Templates** button and referenced from any module using the `References` directive.

See the Script Libraries & Constants reference for the full catalog of available libraries, usage instructions, and code examples.

49.3 Office Object Model Access

Compiled scripts have **full access** to the host Office application through the `Application` object. You can manipulate worksheets, documents, presentations, and databases exactly as you would in regular VBA.



```
References WordConstants

'-------------------------------------------------------------------------
' Shows a welcome message with document info
'-------------------------------------------------------------------------
Sub HelloWorld()
    MsgBox "Hello from VBA Padlock!" & Chr(13) & Chr(13) & _
           "Document Builder is ready.", _
           vbInformation, "VBA Padlock - Word"
End Sub

'-------------------------------------------------------------------------
' Get current document information
' Returns: Formatted string with document stats
'-------------------------------------------------------------------------
Function GetDocumentInfo()
    Dim Doc
    Dim Info

    Set Doc = Application.ActiveDocument

    Info = "Document: " & Doc.Name & Chr(13)
    Info = Info & "Pages: " & Doc.ComputeStatistics(wdStatisticPages) & Chr(13)
```

```
[Info] File watcher started for external script files.
```

⚠ Caution

49.3.8 Where to use `Application.`?

It is vital to understand the distinction between your **Compiled Code** (protected) and your **Office VBA Code** (unprotected caller):

1. **In Protected Code (VBA Padlock Editor):** You **MUST** prefix all Office objects with `Application.`. For example, use `Application.ActiveSheet` instead of just `ActiveSheet`. This is because the compiled code runs as a separate DLL and needs an explicit reference to the host application.
2. **In Standard Office VBA (VBE Editor):** You write regular VBA as you always have. You only use `Application.` if you normally would in Excel.

If you forget `Application.` in your protected code, the compiler will return an "Identifier not found" error.

49.3.9 Protected code vs caller code (recommended workflow)

1. **Write sensitive logic in protected modules** (inside VBA Padlock Studio).
2. **Use explicit `Application.` references** for host objects.
3. **Call protected routines from Office VBA** through `VBAPL_Execute(...)`.

49.3.10 Examples in Protected Code

These examples show how you should write your code **inside the VBA Padlock Editor** to ensure it can be compiled successfully:

```
' PROTECTED CODE (Inside VBA Padlock)
Sub WriteToCell(CellAddress, Value)
    ' Correct: uses Application.
    Application.ActiveSheet.Range(CellAddress).Value = Value
End Sub

Function ReadFromCell(CellAddress)
    ' Correct: uses Application.
    ReadFromCell = Application.ActiveSheet.Range(CellAddress).Value
End Function
```

```
' PROTECTED CODE (Inside VBA Padlock)
Function GetDocumentInfo()
    Dim Doc
    Set Doc = Application.ActiveDocument
    GetDocumentInfo = "Document: " & Doc.Name & Chr(13) & _
                      "Pages: " & Doc.ComputeStatistics(2)
End Function
```

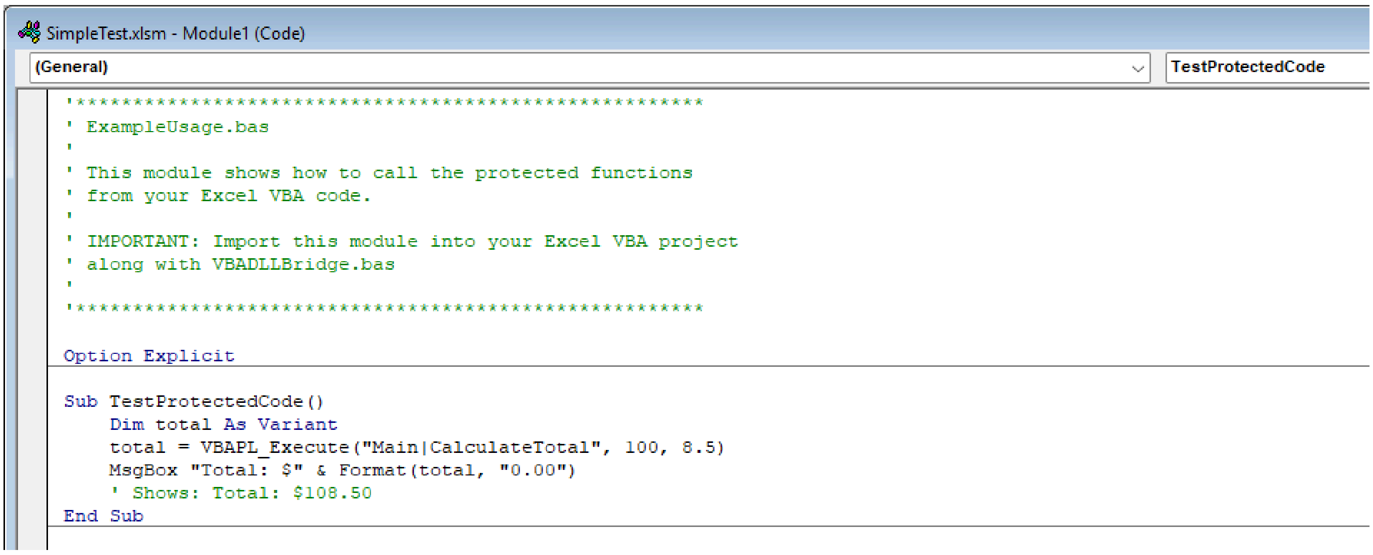
```
' PROTECTED CODE (Inside VBA Padlock)
Function GetDatabaseInfo()
    Dim DB
    Set DB = Application.CurrentDb
    GetDatabaseInfo = "Database: " & DB.Name & Chr(13) & _
                      "Queries: " & DB.QueryDefs.Count
End Function
```

```
' PROTECTED CODE (Inside VBA Padlock)
Function AddTitleSlide(Title, Subtitle)
    Dim Pres, Sld, SlideIndex
    Set Pres = Application.ActivePresentation
    SlideIndex = Pres.Slides.Count + 1
    Set Sld = Pres.Slides.Add(SlideIndex, 1)  ' ppLayoutTitle = 1
    Sld.Shapes.Title.TextFrame.TextRange.Text = Title
    Sld.Shapes.Placeholders(2).TextFrame.TextRange.Text = Subtitle
    AddTitleSlide = SlideIndex
End Function
```

49.3.11 The Caller Side (Office VBE)

On the other hand, the code you write in the **Excel/Word VBA Editor (Alt+F11)** acts as a simple bridge. It does not need any special prefixes:

```
' CALLER CODE (Inside Excel/Word VBE)
Sub RunMyProtectedCode()
    ' Regular VBA call to the compiled function
    Call VBAPL_Execute("WriteToCell", "A1", "Hello!")
End Sub
```



```
'**********************************************************
' ExampleUsage.bas
'
' This module shows how to call the protected functions
' from your Excel VBA code.
'
' IMPORTANT: Import this module into your Excel VBA project
' along with VBADLLBridge.bas
'
'**********************************************************

Option Explicit

Sub TestProtectedCode()
    Dim total As Variant
    total = VBAPL_Execute("Main|CalculateTotal", 100, 8.5)
    MsgBox "Total: $" & Format(total, "0.00")
    ' Shows: Total: $108.50
End Sub
```

49.3.12 COM Object Creation

`CreateObject()` is supported in compiled scripts. You can create and use COM objects such as `Scripting.Dictionary`, `ADODB.Connection`, `ADODB.Stream`, etc.

```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Set Dict = CreateObject("Scripting.Dictionary")

    Dim WS, LastRow, Row, Key
    Set WS = Application.ActiveSheet
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp

    For Row = 2 To LastRow
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Not Dict.Exists(Key) Then
            Dict.Add Key, Row
        End If
    Next Row

    RemoveDuplicates = Dict.Count
End Function
```

49.4 Limitations

These limitations are by design and keep the execution model secure and deterministic.

49.4.13 DLL Calls (Declare)

VBA Padlock supports standard VBA `Declare` statements to call functions in external DLLs (Windows APIs and third-party DLLs). The syntax is identical to regular VBA:

```
Declare Function GetTickCount Lib "kernel32" () As Long
Declare Function GetComputerNameW Lib "kernel32" (ByVal lpBuffer As String,
ByRef nSize As Long) As Long
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

Disabled by default

For security reasons, DLL calls are **blocked by default** at compile time. To enable this feature, you must request activation from G.D.G. Software. Once authorized, the permission flag is set in your project configuration and enforced at runtime.

Tip

Use the **Unicode** (`W`) versions of Windows APIs (e.g., `GetComputerNameW`, `GetTempPathW`) rather than the ANSI (`A`) versions. VBA Padlock uses UTF-16 strings internally, so ANSI functions may corrupt string buffers.

49.4.14 No Class Modules

Class modules (`Class` blocks) are not supported. Use standard modules with `Sub`, `Function`, and `Type` definitions instead.

49.4.15 No Forms (UserForms)

VBA UserForms are not available in compiled scripts. VBA Padlock provides its own built-in dialogs (activation, trial nag, EULA) configured through VBA Padlock Studio.

49.4.16 No Debug Object

The `Debug.Print` statement is not available. Use `DebugPrint` (from the Extension Library) instead, which outputs to `OutputDebugString` — visible with tools like DebugView or Visual Studio's Output window.

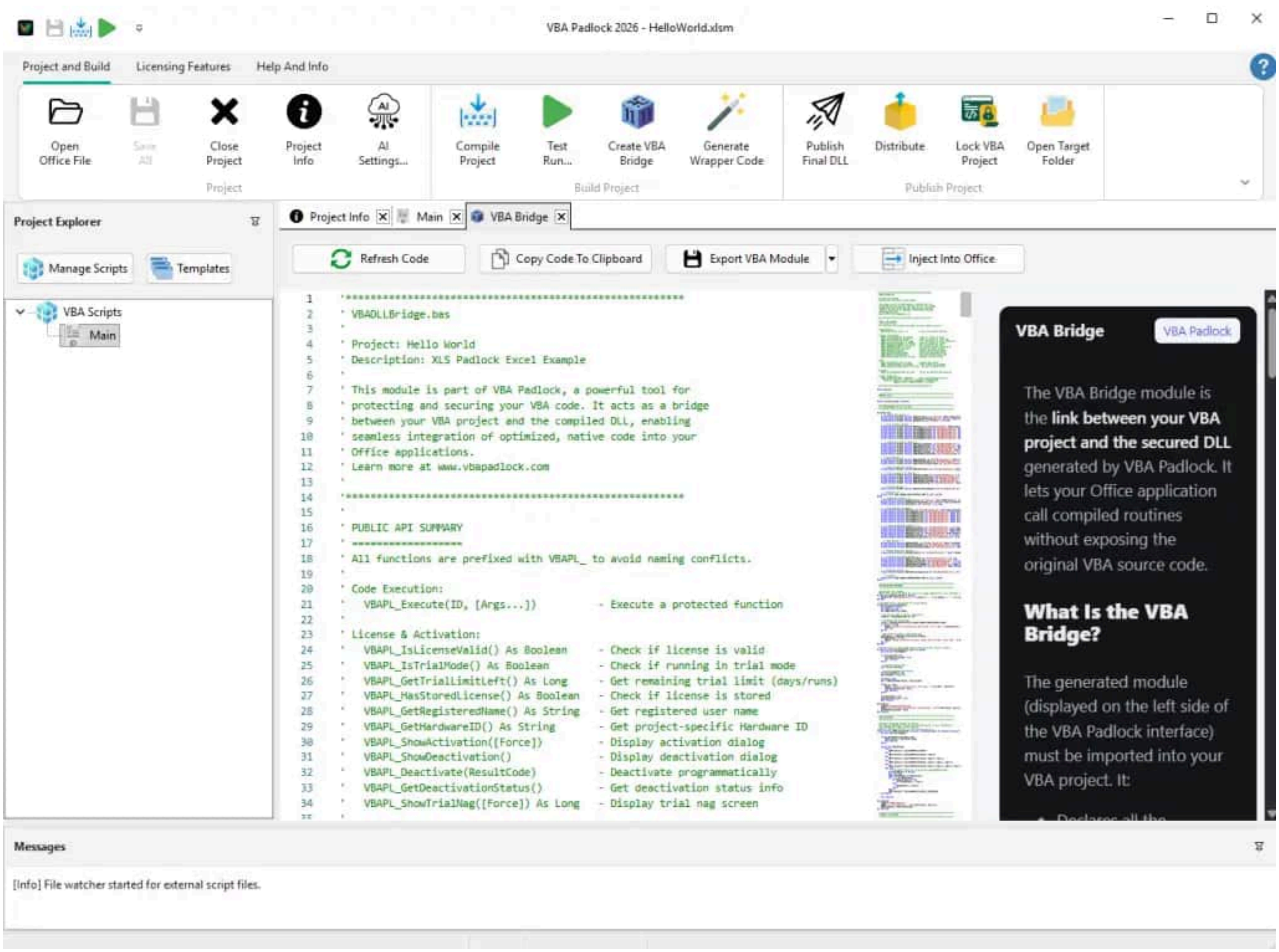
```
' Instead of Debug.Print:
DebugPrint "Value of x = " & x
```

49.4.17 Parameters for VBAPL_Execute

`VBAPL_Execute` supports **any number of parameters**. Calls with 0 to 4 arguments use optimized dedicated DLL exports, while calls with 5 or more arguments are automatically packed into an array and routed through `ExecuteVBAFunctionN`. This is handled transparently.

```
' Call with few parameters
result = VBAPL_Execute("Main|CalculatePrice", quantity, unitPrice)

' Call with many parameters (no limit)
result = VBAPL_Execute("Reports|Generate", title, startDate, endDate, format,
outputPath, includeCharts)
```

When omitting the module name (e.g., `VBAPL_Execute("MyFunction", arg1)`), the function is searched in the default `Main` script. To call a function in another module, use the `"ModuleName|FunctionName"` syntax.



```
'*************************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'*************************************************************
'
' PUBLIC API SUMMARY
' ------------------
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])           - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean      - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean         - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long      - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean    - Check if license is stored
'   VBAPL_GetRegisteredName() As String    - Get registered user name
'   VBAPL_GetHardwareID() As String        - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])          - Display activation dialog
'   VBAPL_ShowDeactivation()               - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)           - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()          - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long    - Display trial nag screen
'
```

VBA Bridge VBA Padlock

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

[Info] File watcher started for external script files.

49.5 Differences from Standard VBA

49.5.18 String Handling

Compiled scripts use Unicode (UTF-16) strings internally, consistent with modern VBA. String comparisons default to binary comparison unless `Option Compare Text` is specified.

49.5.19 Error Handling

`On Error GoTo` and `On Error Resume Next` work as expected. The `Err` object provides `Number` and `Description` properties. However, `Err.Raise` with custom error numbers should use values above 512 to avoid conflicts with built-in errors.

49.5.20 Module Initialization

Module-level code (code outside any `Sub` or `Function`) is executed when the module is first loaded. This happens on the first call to any function in that module.

49.5.21 Numeric Precision

- `Integer` is 16-bit signed (−32,768 to 32,767)
- `Long` is 32-bit signed (−2,147,483,648 to 2,147,483,647)
- `LongLong` is 64-bit signed
- `Double` follows IEEE 754 double-precision
- Arithmetic rounding uses the `Round` function (arithmetic rounding, not banker's rounding)

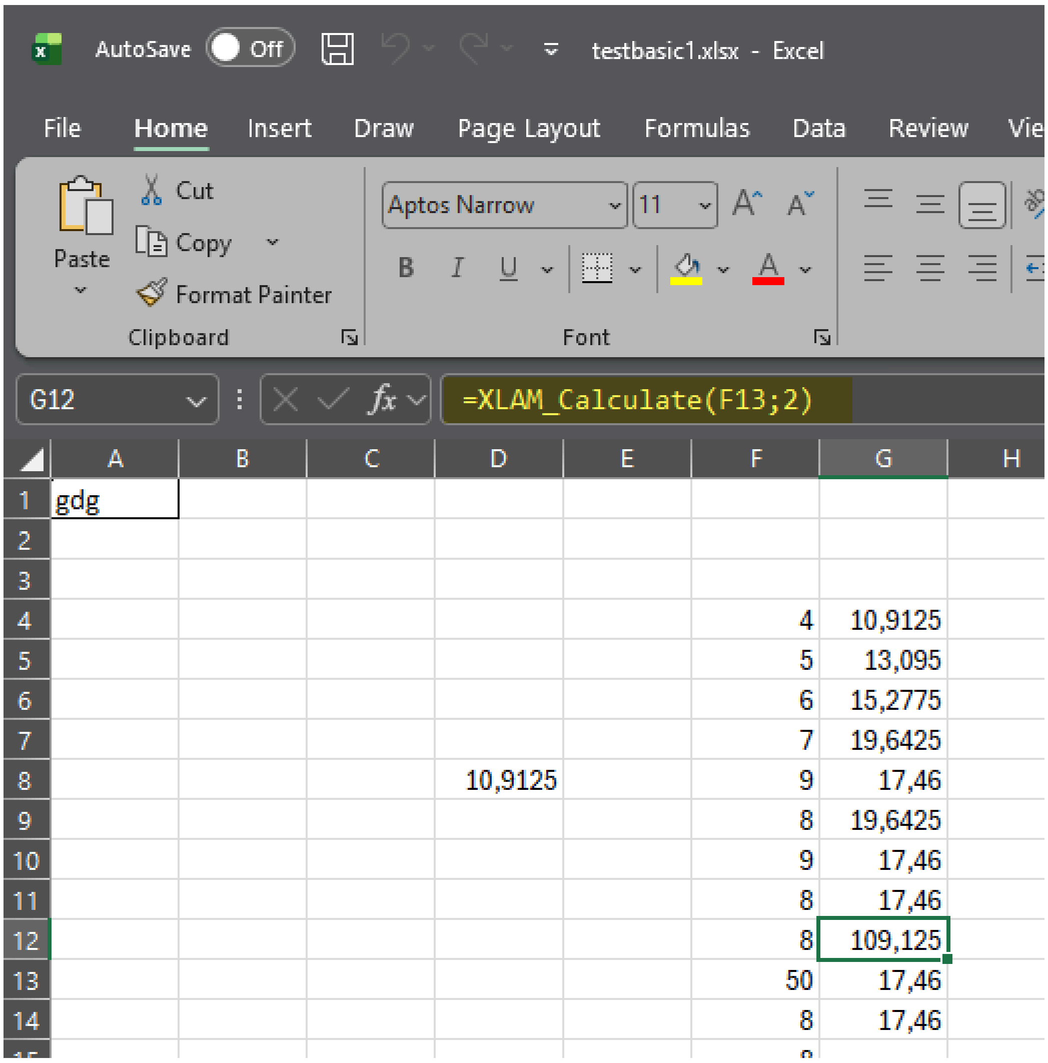
49.6 Supported Office Applications

VBA Padlock can protect projects for these Office applications:

Application	File Types	Notes
Excel	`.xlsm`, `.xlsb`, `.xla`, `.xlam`	Most common use case
Word	`.docm`, `.dotm`	Documents and templates
Access	`.accdb`, `.accde`	Databases
PowerPoint	`.pptm`, `.ppam`	Presentations and add-ins

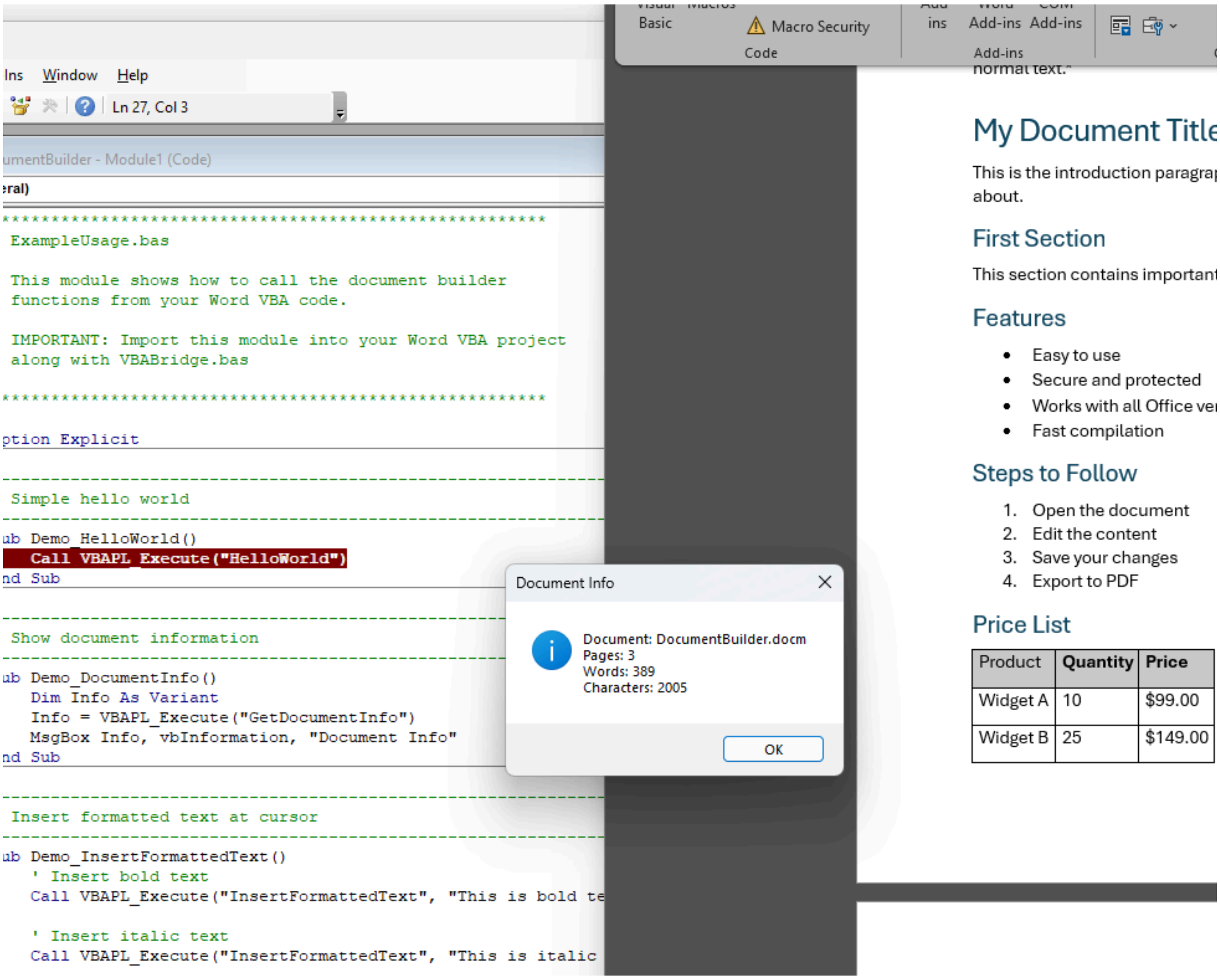
All Office versions from **Office 2016** through **Office 2024** and **Microsoft 365 desktop apps** are supported and tested, in both **32-bit and 64-bit** editions.

> **Windows ARM Support**
>
> Native Office ARM is not supported yet. However, on **Windows ARM** devices (like **Surface Pro 11, Surface Laptop 7, or Snapdragon X Elite** powered laptops), we recommend installing **Office 64-bit**. It runs seamlessly via Microsoft's **Prism** emulation, allowing VBA Padlock's 64-bit DLLs to function perfectly.



G12 =XLAM_Calculate(F13;2)

	A	B	C	D	E	F	G	H
1	gdg							
2								
3								
4						4	10,9125	
5						5	13,095	
6						6	15,2775	
7						7	19,6425	
8				10,9125		9	17,46	
9						8	19,6425	
10						9	17,46	
11						8	17,46	
12						8	109,125	
13						50	17,46	
14						8	17,46	



```
'*********************************************************
' ExampleUsage.bas

' This module shows how to call the document builder
' functions from your Word VBA code.

' IMPORTANT: Import this module into your Word VBA project
' along with VBABridge.bas

'*********************************************************

ption Explicit

'--------------------------------------------------------------
' Simple hello world
'--------------------------------------------------------------
ub Demo_HelloWorld()
    Call VBAPL_Execute("HelloWorld")
nd Sub

'--------------------------------------------------------------
' Show document information
'--------------------------------------------------------------
ub Demo_DocumentInfo()
    Dim Info As Variant
    Info = VBAPL_Execute("GetDocumentInfo")
    MsgBox Info, vbInformation, "Document Info"
nd Sub

'--------------------------------------------------------------
' Insert formatted text at cursor
'--------------------------------------------------------------
ub Demo_InsertFormattedText()
    ' Insert bold text
    Call VBAPL_Execute("InsertFormattedText", "This is bold te

    ' Insert italic text
    Call VBAPL_Execute("InsertFormattedText", "This is italic
```

```
Document: DocumentBuilder.docm
Pages: 3
Words: 389
Characters: 2005
```

normal text."

My Document Title

This is the introduction paragra about.

First Section

This section contains important

Features

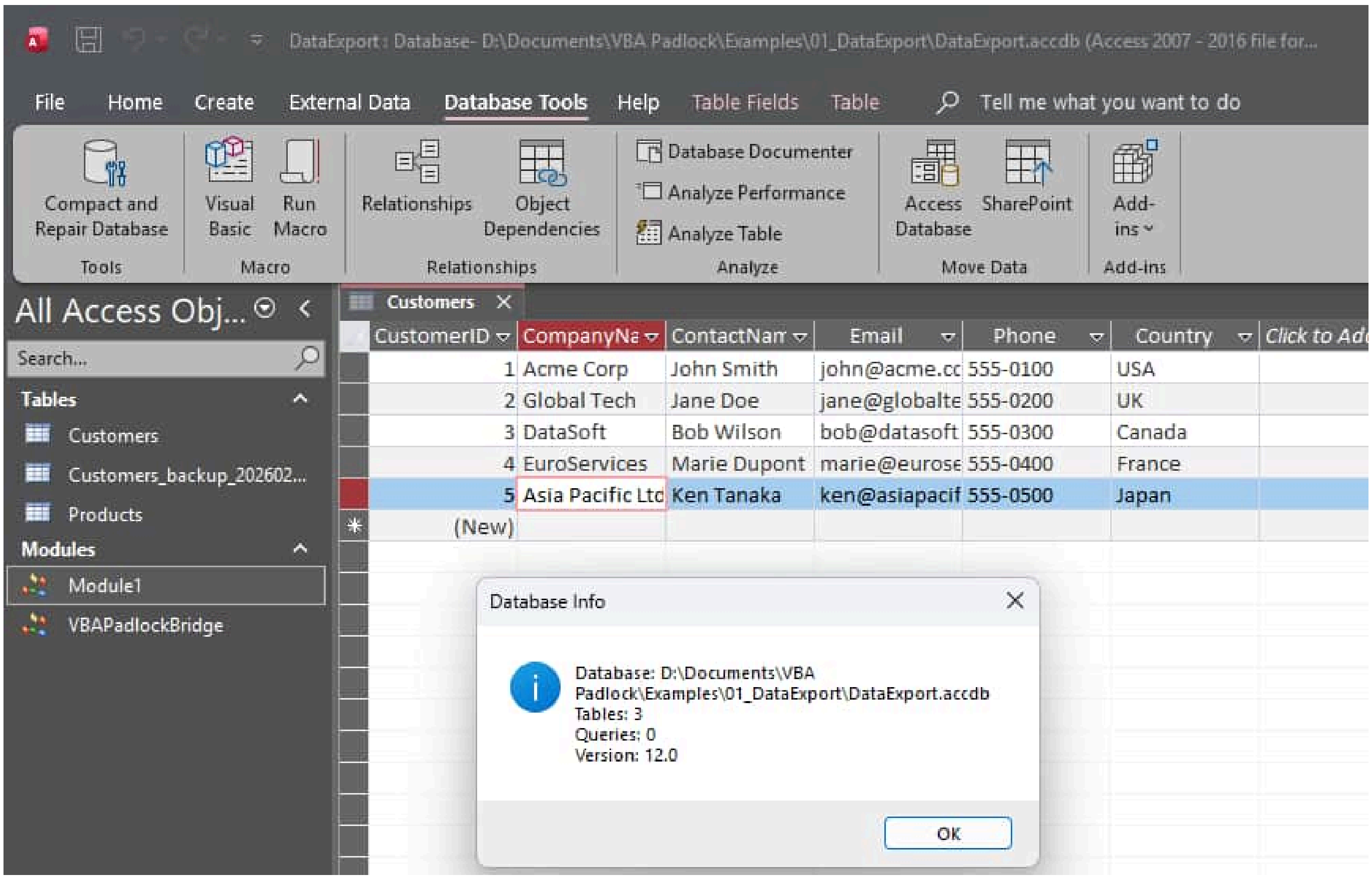
- Easy to use
- Secure and protected
- Works with all Office ve
- Fast compilation

Steps to Follow

1. Open the document
2. Edit the content
3. Save your changes
4. Export to PDF

Price List

Product	Quantity	Price
Widget A	10	$99.00
Widget B	25	$149.00

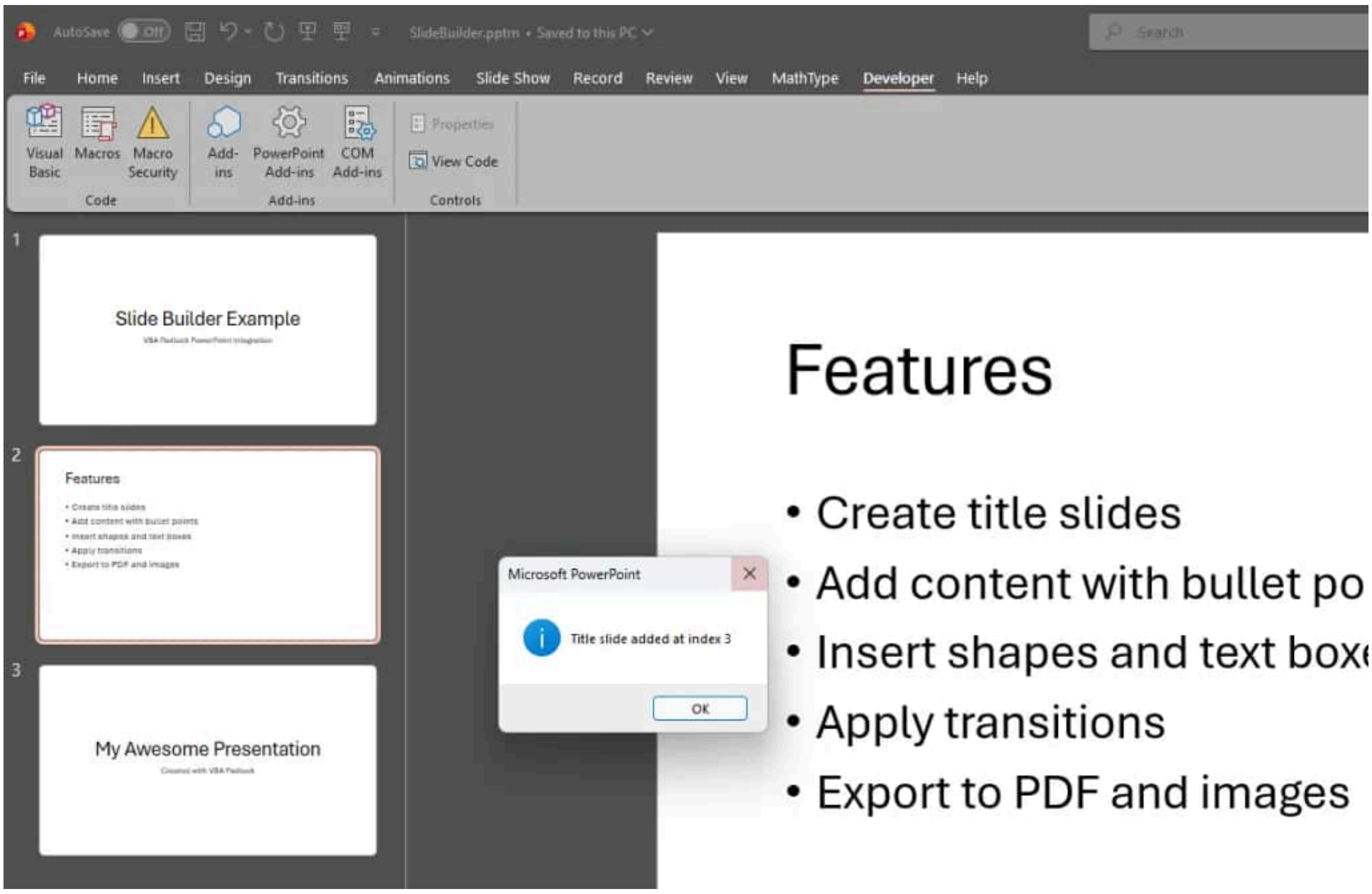

DataExport : Database- D:\Documents\VBA Padlock\Examples\01_DataExport\DataExport.accdb (Access 2007 - 2016 file for...
File Home Create External Data Database Tools Help Table Fields Table Tell me what you want to do
Compact and Repair Database
Visual Basic
Run Macro
Relationships
Object Dependencies
Database Documenter
Analyze Performance
Analyze Table
Access Database
SharePoint
Add-ins
Tools Macro Relationships Analyze Move Data Add-ins
All Access Obj...
Search...
Tables
Customers
Customers_backup_202602...
Products
Modules
Module1
VBAPadlockBridge
Customers

CustomerID	CompanyNa	ContactNam	Email	Phone	Country	Click to Add
1	Acme Corp	John Smith	john@acme.cc	555-0100	USA	
2	Global Tech	Jane Doe	jane@globalte	555-0200	UK	
3	DataSoft	Bob Wilson	bob@datasoft	555-0300	Canada	
4	EuroServices	Marie Dupont	marie@eurose	555-0400	France	
5	Asia Pacific Ltd	Ken Tanaka	ken@asiapacif	555-0500	Japan	
(New)						

Database Info

```
Database: D:\Documents\VBA
Padlock\Examples\01_DataExport\DataExport.accdb
Tables: 3
Queries: 0
Version: 12.0
```

OK



Features

- Create title slides
- Add content with bullet po
- Insert shapes and text boxe
- Apply transitions
- Export to PDF and images

Microsoft PowerPoint

```
Title slide added at index 3
```

OK

49.7 Best Practices

1. **Move sensitive logic into the compiled DLL.** Business rules, pricing algorithms, license checks, and proprietary calculations should all be in compiled scripts — that's where they are protected.

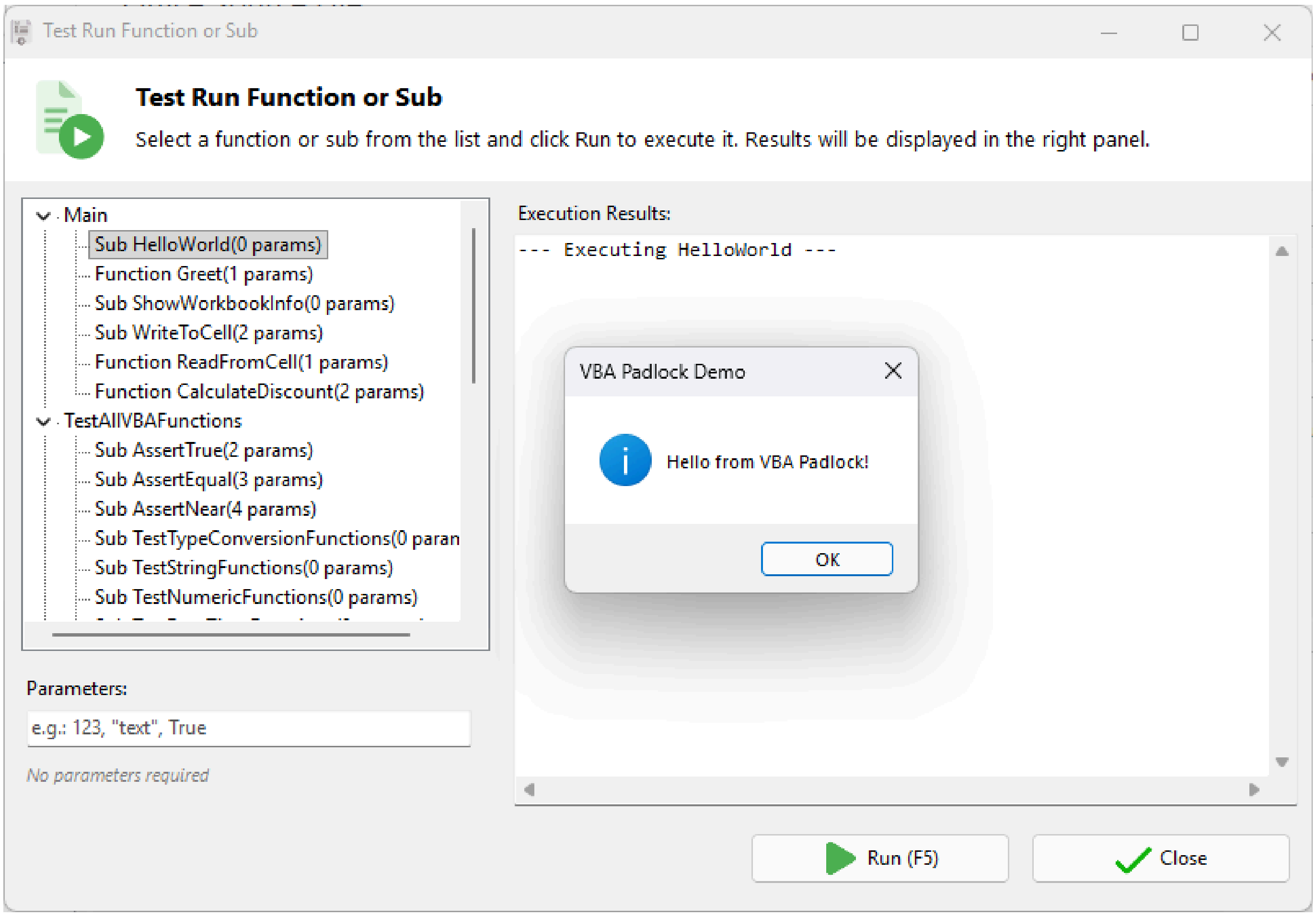
2. **Use `Application` for Office interaction.** Access worksheets, documents, ranges, and databases through the `Application` object inside compiled scripts rather than passing data back and forth unnecessarily.

3. **Test with the Test Runner.** Use VBA Padlock's built-in Test Runner to validate your compiled scripts before publishing.

4. **Use `Option Explicit`.** Declare all variables explicitly to catch typos and type mismatches at compile time.

5. **Keep modules focused.** Split your code into multiple `.bas` modules by responsibility. This improves readability and compilation diagnostics.

6. **Validate frequently with compilation + test run.** Catch compatibility issues early in the build cycle.


Test Run Function or Sub

Test Run Function or Sub

Select a function or sub from the list and click Run to execute it. Results will be displayed in the right panel.

- Main
  - Sub HelloWorld(0 params)
  - Function Greet(1 params)
  - Sub ShowWorkbookInfo(0 params)
  - Sub WriteToCell(2 params)
  - Function ReadFromCell(1 params)
  - Function CalculateDiscount(2 params)
- TestAllVBAFunctions
  - Sub AssertTrue(2 params)
  - Sub AssertEqual(3 params)
  - Sub AssertNear(4 params)
  - Sub TestTypeConversionFunctions(0 paran
  - Sub TestStringFunctions(0 params)
  - Sub TestNumericFunctions(0 params)

Parameters: e.g.: 123, "text", True

No parameters required

Execution Results:

```
--- Executing HelloWorld ---
```

VBA Padlock Demo

Hello from VBA Padlock!

49.8 See Also

- Script Functions Reference — All built-in functions available in compiled scripts

- Script Libraries & Constants — Office constants and utility modules for compiled scripts
- VBA Bridge API Reference — Calling the DLL from Office VBA
- Project Format Reference — Project folder structure
- Code Editor — Writing and editing compiled scripts in VBA Padlock Studio

50. Script Functions Reference

When you write VBA code inside VBA Padlock's code editor (the scripts that get compiled into the DLL), you have access to a set of **built-in functions** beyond standard VBA.

Extension Library →

50+ VBA-compatible functions for strings, math, dates, and more.

License Library →

16 functions to check license status, show dialogs, and validate online.

Locale Library →

8 functions for localization and translation support using JSON files.

Caution

These functions are available **only inside compiled scripts** (the VBA Padlock code editor). They are **not** available in your Office VBA code. For calling the DLL from Office VBA, see the VBA Bridge API Reference.

50.1 Extension Library

The extension library provides VBA-compatible functions that work inside compiled scripts. These mirror standard VBA functions so you can write natural VBA code.

50.1.1 String Functions

All string functions support the VBA `$` suffix variants (e.g., `Mid$`, `Left$`, `Trim$`) which return `String` instead of `Variant`.

Mid

```
Function Mid(S As String, Start As Integer, Optional Length As Integer) As
String
```

Returns a substring starting at `Start` (1-based).

Left / Right

```
Function Left(S As String, Length As Integer) As String
Function Right(S As String, Length As Integer) As String
```

Returns the leftmost or rightmost `Length` characters of a string.

Len

```
Function Len(S As String) As Integer
```

Returns the number of characters in a string.

InStr / InStrRev

```
Function InStr(Start As Integer, String1 As String, String2 As String,
Optional Compare As Integer = vbBinaryCompare) As Integer
Function InStrRev(StringCheck As String, StringMatch As String, Optional Start
As Integer = -1, Optional Compare As Integer = vbBinaryCompare) As Integer
```

Returns the position of the first (or last) occurrence of a substring. Returns `0` if not found.

Replace

```
Function Replace(Expression As String, Find As String, ReplaceWith As
String, Optional Start As Integer = 1, Optional Count As Integer = -1,
Optional Compare As Integer = vbBinaryCompare) As String
```

Replaces occurrences of a substring within a string. Set `Count` to limit the number of replacements.

Trim, LTrim, RTrim

```
Function Trim(S As String) As String
Function LTrim(S As String) As String
Function RTrim(S As String) As String
```

Removes leading and/or trailing spaces from a string. `LTrim` removes leading spaces only, `RTrim` trailing spaces only.

UCase, LCase

```
Function UCase(S As String) As String
Function LCase(S As String) As String
```

Converts a string to uppercase or lowercase.

Split / Join

```
Function Split(Expression As String, Optional Delimiter As String = " ",
Optional Limit As Integer = -1, Optional Compare As Integer =
vbBinaryCompare) As Variant
Function Join(SourceArray As Variant, Optional Delimiter As String = " ") As
String
```

`Split` breaks a string into an array of substrings using a delimiter. `Join` concatenates an array of strings into a single string.

Like

```
Function Like(S As String, Pattern As String) As Boolean
```

Pattern matching. Wildcards: `?` (any character), `*` (any characters), `#` (any digit), `[charlist]` (character list).

Format

```
Function Format(Expression As Variant, Optional FormatStr As String = "")
As String
```

Formats a number, date, or string according to a format specifier (e.g., `"#,##0.00"`, `"yyyy-mm-dd"`).

50.1.2 Numeric Functions

Function	Signature	Description
`Abs`	`Abs(X) As Double`	Absolute value
`Sqr`	`Sqr(X) As Double`	Square root
`Sin`, `Cos`, `Tan`	Trigonometric functions	Sine, Cosine, Tangent (radians)
`Exp`	`Exp(X) As Double`	Exponential (e^X)
`Round`	`Round(X, Optional Digits = 0) As Double`	Arithmetic rounding
`Val`	`Val(S As String) As Double`	Parse string to number
`Str`	`Str(X) As String`	Number to string

50.1.3 Date/Time Functions

Now / Date / Time

```
Function Now() As Double
Function Date() As Double
Function Time() As Double
```

`Now` returns the current date and time. `Date` returns the date portion only. `Time` returns the time portion only. All values are VBA `Date`-compatible doubles.

DateAdd / DateDiff

```
Function DateAdd(Interval As String, Number As Integer, DateValue As
Double) As Double
Function DateDiff(Interval As String, Date1 As Double, Date2 As Double, ...)
As LongLong
```

`DateAdd` adds a time interval to a date (e.g., `DateAdd("m", 3, Now())` adds 3 months). `DateDiff` returns the number of intervals between two dates. Supported intervals: `"yyyy"` (year), `"q"` (quarter), `"m"` (month), `"d"` (day), `"h"` (hour), `"n"` (minute), `"s"` (second).

50.1.4 Control Flow

IIf / Choose / Switch

```
Function IIf(Expression As Boolean, TruePart As Variant, FalsePart As
Variant) As Variant
Function Choose(Index As Integer, Choice1, ...) As Variant
Function Switch(Expr1, Value1, ...) As Variant
```

Inline conditional functions. `IIf` returns one of two values based on a condition. `Choose` returns a value from a list based on an index. `Switch` evaluates expression/value pairs and returns the value for the first `True` expression.

50.1.5 Array & Utility

Function	Signature	Description
`Array`	`Array(args...) As Variant`	Creates a 0-based array
`DoEvents`	`DoEvents() As Integer`	Processes pending Windows messages
`TypeName`	`TypeName(V) As String`	Returns the type name of a variable
`VarType`	`VarType(V) As Integer`	Returns the type code of a variable
`Nz`	`Nz(Value, Optional Default = 0) As Variant`	Returns `Default` if `Value` is `Null` or `Empty`
`RGB`	`RGB(Red, Green, Blue) As Integer`	Creates a color value

50.2 License Library

These functions let your compiled scripts interact with the licensing system at runtime.

50.2.6 License Status

IsLicenseValid

```
Function IsLicenseValid() As Boolean
```

Returns `True` if the current license is valid or if activation is not required.

IsTrialMode

```
Function IsTrialMode() As Boolean
```

Returns `True` if the application is currently running in trial mode.

HasStoredLicense

```
Function HasStoredLicense() As Boolean
```

Returns `True` if a license key is stored on this computer.

GetTrialDaysLeft

```
Function GetTrialDaysLeft() As Integer
```

Returns the remaining trial allowance. Returns `-1` if not in trial mode.

> **Note**
>
> Despite its name, this function returns the remaining limit in the configured unit (days, runs, or days until expiration date), not necessarily days.

GetTrialLimitType

```
Function GetTrialLimitType() As Integer
```

Returns the type of trial limit configured for this project.

Return	Meaning
`0`	Days (`tltDays`)
`1`	Runs (`tltRuns`)
`2`	Expiration date (`tltExpiration`)
`-1`	Not in trial mode

GetRegisteredName

```
Function GetRegisteredName() As String
```

Returns the licensee name, or an empty string if no license is stored.

GetHardwareID

```
Function GetHardwareID() As String
```

Returns the Hardware ID string for the current machine.

50.2.7 License Properties

GetLicenseProperty

```
Function GetLicenseProperty(PropertyName As String, Optional DefaultValue
As Variant = "") As Variant
```

Returns a project or license property by name.

Property Name	Description
`registeredname`	Licensee name
`licensekey`	Stored license key
`apptitle`	Application title
`projectguid`	Project unique identifier
`purchaseurl`	Purchase URL (configured in VBA Padlock Studio)

Example:

```
Dim title As String
title = GetLicenseProperty("apptitle", "My App")
```

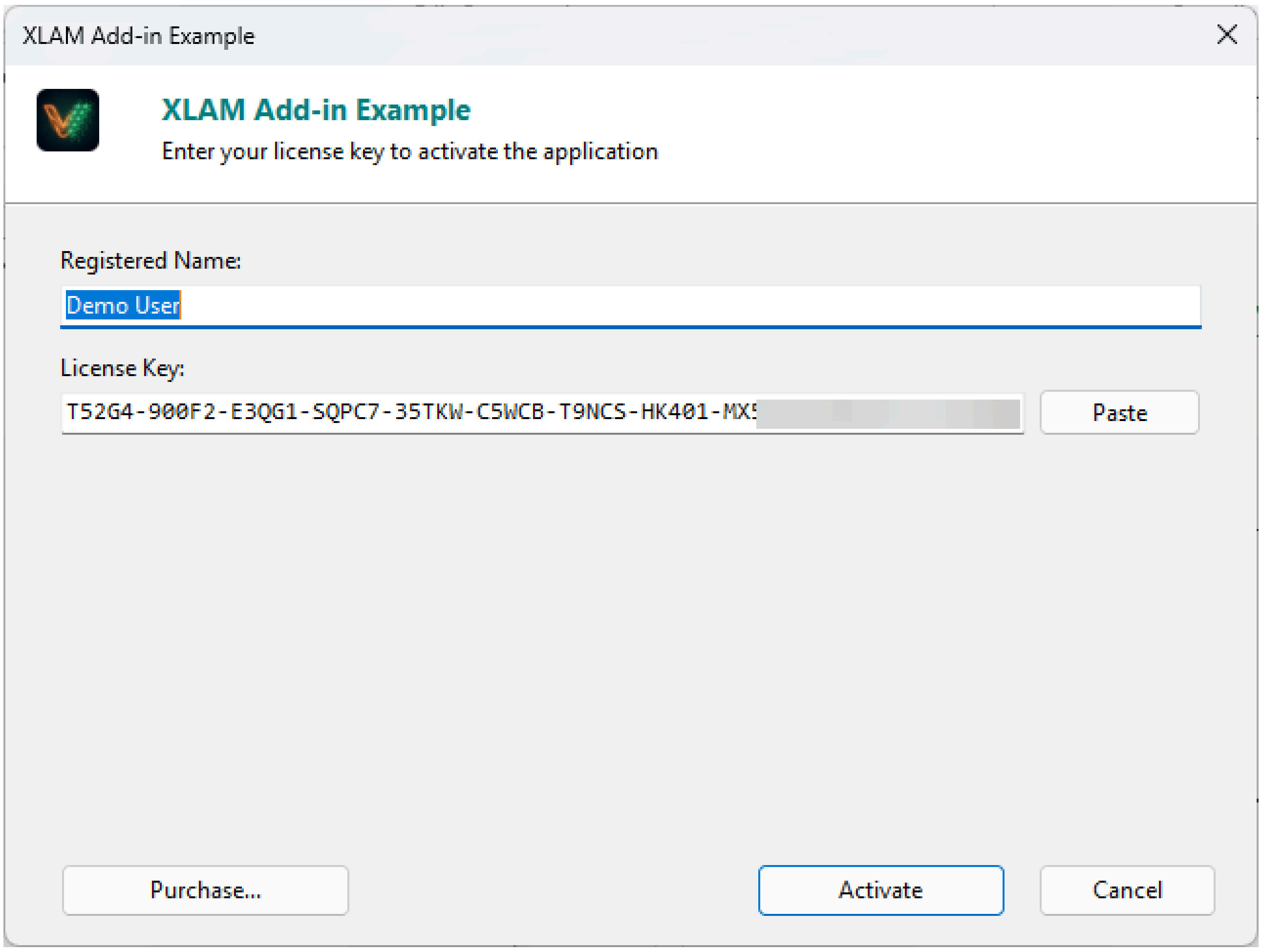
50.2.8 Dialogs & UI

VBA Padlock provides built-in dialogs that you can trigger directly from your scripts.

ShowActivationDialog

```
Function ShowActivationDialog(Optional ForceDisplay As Boolean = False) As
Integer
```

Displays the activation dialog. Returns `1` on success, `0` on cancel.



XLAM Add-in Example

XLAM Add-in Example

Enter your license key to activate the application

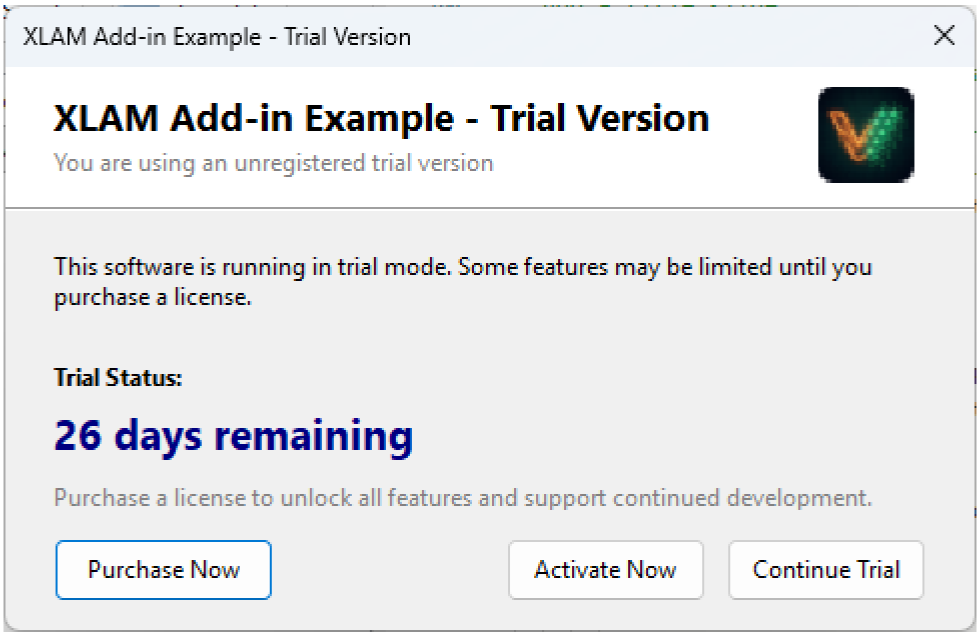
Registered Name: Demo User

License Key: T52G4-900F2-E3QG1-SQPC7-35TKW-C5WCB-T9NCS-HK401-MX5[redacted]

ShowTrialNag

```
Function ShowTrialNag(Optional ForceDisplay As Boolean = False) As Integer
```

Shows the trial nag screen. Returns `0` (continue), `1` (activate), or `2` (purchase).


XLAM Add-in Example - Trial Version

XLAM Add-in Example - Trial Version

You are using an unregistered trial version

This software is running in trial mode. Some features may be limited until you purchase a license.

Trial Status:

26 days remaining

Purchase a license to unlock all features and support continued development.

ShowDeactivateDialog

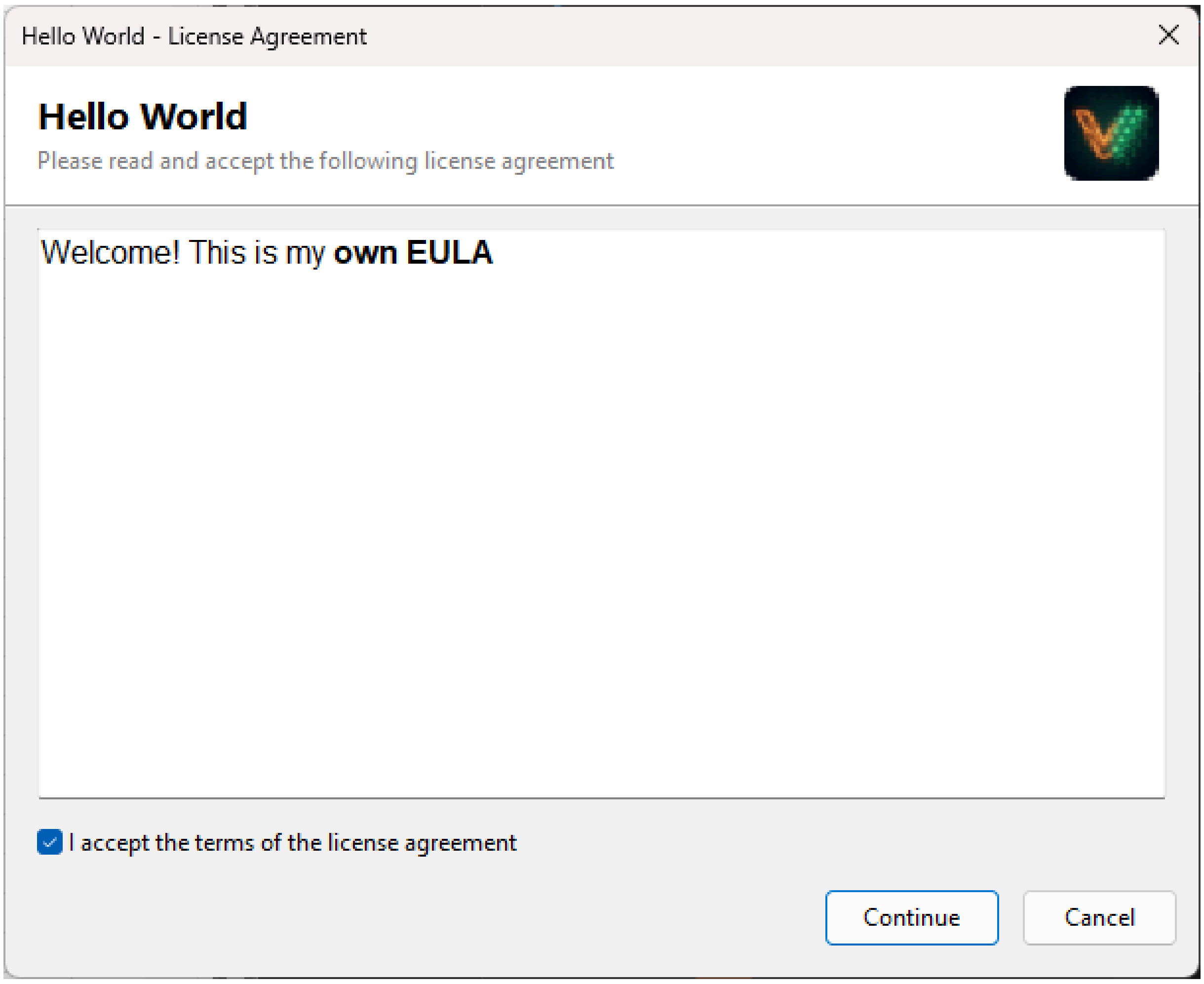
```
Function ShowDeactivateDialog() As Integer
```

Shows the deactivation dialog. Returns `0` on success, `1` on cancel.

ShowEULA

```
Function ShowEULA(Optional ForceDisplay As Boolean = False) As Integer
```

Displays the EULA dialog. Returns `1` (accepted), `0` (declined).


Hello World - License Agreement

Hello World

Please read and accept the following license agreement

Welcome! This is my **own EULA**

IsEulaAccepted

```
Function IsEulaAccepted() As Boolean
```

Returns `True` if the EULA has been accepted or if no EULA is configured.

GetEulaAcceptanceDate

```
Function GetEulaAcceptanceDate() As Double
```

Returns the EULA acceptance date as a `TDateTime` value (VBA `Date` compatible). Returns `0` if not accepted.

50.2.9 Online Validation

ValidateOnline

```
Function ValidateOnline(Optional Silent As Boolean = True) As Integer
```

Performs online license validation against the activation server.

Return	Meaning
`1`	Validation successful
`0`	Failed or license revoked
`-1`	Not initialized
`-2`	Online validation not configured
`-3`	No license stored
`-4`	Network error
`-5`	Server error
`-6`	Security validation failed

Example:

```
Dim result As Integer
result = ValidateOnline(False) ' Show errors to user

If result = 0 Then
    MsgBox "Your license has been revoked.", vbCritical
End If
```

IsValidationRequired

```
Function IsValidationRequired() As Integer
```

Returns `1` if an online validation check is due, `0` if not yet required.

50.3 Locale Library

These functions provide localization support for your compiled scripts. You can load custom translations from JSON files and use them in your code.

50.3.10 Implementation Steps

1. **Prepare a JSON file** with your translations (e.g., `{"welcome_message": "Bienvenue"}`).
2. **Load the locale** in your script using `LoadLocale(jsonContent)`.
3. **Retrieve strings** anywhere using `L("key")`.

50.3.11 Functions Reference

L

```
Function L(Key As String) As String
```

Returns the localized string for the given key. If the key is not found, returns the key itself.

Example:

```
MsgBox L("welcome_message")
```

LFmt

```
Function LFmt(Key As String, Args As Variant) As String
```

Returns a formatted localized string with placeholder substitution (up to 5 arguments).

Example:

```
' If the locale defines: "greeting" = "Hello, %s! You have %d messages."
Dim msg As String
msg = LFmt("greeting", Array("John", 5))
' Result: "Hello, John! You have 5 messages."
```

LPlural

```
Function LPlural(Key As String, Count As Integer) As String
```

Returns the appropriate plural form of a localized string based on the count.

LoadLocale

```
Function LoadLocale(JSONContent As String) As Boolean
```

Loads localization strings from a JSON string. Returns `True` on success.

GetCurrentLocale

```
Function GetCurrentLocale() As String
```

Returns the current locale code (e.g., `"en"`, `"fr"`), or an empty string if no locale is loaded.

IsLocaleLoaded

```
Function IsLocaleLoaded() As Boolean
```

Returns `True` if a custom locale has been loaded.

HasLocaleKey

```
Function HasLocaleKey(Key As String) As Boolean
```

Returns `True` if the given key exists in the current locale.

SetLocaleString

```
Function SetLocaleString(Key As String, Value As String) As Boolean
```

Adds or overrides a single locale string. Returns `True` on success.

50.4 Constants

Constant	Value	Description
`vbUpperCase`	`1`	Convert to uppercase
`vbLowerCase`	`2`	Convert to lowercase
`vbProperCase`	`3`	Capitalize first letter of each word
`vbUnicode`	`64`	Convert to Unicode
`vbFromUnicode`	`128`	Convert from Unicode

Constant	Value
`vbSunday`	`1`
`vbMonday`	`2`
`vbTuesday`	`3`
`vbWednesday`	`4`
`vbThursday`	`5`
`vbFriday`	`6`
`vbSaturday`	`7`

Constant	Description
`vbEmpty`	Uninitialized
`vbNull`	Null
`vbInteger`	Integer
`vbLong`	Long integer
`vbString`	String
`vbObject`	Object
`vbBoolean`	Boolean
`vbArray`	Array

Constant	Value	Description
`vbCrLf`	`Chr(13) & Chr(10)`	Carriage return + line feed
`vbTab`	`Chr(9)`	Tab
`vbNullString`	`""`	Empty string

50.5 See Also

- Script Libraries & Constants — Office constants and utility modules for compiled scripts
- VBA Bridge API Reference — Functions for calling the DLL from Office VBA
- VBA Compatibility — Supported VBA features and limitations

51. Script Libraries & Constants

VBA Padlock ships with a set of **ready-to-use script libraries** — `.bas` modules containing thousands of Office VBA constants and utility functions that you can add to any project. These libraries let you use named constants like `xlCenter`, `wdAlignParagraphLeft`, or `msoShapeRectangle` directly in your compiled scripts, without needing a type library reference.

> **Why use libraries?**
>
> This is especially useful for **late-binding** scenarios where your compiled code interacts with Office objects through `Application` and `CreateObject` calls — you get readable constant names instead of raw numeric values, making your code easier to maintain and debug.

51.1 Available Libraries

Excel Constants

~2,900 `xl*` and `rgb*` constants for full Excel automation.

Word Constants

~2,200 `wd*` constants for document formatting and manipulation.

PowerPoint Constants

~1,000 `pp*` constants for slides, transitions, and effects.

Access Constants

~1,400 `ac*` constants for forms, reports, and data operations.

Office Shared

~3,350 `mso*` constants for shapes, fills, and shared UI elements.

Utility Libraries

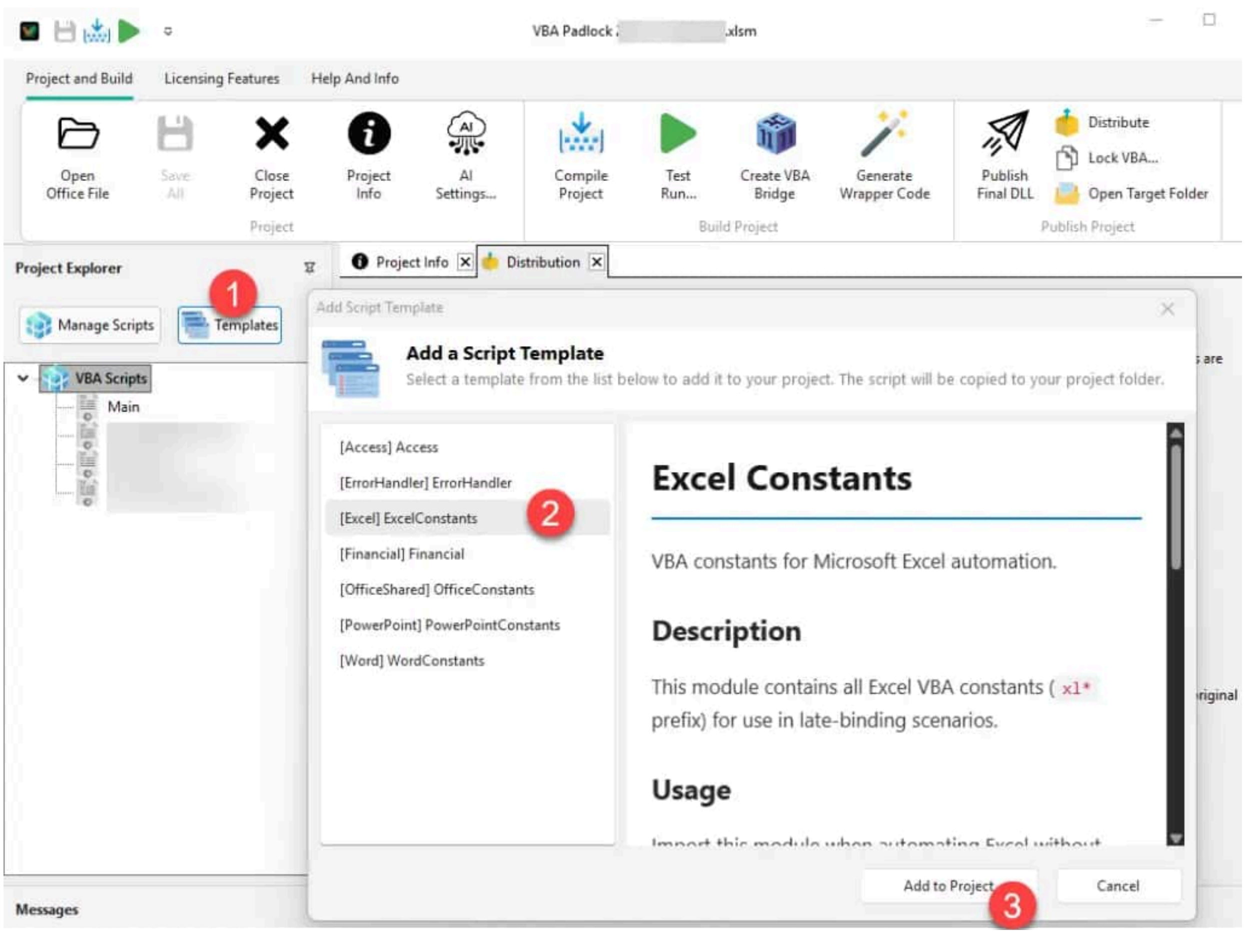
Error handling and Financial functions ready for your business logic.

51.2 Adding a Library to Your Project

Learn how to import these modules into your VBA Padlock project in three easy steps.

1 **Open the Templates gallery**

In the VBA Padlock code editor, click the **Templates** button in the toolbar.



Add Script Template

Add a Script Template

Select a template from the list below to add it to your project. The script will be copied to your project folder.

[Access] Access
[ErrorHandler] ErrorHandler
[Excel] ExcelConstants
[Financial] Financial
[OfficeShared] OfficeConstants
[PowerPoint] PowerPointConstants
[Word] WordConstants

Excel Constants

VBA constants for Microsoft Excel automation.

Description

This module contains all Excel VBA constants (`xl*` prefix) for use in late-binding scenarios.

Usage

Add to Project Cancel

2 **Select the library you need**

Browse the categorized list on the left and click any library to preview its content and documentation.

3 **Add it to your project**

Click **Add to Project**. This creates a new `.bas` module in your project containing all the constants or functions from the selected library.

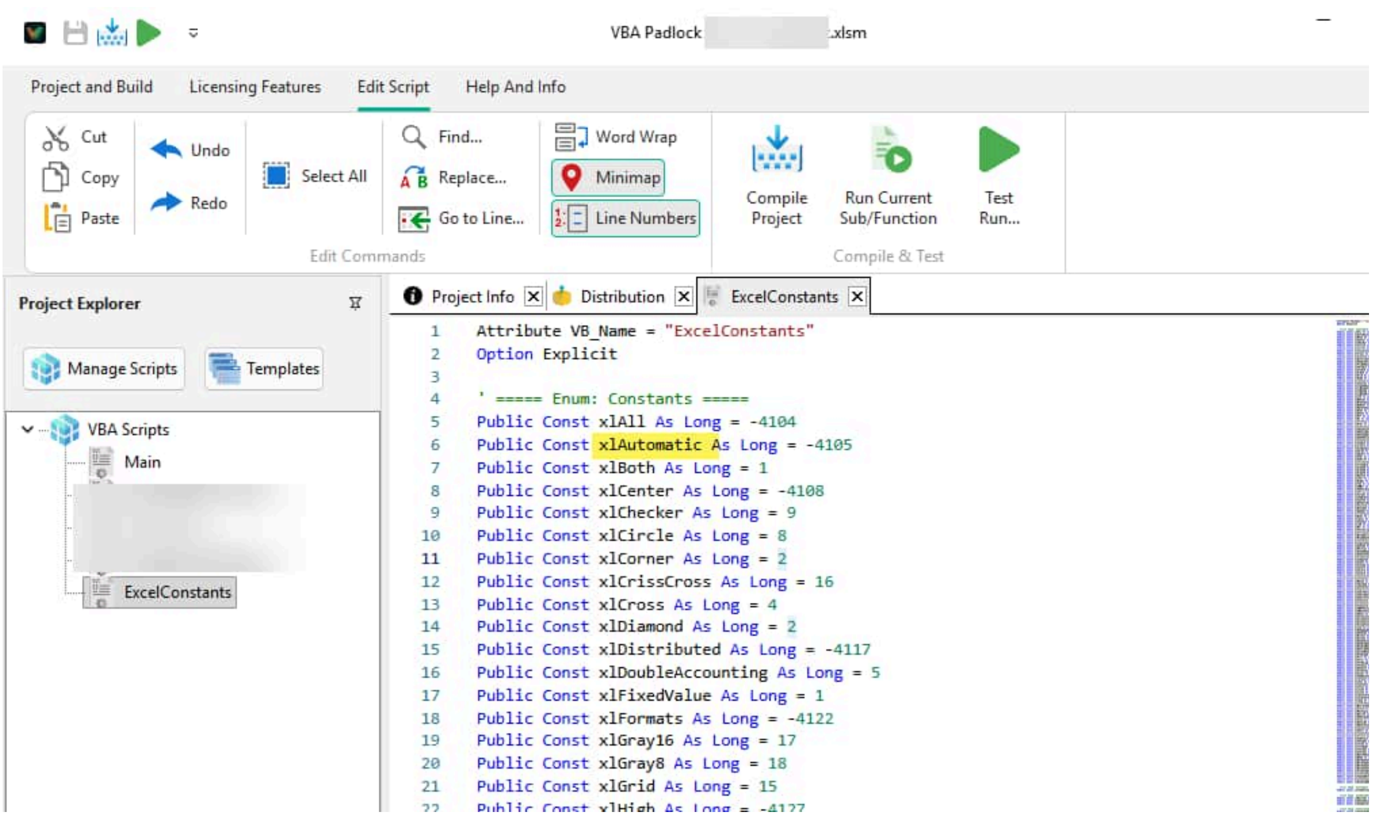
Note

You only need to add a library **once** per project. The module stays in your project folder and is compiled into your DLL along with the rest of your code.

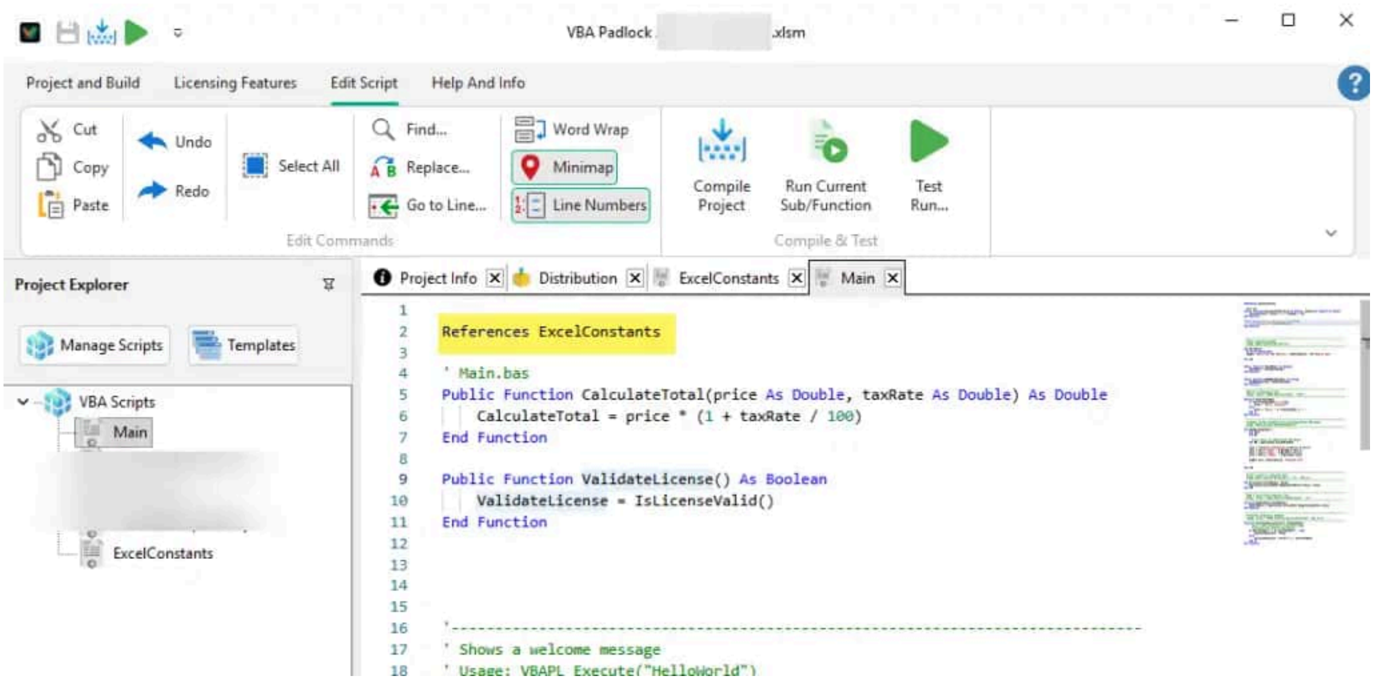
51.3 Referencing a Library

Once a library module is part of your project, any other module can access its constants by adding a `References` directive at the very top of the file. For instance:

```
References ExcelConstants
```



```
Attribute VB_Name = "ExcelConstants"
Option Explicit

' ===== Enum: Constants =====
Public Const xlAll As Long = -4104
Public Const xlAutomatic As Long = -4105
Public Const xlBoth As Long = 1
Public Const xlCenter As Long = -4108
Public Const xlChecker As Long = 9
Public Const xlCircle As Long = 8
Public Const xlCorner As Long = 2
Public Const xlCrissCross As Long = 16
Public Const xlCross As Long = 4
Public Const xlDiamond As Long = 2
Public Const xlDistributed As Long = -4117
Public Const xlDoubleAccounting As Long = 5
Public Const xlFixedValue As Long = 1
Public Const xlFormats As Long = -4122
Public Const xlGray16 As Long = 17
Public Const xlGray8 As Long = 18
Public Const xlGrid As Long = 15
Public Const xlHigh As Long = -4127
```



```

References ExcelConstants

' Main.bas
Public Function CalculateTotal(price As Double, taxRate As Double) As Double
    CalculateTotal = price * (1 + taxRate / 100)
End Function

Public Function ValidateLicense() As Boolean
    ValidateLicense = IsLicenseValid()
End Function





'-----------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
```

51.3.1 Key Rules

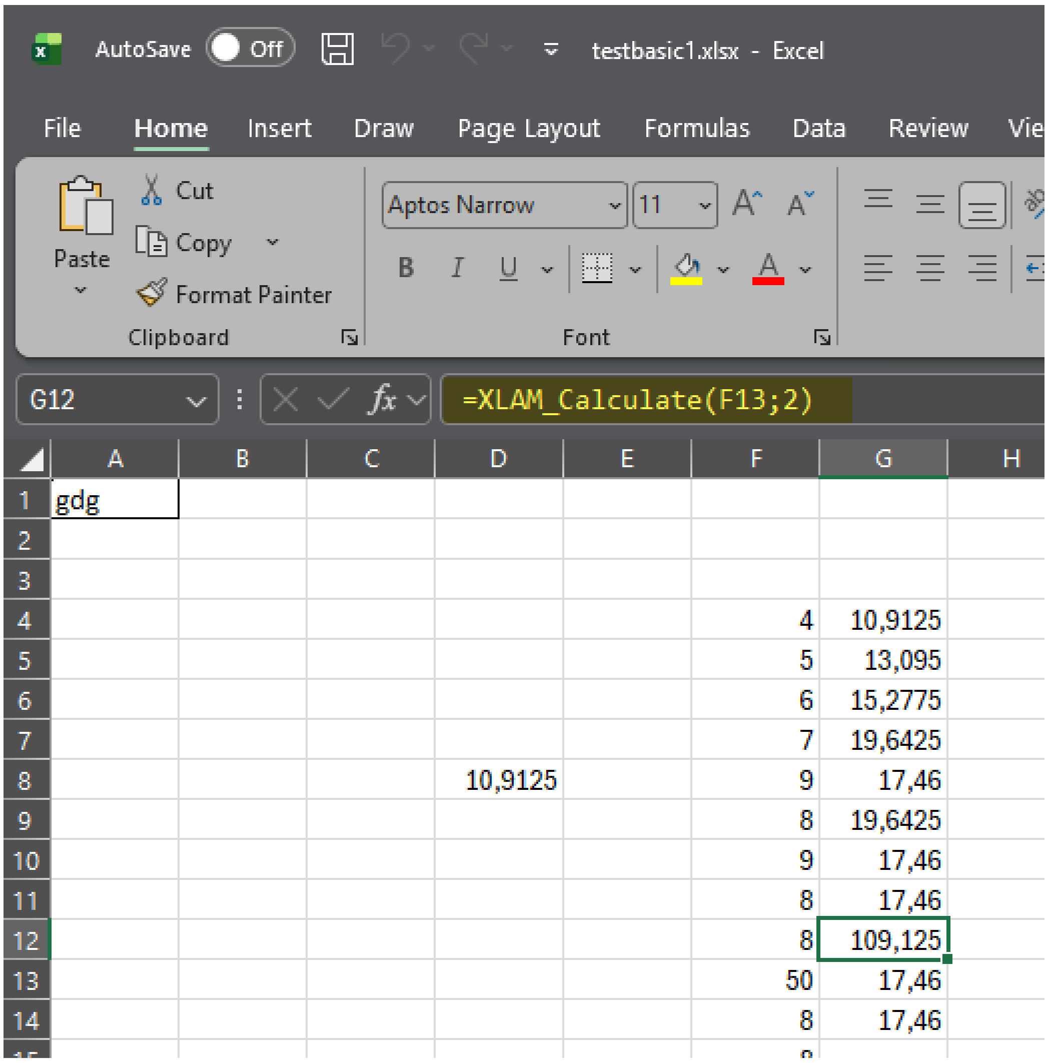
- The `References` directive must appear **before** any `Option` statement or code.
- It uses the **module name** (as shown in the Project Explorer), not the file name.
- You can reference multiple modules by adding several `References` lines.

51.4 Office Constants Reference

Explore the constants available for each Microsoft Office application.

51.4.2 ExcelConstants

Contains ~2,900 constants for automating Microsoft Excel. Covers the most commonly used `xl*` enumerations plus 150+ named RGB colors.



G12 =XLAM_Calculate(F13;2)

	A	B	C	D	E	F	G	H
1	gdg							
2								
3								
4						4	10,9125	
5						5	13,095	
6						6	15,2775	
7						7	19,6425	
8				10,9125		9	17,46	
9						8	19,6425	
10						9	17,46	
11						8	17,46	
12						8	109,125	
13						50	17,46	
14						8	17,46	

Key Categories:

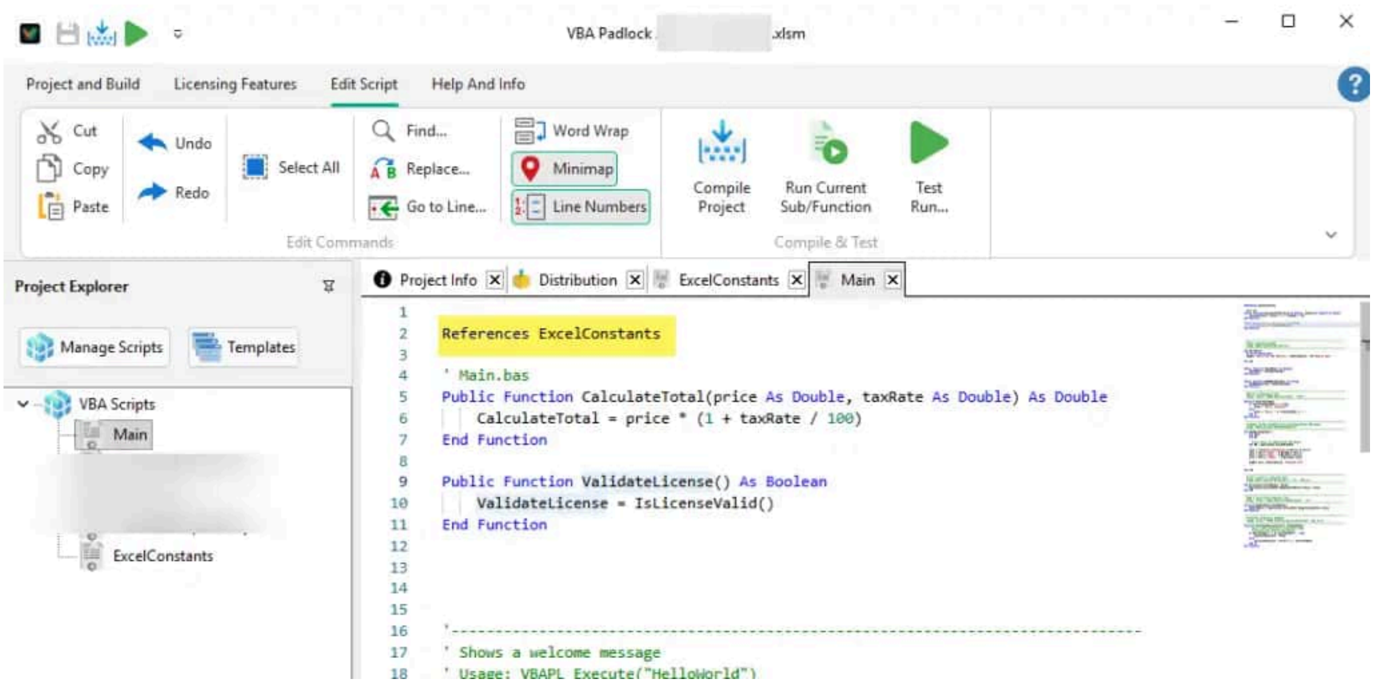
Category	Examples	Count
Alignment	`xlCenter`, `xlLeft`, `xlRight`	20+
Borders	`xlContinuous`, `xlDash`, `xlDot`	15+
Charts	`xlColumnClustered`, `xlLine`, `xlPie`	40+
Colors	`rgbRed`, `rgbBlue`, `rgbCornflowerBlue`	150+
Formats	`xlOpenXMLWorkbook`, `xlCSV`, `xlAddIn`	40+

Example:

```
References ExcelConstants

Function FormatReport()
    Dim WS
    Set WS = Application.ActiveSheet
    WS.Range("A1:D1").HorizontalAlignment = xlCenter
    WS.Range("A1:D10").Borders.LineStyle = xlContinuous
    WS.Range("A1:D1").Interior.Color = rgbSteelBlue
    FormatReport = "Done"
End Function
```

51.4.3 WordConstants

Contains ~2,200 constants for automating Microsoft Word. Covers paragraph formatting, document units, find/replace, and more.



```
References ExcelConstants

' Main.bas
Public Function CalculateTotal(price As Double, taxRate As Double) As Double
    CalculateTotal = price * (1 + taxRate / 100)
End Function

Public Function ValidateLicense() As Boolean
    ValidateLicense = IsLicenseValid()
End Function




'--------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL Execute("HelloWorld")
```

Key Categories:

Category	Examples	Count
Alignment	`wdAlignParagraphLeft`, `wdAlignParagraphCenter`	9
Units	`wdCharacter`, `wdWord`, `wdParagraph`	16
Breaks	`wdPageBreak`, `wdSectionBreakNextPage`	11
Formats	`wdFormatDocument`, `wdFormatPDF`, `wdFormatRTF`	24+

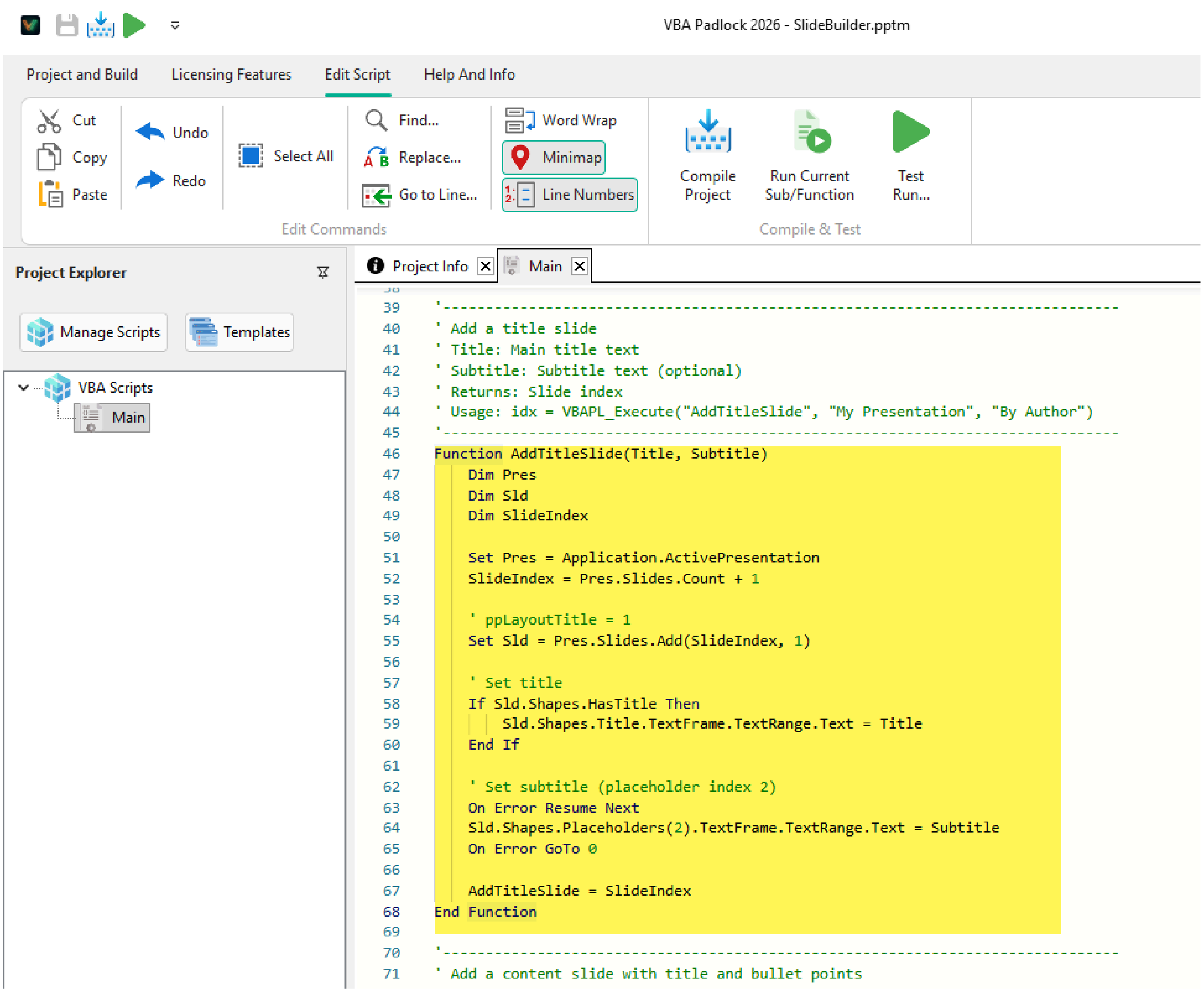
Category	Examples	Count
Tables	`wdTableFormatSimple1`, `wdTableFormatElegant`	42+

Example:

```
References WordConstants

Function FormatDocument()
    Dim Doc
    Set Doc = Application.ActiveDocument
    Doc.Paragraphs(1).Alignment = wdAlignParagraphCenter
    Doc.Range.InsertBreak wdPageBreak
    Doc.Tables(1).Borders.OutsideLineStyle = wdLineStyleSingle
    FormatDocument = "Done"
End Function
```

51.4.4 PowerPointConstants

Contains ~1,000 constants for automating Microsoft PowerPoint. Covers slide layouts, transitions, animation effects, and export formats.



```
'-------------------------------------------------------------------------------
' Add a title slide
' Title: Main title text
' Subtitle: Subtitle text (optional)
' Returns: Slide index
' Usage: idx = VBAPL_Execute("AddTitleSlide", "My Presentation", "By Author")
'-------------------------------------------------------------------------------
Function AddTitleSlide(Title, Subtitle)
    Dim Pres
    Dim Sld
    Dim SlideIndex

    Set Pres = Application.ActivePresentation
    SlideIndex = Pres.Slides.Count + 1

    ' ppLayoutTitle = 1
    Set Sld = Pres.Slides.Add(SlideIndex, 1)

    ' Set title
    If Sld.Shapes.HasTitle Then
        Sld.Shapes.Title.TextFrame.TextRange.Text = Title
    End If

    ' Set subtitle (placeholder index 2)
    On Error Resume Next
    Sld.Shapes.Placeholders(2).TextFrame.TextRange.Text = Subtitle
    On Error GoTo 0

    AddTitleSlide = SlideIndex
End Function

'-------------------------------------------------------------------------------
' Add a content slide with title and bullet points
```

Key Categories:

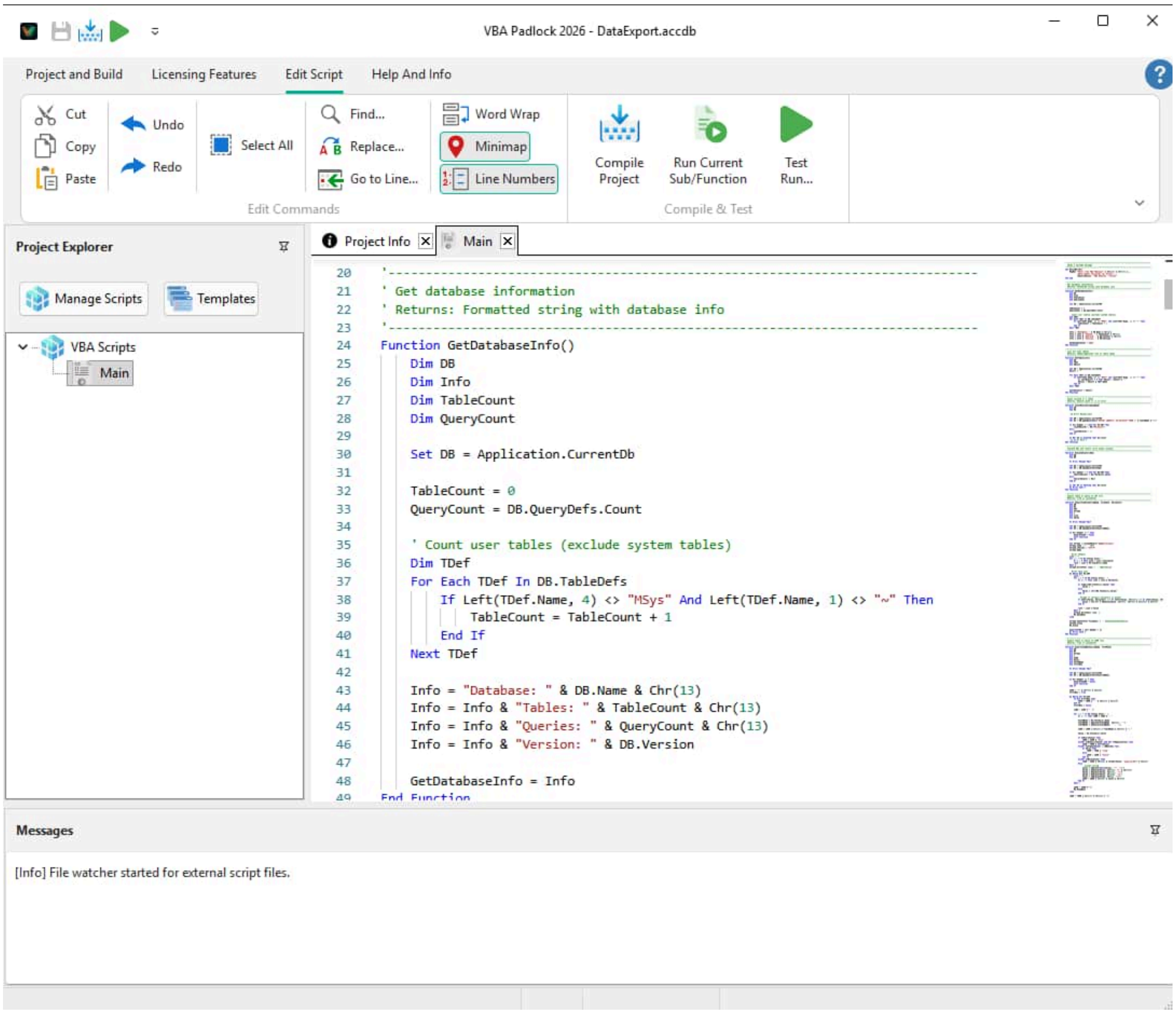
Category	Examples	Count
Layouts	`ppLayoutTitle`, `ppLayoutText`, `ppLayoutBlank`	36
Effects	`ppEffectFade`, `ppEffectPush`, `ppEffectWipe`	150+
Animations	`msoAnimEffectAppear`, `msoAnimEffectFly`	150+
Formats	`ppSaveAsPresentation`, `ppSaveAsPDF`, `ppSaveAsPNG`	40+

Example:

```
References PowerPointConstants

Function BuildSlides()
    Dim Pres
    Set Pres = Application.ActivePresentation
    Dim Slide1
    Set Slide1 = Pres.Slides.Add(1, ppLayoutTitle)
    Slide1.SlideShowTransition.EntryEffect = ppEffectFade
    BuildSlides = Pres.Slides.Count
End Function
```

51.4.5 AccessConstants

Contains ~1,400 constants for automating Microsoft Access. Covers form/report operations, control types, and data transfer.



```
'------------------------------------------------------------------------
' Get database information
' Returns: Formatted string with database info
'------------------------------------------------------------------------
Function GetDatabaseInfo()
    Dim DB
    Dim Info
    Dim TableCount
    Dim QueryCount

    Set DB = Application.CurrentDb

    TableCount = 0
    QueryCount = DB.QueryDefs.Count

    ' Count user tables (exclude system tables)
    Dim TDef
    For Each TDef In DB.TableDefs
        If Left(TDef.Name, 4) <> "MSys" And Left(TDef.Name, 1) <> "~" Then
            TableCount = TableCount + 1
        End If
    Next TDef

    Info = "Database: " & DB.Name & Chr(13)
    Info = Info & "Tables: " & TableCount & Chr(13)
    Info = Info & "Queries: " & QueryCount & Chr(13)
    Info = Info & "Version: " & DB.Version

    GetDatabaseInfo = Info
End Function
```

```
[Info] File watcher started for external script files.
```

Key Categories:

Category	Examples	Count
Objects	`acForm`, `acReport`, `acQuery`, `acTable`	12
Controls	`acTextBox`, `acComboBox`, `acCommandButton`	29
Transfer	`acImport`, `acExport`, `acLink`	3
Commands	`acCmdRunMacro`, `acCmdSave`, `acCmdPrint`	50+

Example:

```
References AccessConstants

Function ExportTableData()
    Application.DoCmd.OpenForm "Customers", acNormal, , , acFormReadOnly
    Application.DoCmd.TransferSpreadsheet acExport, , "Orders",
"C:\Output\Orders.xlsx"
    ExportTableData = "Exported"
End Function
```

51.4.6 OfficeConstants

Contains ~3,350 shared **MSO** (Microsoft Office) constants used across all Office applications. These cover shapes, line styles, fill types, gradients, and more.

Tip

Import `OfficeConstants` alongside the application-specific module (e.g., `ExcelConstants`) when you need to work with shapes or drawing objects.

Key Categories:

Category	Examples	Count
Shapes	`msoShapeRectangle`, `msoShapeOval`, `msoShapeArrow`	183
Fills	`msoFillSolid`, `msoFillGradient`, `msoFillPattern`	6
Gradients	`msoGradientHorizontal`, `msoGradientVertical`	7
Textures	`msoTexturePapyrus`, `msoTextureMarble`	25
3D	`msoMaterialMatte`, `msoMaterialMetal`	35+

Example:

```
References OfficeConstants
References ExcelConstants

Function AddShapes()
    Dim WS
    Set WS = Application.ActiveSheet
    Dim Shp
    Set Shp = WS.Shapes.AddShape(msoShapeRectangle, 50, 50, 200, 100)
    Shp.Fill.OneColorGradient msoGradientHorizontal, 1, 0.5
    Shp.Line.DashStyle = msoLineSolid
    AddShapes = "Shape added"
End Function
```

51.5 Utility Libraries Reference

Helper modules for common programming tasks.

51.5.7 ErrorHandler

Provides 6 helper functions for structured error handling inside compiled scripts.

Function	Signature	Description
`GetErrorInfo`	`GetErrorInfo()`	Returns formatted string with Error details
`FormatError`	`FormatError(Num, [Desc])`	Returns formatted error message
`HasError`	`HasError()`	`True` if `Err.Number <> 0`
`ClearError`	`ClearError()`	Clears the `Err` object

Example:

```
References ErrorHandler

Function SafeDivide(A, B)
    On Error Resume Next
    SafeDivide = A / B
    If HasError() Then
        SafeDivide = "Error: " & GetErrorInfo()
        ClearError
    End If
End Function
```

51.5.8 Financial

Provides 12 financial calculation functions covering loans, investments, and asset depreciation.

Calculations Included:

- **Time Value of Money:** `PMT`, `PV`, `FV`, `NPer`, `Rate`, `IPMT`, `PPMT`
- **Investment Analysis:** `NPV`, `IRR`
- **Asset Depreciation:** `SLN`, `SYD`, `DDB`

Note

Payment functions (`PMT`, `IPMT`, `PPMT`) return **negative values** for cash outflows. For annual rates, divide by 12 for monthly calculations.

Example — Loan payment:

```
References Financial

Function CalculateMortgage(LoanAmount, AnnualRate, Years)
    Dim MonthlyRate, Months
    MonthlyRate = AnnualRate / 12
    Months = Years * 12
    CalculateMortgage = Abs(PMT(MonthlyRate, Months, LoanAmount))
End Function
```

51.6 See Also

Script Functions

Built-in functions for License, Locale, and more.

VBA Compatibility

Supported VBA features and limitations.

Code Editor

Learn how to use the VBA Padlock code editor.

VBA Bridge

Call compiled functions from your Office VBA code.

52. VBA Bridge API Reference

The VBA Bridge is a standard VBA module (`VBADLLBridge`) that VBA Padlock generates and injects into your Office file. It provides **19 public functions** that your VBA code calls to interact with the compiled DLL. These functions handle code execution, licensing, activation, deactivation, online validation, EULA management, and security verification.

Tip

The VBA Bridge is generated automatically by the **Create VBA Bridge** action in VBA Padlock Studio. You do not write these functions yourself. This page documents them so you can call them from your own VBA code.

52.1 Naming Convention

All VBA Bridge functions are prefixed with `VBAPL_`. Internally, each function:

1. Ensures the DLL is loaded and initialized (once per session)
2. Calls the corresponding native DLL export (32-bit or 64-bit, selected automatically)
3. Returns the result to your VBA code

52.2 Code Execution

52.2.1 VBAPL_Execute

Executes a compiled function inside the protected DLL.

```
Public Function VBAPL_Execute(ID As Variant, ParamArray Args() As Variant)
As Variant
```

Parameter	Type	Description
`ID`	`Variant`	Function identifier. Use `"ModuleName|FunctionName"` to call a function in a specific script module, or just `"FunctionName"` to call a function in the default `Main` script. Returns a runtime error if the function is not found.
`Args`	`ParamArray`	Arguments to pass to the compiled function. Any number of arguments is supported.

Returns: The value returned by the compiled function, or `CVErr(xlErrValue)` on error.

Example:

```
' Call a function with no parameters
Dim result As Variant
result = VBAPL_Execute("Main|Initialize")

' Call a function with parameters
Dim total As Variant
total = VBAPL_Execute("Main|CalculatePrice", quantity, unitPrice, taxRate)

' Call a function without specifying the module
Dim msg As Variant
msg = VBAPL_Execute("GetWelcomeMessage", userName)

' Call a function with many parameters (no limit)
Dim report As Variant
report = VBAPL_Execute("Reports|Generate", title, startDate, endDate, format,
outputPath, includeCharts)
```

Note

Internally, calls with 0 to 4 arguments use optimized dedicated DLL exports (`ExecuteVBAFunction0` through `ExecuteVBAFunction4`). Calls with 5 or more arguments are automatically packed into an array and routed through `ExecuteVBAFunctionN`. This is handled transparently — you always call `VBAPL_Execute` the same way regardless of the number of arguments.

52.3 License Status

52.3.2 VBAPL_IsLicenseValid

Checks whether the current license is valid.

```
Public Function VBAPL_IsLicenseValid() As Boolean
```

Returns: `True` if the license is valid or if activation is not required. `False` otherwise.

Example:

```
If Not VBAPL_IsLicenseValid() Then
    MsgBox "Your license is invalid or has expired.", vbExclamation
    VBAPL_ShowActivation
End If
```

52.3.3 VBAPL_HasStoredLicense

Checks whether a license key is stored on this computer.

```
Public Function VBAPL_HasStoredLicense() As Boolean
```

Returns: `True` if a license key has been registered. `False` if no license is stored.

Example:

```
If VBAPL_HasStoredLicense() Then
    MsgBox "Licensed to: " & VBAPL_GetRegisteredName()
Else
    MsgBox "No license found. Please activate.", vbExclamation
End If
```

52.3.4 VBAPL_GetRegisteredName

Returns the registered user name (licensee name) from the stored license.

```
Public Function VBAPL_GetRegisteredName() As String
```

Returns: The registered name string, or an empty string if no license is stored.

Example:

```
Dim licensee As String
licensee = VBAPL_GetRegisteredName()
If licensee <> "" Then
    StatusBar.Text = "Licensed to " & licensee
End If
```

52.3.5 VBAPL_GetHardwareID

Returns the Hardware ID for the current computer. This ID is unique per machine and per project.

```
Public Function VBAPL_GetHardwareID() As String
```

Returns: A formatted Hardware ID string (e.g., `XXXXX-XXXXX-XXXXX`), or an empty string on failure.

Example:

```
Dim hwid As String
hwid = VBAPL_GetHardwareID()
MsgBox "Your Hardware ID is: " & hwid & vbCrLf & _
       "Send this to your vendor to receive a license key.", vbInformation
```

Note

The Hardware ID is **project-specific**. The same computer produces different Hardware IDs for different VBA Padlock projects. This prevents a key generated for one application from being used with another.

52.4 Activation

52.4.6 VBAPL_ShowActivation

Displays the activation dialog where the user can enter a license key.

```
Public Function VBAPL_ShowActivation(Optional ForceDisplay As Boolean =
False) As Long
```

Parameter	Type	Default	Description
`ForceDisplay`	`Boolean`	`False`	When `True`, shows the dialog even if a license is already active.

Return codes:

Value	Meaning
`1`	Activation successful
`0`	User cancelled or activation failed
`-1`	Error (DLL not initialized)
`-2`	Activation is not configured for this project

Example:

```
Dim result As Long
result = VBAPL_ShowActivation()

Select Case result
    Case 1
        MsgBox "Activation successful! Welcome.", vbInformation
    Case 0
        MsgBox "Activation cancelled.", vbExclamation
    Case -2
        ' Activation not configured — this project runs without a license
End Select
```

52.5 Trial Mode

52.5.7 VBAPL_IsTrialMode

Checks whether the application is currently running in trial mode.

```
Public Function VBAPL_IsTrialMode() As Boolean
```

Returns: `True` if in trial mode. `False` if licensed or if trial mode is not configured.

52.5.8 VBAPL_GetTrialLimitLeft

Returns the remaining trial allowance (days, runs, or days until expiration, depending on configuration).

```
Public Function VBAPL_GetTrialLimitLeft() As Long
```

Return codes:

Value	Meaning
`>= 0`	Remaining trial limit
`-1`	DLL not initialized
`-2`	Not in trial mode

Example:

```
If VBAPL_IsTrialMode() Then
    Dim daysLeft As Long
    daysLeft = VBAPL_GetTrialLimitLeft()
    MsgBox "Trial mode: " & daysLeft & " days remaining.", vbInformation
End If
```

52.5.9 VBAPL_ShowTrialNag

Displays the trial nag screen that reminds the user to activate.

```
Public Function VBAPL_ShowTrialNag(Optional ForceDisplay As Boolean =
False) As Long
```

Parameter	Type	Default	Description
`ForceDisplay`	`Boolean`	`False`	When `True`, shows the dialog even if not in trial mode.

Return codes:

Value	Meaning
`0`	User chose to continue in trial mode
`1`	User wants to activate (call `VBAPL_ShowActivation` next)
`2`	User clicked the purchase link (URL opened in browser)
`-1`	Error or not initialized
`-2`	Not in trial mode (and `ForceDisplay = False`)
`-3`	Silent mode is enabled (nag screen disabled in settings)

Example:

```
If VBAPL_IsTrialMode() Then
    Dim nagResult As Long
    nagResult = VBAPL_ShowTrialNag()

    If nagResult = 1 Then
        ' User wants to activate
        VBAPL_ShowActivation
    End If
End If
```

52.6 Deactivation

52.6.10 VBAPL_ShowDeactivation

Displays the deactivation dialog, allowing the user to deactivate their license (e.g., before transferring to a new computer).

```
Public Function VBAPL_ShowDeactivation() As Long
```

Return codes:

Value	Meaning
`0`	License deactivated successfully
`1`	User cancelled
`-1`	DLL not initialized
`-2`	No license to deactivate
`-3`	License already deactivated
`-4`	Deactivation not allowed for this project

Example:

```
Dim result As Long
result = VBAPL_ShowDeactivation()
If result = 0 Then
    MsgBox "License deactivated. You can now activate on another computer.",
vbInformation
End If
```

52.6.11 VBAPL_Deactivate

Deactivates the license programmatically without showing a dialog. Returns a deactivation certificate that the user can send to the vendor as proof of deactivation.

```
Public Function VBAPL_Deactivate(ByRef ResultCode As Long) As String
```

Parameter	Type	Description
`ResultCode`	`Long` (ByRef)	Receives the result code: `0` = success.

Returns: A deactivation certificate string, or an empty string on failure.

Example:

```
Dim resultCode As Long
Dim certificate As String
certificate = VBAPL_Deactivate(resultCode)

If resultCode = 0 Then
    ' Save or display the certificate
    MsgBox "Deactivation certificate:" & vbCrLf & certificate, vbInformation
End If
```

52.6.12 VBAPL_GetDeactivationStatus

Returns the current deactivation status without performing any action.

```
Public Function VBAPL_GetDeactivationStatus() As Long
```

Return codes:

Value	Meaning
`0`	Deactivation allowed (license is active)
`1`	Deactivation allowed (no license stored)
`2`	Deactivation not allowed for this application
`3`	License already deactivated or blacklisted
`-1`	DLL not initialized

52.7 EULA

52.7.13 VBAPL_ShowEULA

Displays the End User License Agreement dialog.

```
Public Function VBAPL_ShowEULA(Optional ForceDisplay As Boolean = False)
As Long
```

Parameter	Type	Default	Description
`ForceDisplay`	`Boolean`	`False`	When `True`, shows the EULA even if already accepted.

Return codes:

Value	Meaning
`1`	EULA accepted
`0`	EULA declined
`-1`	Error
`-2`	EULA not configured for this project

Example:

```
' Show EULA on first launch
Dim eulaResult As Long
eulaResult = VBAPL_ShowEULA()

If eulaResult = 0 Then
    MsgBox "You must accept the license agreement to use this application.",
vbCritical
    ThisWorkbook.Close SaveChanges:=False
End If
```

52.7.14 VBAPL_IsEulaAccepted

Checks whether the EULA has been accepted without displaying the dialog.

```
Public Function VBAPL_IsEulaAccepted() As Boolean
```

Returns: `True` if the EULA has been accepted (or if no EULA is configured). `False` otherwise.

52.7.15 VBAPL_GetEulaAcceptanceDate

Returns the date and time when the EULA was accepted.

```
Public Function VBAPL_GetEulaAcceptanceDate() As String
```

Returns: A date/time string in `YYYY-MM-DD HH:NN:SS` format, or an empty string if the EULA has not been accepted.

Example:

```
If VBAPL_IsEulaAccepted() Then
    Debug.Print "EULA accepted on: " & VBAPL_GetEulaAcceptanceDate()
End If
```

52.8 Project Properties

52.8.16 VBAPL_GetProperty

Returns a project property value that was set in VBA Padlock Studio.

```
Public Function VBAPL_GetProperty(PropName As String, Optional
DefaultValue As Variant = "") As Variant
```

Parameter	Type	Default	Description
`PropName`	`String`	—	Property name (case-insensitive).
`DefaultValue`	`Variant`	`""`	Value returned if the property is not found.

Available properties:

Property Name	Description
`AppTitle`	Application title
`AppVersion`	Version number
`AppCompany`	Company name

Property Name	Description
`AppCopyright`	Copyright notice
`AppDescription`	Application description
`RegisteredName`	Licensee name (same as `VBAPL_GetRegisteredName`)
`LicenseKey`	The stored license key string

Example:

```
Dim appTitle As String
appTitle = VBAPL_GetProperty("AppTitle", "My Application")

Dim version As String
version = VBAPL_GetProperty("AppVersion", "1.0.0.0")

MsgBox appTitle & " v" & version, vbInformation
```

52.9 Online Validation

52.9.17 VBAPL_ValidateOnline

Performs an online license validation against the activation server. This checks that the user's license is still valid and has not been revoked.

```
Public Function VBAPL_ValidateOnline(Optional Silent As Boolean = True) As
Long
```

Parameter	Type	Default	Description
`Silent`	`Boolean`	`True`	When `True`, no UI is shown on failure. When `False`, error dialogs are displayed.

Return codes:

Value	Meaning
`1`	Validation successful
`0`	Validation failed (license revoked or invalid)

Value	Meaning
`-1`	DLL not initialized
`-2`	Online validation not configured
`-3`	No stored license to validate
`-4`	Network error (no internet connection)
`-5`	Server error
`-6`	Security validation failed

Example:

```
Dim result As Long
result = VBAPL_ValidateOnline(False) ' Show errors to the user

Select Case result
    Case 1
        ' License is valid
    Case 0
        MsgBox "Your license has been revoked.", vbCritical
    Case -4
        MsgBox "Could not connect to validation server.", vbExclamation
End Select
```

> **Note**
>
> When Online Validation is enabled, the DLL **automatically** validates the license during initialization based on the configured policy (every startup, daily, weekly, or random). You only need to call `VBAPL_ValidateOnline` manually if you want to trigger an additional on-demand validation.

52.9.18 VBAPL_IsValidationRequired

Checks whether an online validation is due now, based on the configured validation policy.

```
Public Function VBAPL_IsValidationRequired() As Long
```

Return codes:

Value	Meaning
`1`	Validation is required
`0`	Validation not required (skipped based on policy)
`-1`	DLL not initialized
`-2`	Online validation not configured

Example:

```
If VBAPL_IsValidationRequired() = 1 Then
    Dim result As Long
    result = VBAPL_ValidateOnline(False)
    If result <> 1 Then
        MsgBox "License validation failed.", vbCritical
    End If
End If
```

52.10 Security

52.10.19 VBAPL_VerifyDLLSignature

Verifies the Authenticode digital signature of the runtime DLL.

```
Public Function VBAPL_VerifyDLLSignature() As Long
```

Return codes:

Value	Meaning
`0`	Valid signature (signed by G.D.G. Software)
`1`	DLL is not signed
`2`	Signature is invalid or corrupted
`3`	Signed by a different publisher (not G.D.G. Software)
`4`	Signing certificate has expired
`5`	System error during verification

Value	Meaning
`-1`	DLL not initialized

Tip

Signature verification is also performed automatically during DLL initialization. Use this function for additional on-demand verification if your security policy requires it.

Example:

```
Dim sigResult As Long
sigResult = VBAPL_VerifyDLLSignature()

If sigResult <> 0 Then
    MsgBox "WARNING: DLL signature verification failed (code " & sigResult &
")." & vbCrLf & _
           "The DLL may have been tampered with.", vbCritical
    ThisWorkbook.Close SaveChanges:=False
End If
```

52.11 Quick Reference Table

Function	Category	Returns
`VBAPL_Execute`	Execution	`Variant`
`VBAPL_IsLicenseValid`	License	`Boolean`
`VBAPL_HasStoredLicense`	License	`Boolean`
`VBAPL_GetRegisteredName`	License	`String`
`VBAPL_GetHardwareID`	License	`String`
`VBAPL_ShowActivation`	Activation	`Long` (code)
`VBAPL_IsTrialMode`	Trial	`Boolean`
`VBAPL_GetTrialLimitLeft`	Trial	`Long`
`VBAPL_ShowTrialNag`	Trial	`Long` (code)
`VBAPL_ShowDeactivation`	Deactivation	`Long` (code)
`VBAPL_Deactivate`	Deactivation	`String`

Function	Category	Returns
`VBAPL_GetDeactivationStatus`	Deactivation	`Long` (code)
`VBAPL_ShowEULA`	EULA	`Long` (code)
`VBAPL_IsEulaAccepted`	EULA	`Boolean`
`VBAPL_GetEulaAcceptanceDate`	EULA	`String`
`VBAPL_ValidateOnline`	Online Validation	`Long` (code)
`VBAPL_IsValidationRequired`	Online Validation	`Long` (code)
`VBAPL_GetProperty`	Properties	`Variant`
`VBAPL_VerifyDLLSignature`	Security	`Long` (code)

52.12 Typical Initialization Flow

A common pattern for initializing your protected application:

```
Public Sub Workbook_Open()
    ' 1. Show EULA on first launch
    If VBAPL_ShowEULA() = 0 Then
        ThisWorkbook.Close SaveChanges:=False
        Exit Sub
    End If

    ' 2. Check license
    If Not VBAPL_IsLicenseValid() Then
        If VBAPL_IsTrialMode() Then
            ' Show trial nag screen
            Dim nagResult As Long
            nagResult = VBAPL_ShowTrialNag()
            If nagResult = 1 Then
                VBAPL_ShowActivation
            End If
        Else
            ' No trial — require activation
            If VBAPL_ShowActivation() <> 1 Then
                MsgBox "Activation required.", vbCritical
                ThisWorkbook.Close SaveChanges:=False
                Exit Sub
            End If
        End If
    End If

    ' 3. Run the main application logic
    VBAPL_Execute "Main|Initialize"
End Sub
```

52.13 See Also

- DLL Export Reference — Low-level DLL function signatures for advanced `Declare` usage
- License Return Codes — Complete list of all error and return codes
- Script Functions Reference — Functions available inside compiled scripts
- Protect an Excel Workbook — Step-by-step tutorial using the VBA Bridge

53. DLL Export Reference

The VBA Padlock runtime DLL exports a set of `stdcall` functions that can be called directly from any language that supports Windows DLL calls. This page documents every exported function with its native signature.

> **Note**
>
> Most developers should use the VBA Bridge API instead of calling these exports directly. The VBA Bridge wraps these functions with error handling, automatic DLL loading, and 32/64-bit detection. This page is intended for advanced scenarios such as calling VBA Padlock from C#, Delphi, Python, or other languages.

53.1 DLL Files

VBA Padlock produces two runtime DLLs per project, one for each Office architecture. The DLL names are derived from your project name using the pattern `{ProjectName}run32.dll` and `{ProjectName}run64.dll`:

File	Architecture	Usage
`[YourProject]run32.dll`	32-bit (x86)	Used with 32-bit Office
`[YourProject]run64.dll`	64-bit (x64)	Used with 64-bit Office

For example, if your project is named `MyApp`, the runtime DLLs are `MyApprun32.dll` and `MyApprun64.dll`.

Both DLLs export the same functions with the same `stdcall` calling convention.

53.2 Initialization

Before calling any other function, the DLL must be initialized by executing a compiled function (which triggers internal setup). The VBA Bridge handles this automatically via `VBAPL_Execute`.

53.3 EULA Functions

53.3.1 VBAPLShowEULA

Displays the EULA dialog.

```
function VBAPLShowEULA(ForceDisplay: Integer): Integer; stdcall;
```

```
' 32-bit
Private Declare Function VBAPLShowEULA Lib "MyApprun32.dll" _
    (ByVal ForceDisplay As Long) As Long

' 64-bit
Private Declare PtrSafe Function VBAPLShowEULA Lib "MyApprun64.dll" _
    (ByVal ForceDisplay As Long) As Long
```

```
[DllImport("MyApprun32.dll", CallingConvention =
CallingConvention.StdCall)]
static extern int VBAPLShowEULA(int ForceDisplay);
```

Parameter	Type	Description
`ForceDisplay`	`Integer`	`1` = force display even if already accepted. `0` = normal behavior.

Return	Meaning
`1`	Accepted
`0`	Declined or cancelled
`-1`	Error showing dialog
`-2`	EULA not configured

53.3.2 VBAPLIsEulaAccepted

Checks if the EULA has been accepted.

```
function VBAPLIsEulaAccepted: Integer; stdcall;
```

Return	Meaning
`1`	Accepted
`0`	Not accepted
`-1`	Not configured

53.3.3 VBAPLGetEulaAcceptanceDate

Returns the EULA acceptance date as an ISO string.

```
function VBAPLGetEulaAcceptanceDate(out OutDate: WideString): Integer;
stdcall;
```

Parameter	Type	Direction	Description
`OutDate`	`WideString`	Out	Receives the date in `YYYY-MM-DD HH:NN:SS` format.

Return	Meaning
`0`	Success
`-1`	Not configured or not accepted

53.4 License Status Functions

53.4.4 VBAPLIsLicenseValid

Checks if the current license is valid.

```
function VBAPLIsLicenseValid: Integer; stdcall;
```

Return	Meaning
`1`	Valid (or activation not required)
`0`	Invalid
`-1`	Not initialized

53.4.5 VBAPLGetHardwareID

Returns the project-specific Hardware ID for this machine.

```
function VBAPLGetHardwareID(out OutHardwareID: WideString): Integer;
stdcall;
```

Parameter	Type	Direction	Description
`OutHardwareID`	`WideString`	Out	Receives the Hardware ID string.

Return	Meaning
`0`	Success
`-1`	Not initialized
`-2`	Error generating Hardware ID

53.4.6 VBAPLShowActivation

Shows the activation dialog.

```
function VBAPLShowActivation(ForceDisplay: Integer): Integer; stdcall;
```

Parameter	Type	Description
`ForceDisplay`	`Integer`	`1` = force display, `0` = normal.

Return	Meaning
`1`	Activation successful
`0`	Cancelled or failed
`-1`	Not initialized
`-2`	Activation not configured

53.5 Deactivation Functions

53.5.7 VBAPLDeactivate

Deactivates the license programmatically (no GUI).

```
function VBAPLDeactivate(out OutCertificate: WideString): Integer;
stdcall;
```

Parameter	Type	Direction	Description
`OutCertificate`	`WideString`	Out	Receives the deactivation certificate string.

Return	Meaning
`0`	Success
`-1`	Not initialized
`-2`	No license stored
`-3`	Already deactivated
`-4`	Deactivation not allowed
`-5`	Internal error

53.5.8 VBAPLGetDeactivationInfo

Returns the deactivation status.

```
function VBAPLGetDeactivationInfo: Integer; stdcall;
```

Return	Meaning
`0`	Allowed (license active)
`1`	Allowed (no license stored)
`2`	Not allowed
`3`	Already deactivated
`-1`	Not initialized

53.5.9 VBAPLDeactivateGUI

Shows a deactivation dialog with certificate display.

```
function VBAPLDeactivateGUI: Integer; stdcall;
```

Return	Meaning
`0`	Success
`1`	Cancelled by user
`-1`	Not initialized
`-2`	No license stored
`-3`	Already deactivated
`-4`	Not allowed
`-5`	Internal error

53.6 Trial and License Info Functions

53.6.10 VBAPLIsTrialMode

Checks if running in trial mode.

```
function VBAPLIsTrialMode: Integer; stdcall;
```

Return	Meaning
`1`	Trial mode
`0`	Not trial mode
`-1`	Not initialized

53.6.11 VBAPLGetTrialLimitLeft

Returns the remaining trial limit.

```
function VBAPLGetTrialLimitLeft: Integer; stdcall;
```

Return	Meaning
`>= 0`	Remaining limit (days, runs, or days until expiration)
`-1`	Not initialized
`-2`	Not in trial mode

53.6.12 VBAPLHasStoredLicense

Checks if a license key is stored.

```
function VBAPLHasStoredLicense: Integer; stdcall;
```

Return	Meaning
`1`	License stored
`0`	No license
`-1`	Not initialized

53.6.13 VBAPLGetRegisteredName

Returns the registered user name.

```
function VBAPLGetRegisteredName(out OutName: WideString): Integer;
stdcall;
```

Parameter	Type	Direction	Description
`OutName`	`WideString`	Out	Receives the registered name.

Return	Meaning
`0`	Success
`-1`	Not initialized
`-2`	No license stored

53.6.14 VBAPLShowTrialNag

Shows the trial nag screen.

```
function VBAPLShowTrialNag(ForceDisplay: Integer): Integer; stdcall;
```

Parameter	Type	Description
`ForceDisplay`	`Integer`	`1` = force display, `0` = normal.

Return	Meaning
`0`	Continue trial
`1`	Activate
`2`	Purchase (URL opened)
`-1`	Not initialized
`-2`	Not in trial mode
`-3`	Silent mode enabled

53.7 Project Properties

53.7.15 VBAPLGetProperty

Returns a project property value.

```
function VBAPLGetProperty(const PropName: OleVariant; const DefValue:
OleVariant): OleVariant; stdcall;
```

Parameter	Type	Description
`PropName`	`OleVariant`	Property name (case-insensitive): `AppTitle`, `AppVersion`, `AppCompany`, `AppCopyright`, `AppDescription`, `RegisteredName`, `LicenseKey`.
`DefValue`	`OleVariant`	Default value returned if the property is not found.

Returns: The property value, or `DefValue` if not found.

53.8 DLL Signature Verification

53.8.16 VBAPLVerifyDLLSignature

Verifies the Authenticode signature of the runtime DLL.

```
function VBAPLVerifyDLLSignature: Integer; stdcall;
```

Return	Meaning
`0`	Valid (signed by G.D.G. Software)
`1`	Not signed
`2`	Invalid or corrupted signature
`3`	Wrong publisher
`4`	Certificate expired
`5`	System error

53.9 Online Validation Functions

53.9.17 VBAPLValidateOnline

Performs an online license validation against the activation server.

```
function VBAPLValidateOnline(Silent: Integer = 1): Integer; stdcall;
```

Parameter	Type	Default	Description
`Silent`	`Integer`	`1`	`0` = show error dialogs, `1` = silent mode.

Return	Meaning
`1`	Validation successful
`0`	Failed or license revoked
`-1`	Not initialized
`-2`	Online validation not configured
`-3`	No license stored

Return	Meaning
`-4`	Network error
`-5`	Server error
`-6`	Security validation failed

53.9.18 VBAPLIsValidationRequired

Checks if an online validation is due (based on the configured interval).

```
function VBAPLIsValidationRequired: Integer; stdcall;
```

Return	Meaning
`1`	Validation required now
`0`	Not required yet
`-1`	Not initialized
`-2`	Online validation not configured

53.10 Localization Functions

53.10.19 VBAPLLoadLocale

Loads localization strings from a JSON string. This overrides the built-in English strings used in activation dialogs, trial nag screens, and other UI elements.

```
function VBAPLLoadLocale(const JSONContent: WideString): Integer; stdcall;
```

Parameter	Type	Description
`JSONContent`	`WideString`	A JSON object containing key-value pairs for localized strings.

Return	Meaning
`1`	Success
`0`	Failed (invalid JSON)

Return	Meaning
`-1`	Empty content

53.10.20 VBAPLGetLocaleInfo

Returns information about the currently loaded locale.

```
function VBAPLGetLocaleInfo(out OutLocale: WideString): Integer; stdcall;
```

Parameter	Type	Direction	Description
`OutLocale`	`WideString`	Out	Receives locale information.

Return	Meaning
`1`	Custom locale loaded
`0`	Using default locale

53.11 Complete Export List

#	Function	Category
1	`VBAPLShowEULA`	EULA
2	`VBAPLIsEulaAccepted`	EULA
3	`VBAPLGetEulaAcceptanceDate`	EULA
4	`VBAPLIsLicenseValid`	License Status
5	`VBAPLGetHardwareID`	License Status
6	`VBAPLShowActivation`	License Status
7	`VBAPLDeactivate`	Deactivation
8	`VBAPLGetDeactivationInfo`	Deactivation
9	`VBAPLDeactivateGUI`	Deactivation
10	`VBAPLIsTrialMode`	Trial
11	`VBAPLGetTrialLimitLeft`	Trial
12	`VBAPLHasStoredLicense`	License Info
13	`VBAPLGetRegisteredName`	License Info

#	Function	Category
14	`VBAPLShowTrialNag`	Trial
15	`VBAPLGetProperty`	Properties
16	`VBAPLVerifyDLLSignature`	Security
17	`VBAPLValidateOnline`	Online Validation
18	`VBAPLIsValidationRequired`	Online Validation
19	`VBAPLLoadLocale`	Localization
20	`VBAPLGetLocaleInfo`	Localization

53.12 See Also

- VBA Bridge API Reference — High-level VBA wrapper functions (recommended for most users)
- License Return Codes — All return codes and error codes explained
- Script Functions Reference — Functions available inside compiled scripts

54. License Return Codes

This page documents every return code, error code, and enumeration value used by the VBA Padlock licensing system. Use this as a reference when handling results from the VBA Bridge API or DLL exports.

54.1 License Error Codes

These codes are returned when decoding or validating a license key.

Code	Constant	Description
`0`	`LICENSE_OK`	License is valid
`-1`	`LICENSE_ERR_INVALID_CHECKSUM`	The license key checksum does not match (corrupted or truncated key)
`-2`	`LICENSE_ERR_INVALID_FORMAT`	The license key format is not recognized
`-3`	`LICENSE_ERR_INVALID_VERSION`	The license key was generated with an unsupported format version
`-4`	`LICENSE_ERR_INVALID_SIGNATURE`	The cryptographic signature is invalid (the key was not generated with the correct private key)
`-5`	`LICENSE_ERR_DECOMPRESSION`	Internal decompression of the license payload failed
`-6`	`LICENSE_ERR_EXPIRED`	The license has expired (past the expiration date)
`-7`	`LICENSE_ERR_MAX_RUNS`	The maximum number of executions has been reached
`-8`	`LICENSE_ERR_HARDWARE_MISMATCH`	The license is bound to a different Hardware ID
`-9`	`LICENSE_ERR_WRONG_APP`	The license was generated for a different VBA Padlock project
`-10`	`LICENSE_ERR_DECODE_FAILED`	General decoding failure (invalid license string)
`-11`	`LICENSE_ERR_EMPTY_KEY`	No license key was provided (empty string)

Tip

The most common errors your users will encounter are `-8` (hardware mismatch, if the user changed computers) and `-6` (expired). Design your error messages to guide users toward the appropriate resolution for each case.

54.2 Activation Return Codes

Returned by `VBAPL_ShowActivation` / `VBAPLShowActivation`.

Code	Meaning	Suggested Action
`1`	Activation successful	Proceed normally
`0`	Cancelled or failed	Prompt user to try again or exit

Code	Meaning	Suggested Action
`-1`	Not initialized (DLL error)	Check DLL file paths
`-2`	Activation not configured	No action needed (project does not require activation)

54.3 Deactivation Return Codes

Returned by `VBAPL_ShowDeactivation` / `VBAPLDeactivateGUI`.

Code	Meaning	Suggested Action
`0`	License deactivated successfully	Inform user they can activate on a new computer
`1`	Cancelled by user	No action needed
`-1`	Not initialized	Check DLL initialization
`-2`	No license to deactivate	User has not activated yet
`-3`	Already deactivated	License was already removed
`-4`	Deactivation not allowed	This project does not permit deactivation
`-5`	Internal error	Contact support

The same codes apply to `VBAPL_Deactivate` / `VBAPLDeactivate` (programmatic deactivation), except there is no `1` (cancelled) since no dialog is shown.

54.4 Deactivation Status Codes

Returned by `VBAPL_GetDeactivationStatus` / `VBAPLGetDeactivationInfo`.

Code	Meaning
`0`	Deactivation is allowed (license is active)
`1`	Deactivation is allowed (but no license is stored)
`2`	Deactivation is not allowed for this application
`3`	License already deactivated or blacklisted
`-1`	Not initialized

54.5 Trial Nag Return Codes

Returned by `VBAPL_ShowTrialNag` / `VBAPLShowTrialNag`.

Code	Meaning	Suggested Action
`0`	User chose to continue trial	Proceed normally (limited functionality)
`1`	User wants to activate	Call `VBAPL_ShowActivation`
`2`	User clicked purchase link	URL opened in browser; user continues in trial
`-1`	Error or not initialized	Check DLL initialization
`-2`	Not in trial mode	No action needed
`-3`	Silent mode enabled	Nag screen is disabled in project settings

54.6 EULA Return Codes

Returned by `VBAPL_ShowEULA` / `VBAPLShowEULA`.

Code	Meaning	Suggested Action
`1`	EULA accepted	Proceed normally
`0`	EULA declined	Close the application
`-1`	Error	Check DLL initialization
`-2`	EULA not configured	No action needed (project has no EULA)

54.7 DLL Signature Verification Codes

Returned by `VBAPL_VerifyDLLSignature` / `VBAPLVerifyDLLSignature`.

Code	Meaning	Suggested Action
`0`	Valid signature (G.D.G. Software)	DLL is authentic
`1`	DLL is not signed	DLL may have been tampered with
`2`	Invalid or corrupted signature	DLL has been modified
`3`	Signed by wrong publisher	DLL is not from G.D.G. Software
`4`	Signing certificate expired	Contact vendor for updated DLLs
`5`	System error during verification	Windows API error
`-1`	Not initialized	Call after DLL initialization

54.8 Online Validation Return Codes

Returned by `VBAPL_ValidateOnline` / `VBAPLValidateOnline`.

Code	Meaning	Suggested Action
`1`	Validation successful	License is valid on the server
`0`	Failed or license revoked	Disable the application
`-1`	Not initialized	Check DLL initialization
`-2`	Online validation not configured	No action needed
`-3`	No license stored	User must activate first
`-4`	Network error	Retry later; optionally allow offline grace period
`-5`	Server error	Server-side issue; retry later
`-6`	Security validation failed	Server response could not be verified

Returned by `VBAPLIsValidationRequired`:

Code	Meaning
`1`	Validation is required now
`0`	Not required yet (within the configured interval)
`-1`	Not initialized
`-2`	Online validation not configured

54.9 License Types

Used in the license key payload. Visible through `GetLicenseProperty` or decoded from the key.

Code	Type	Description
`0`	Full	Permanent license with no restrictions
`1`	Trial	Time-limited or run-limited trial
`2`	Subscription	Recurring license requiring periodic validation
`3`	Time-Limited	License that expires on a specific date

54.10 Edition Codes

Code	Edition	Description
`0`	Standard	Standard edition
`1`	Professional	Professional edition
`2`	Enterprise	Enterprise edition
`3`	Custom	Custom edition (uses the `CustomEditionName` field)

Your compiled scripts can check the edition to enable or disable features:

```
Dim edition As String
edition = GetLicenseProperty("edition", "Standard")

Select Case edition
    Case "Professional", "Enterprise"
        ' Enable advanced features
        EnableAdvancedMode
    Case Else
        ' Standard features only
End Select
```

54.11 Trial Limit Types

Code	Constant	Description
`0`	`tltDays`	Trial measured in days since first launch
`1`	`tltRuns`	Trial measured in number of executions
`2`	`tltExpiration`	Trial expires on a fixed date

54.12 Handling Errors in VBA

Here is a pattern for comprehensive error handling using the return codes:

```
Public Sub StartApplication()
    ' 1. Show EULA
    Dim eulaResult As Long
    eulaResult = VBAPL_ShowEULA()
    Select Case eulaResult
        Case 0
            MsgBox "You must accept the license agreement.", vbCritical
            ThisWorkbook.Close False
            Exit Sub
        Case -1
            MsgBox "Initialization error.", vbCritical
            Exit Sub
        Case -2, 1
            ' EULA accepted or not configured — continue
    End Select

    ' 2. Check license
    If Not VBAPL_IsLicenseValid() Then
        If VBAPL_IsTrialMode() Then
            Dim trialLeft As Long
            trialLeft = VBAPL_GetTrialLimitLeft()
            If trialLeft = 0 Then
                MsgBox "Your trial has expired.", vbExclamation
                VBAPL_ShowActivation
            Else
                VBAPL_ShowTrialNag
            End If
        Else
            Dim activResult As Long
            activResult = VBAPL_ShowActivation()
            If activResult <> 1 Then
                MsgBox "Activation is required to use this application.",
vbCritical
                ThisWorkbook.Close False
                Exit Sub
            End If
        End If
    End If

    ' 3. Online validation (if configured)
```

```
        If VBAPL_IsLicenseValid() Then
            Dim valResult As Long
            valResult = VBAPL_Execute("Main|CheckOnlineValidation")
        End If

        ' 4. Run main application
        VBAPL_Execute "Main|Initialize"
    End Sub
```

54.13 See Also

- VBA Bridge API Reference — All 20 VBAPL_* functions
- DLL Export Reference — Raw DLL function signatures
- Script Functions Reference — Functions available inside compiled scripts
- Troubleshooting — Common issues and solutions

55. Project Format Reference

Every VBA Padlock project is stored as a **directory next to your Office file**. Understanding this layout makes backup, version control, CI builds, and troubleshooting much easier.

55.1 At a Glance

Project folder

`<OfficeFileName>.vbapadlock` next to your `.xlsm`, `.docm`, `.accdb`, or `.pptm` file.

Core source

`Main.bas` + additional `.bas` modules are your protected script source files.

Build output

Final binaries are generated into a sibling `bin/` directory.

55.2 How VBA Padlock Resolves a Project

When you open an Office file, VBA Padlock automatically looks for a folder with the same base name and `.vbapadlock` suffix.

1. Open `MyWorkbook.xlsm` in VBA Padlock Studio.
2. VBA Padlock checks for `MyWorkbook.vbapadlock/` in the same folder.
3. If missing, it creates the folder and initializes project files.

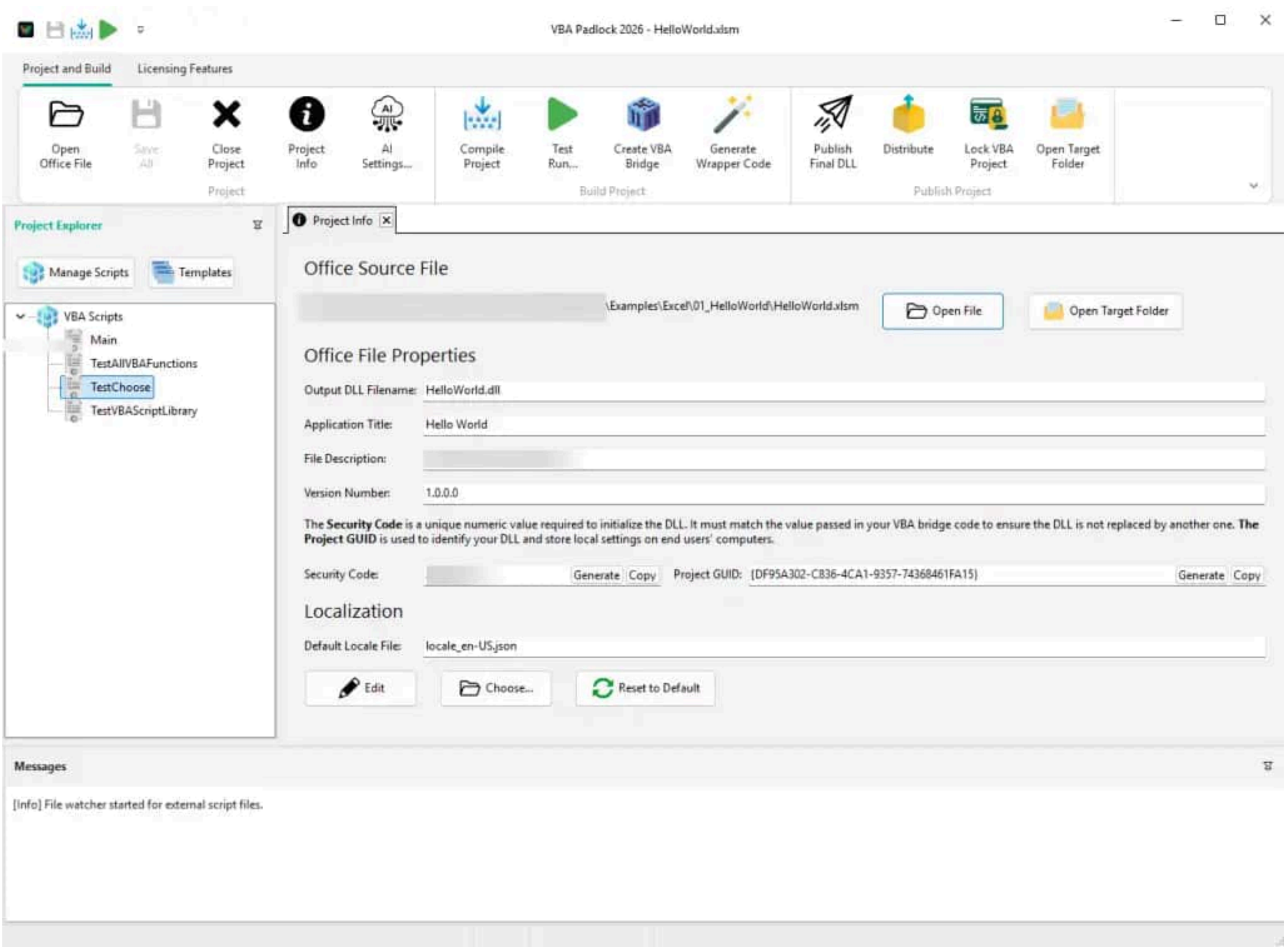
Office File	Expected Project Directory
`MyWorkbook.xlsm`	`MyWorkbook.vbapadlock\`
`Report.docm`	`Report.vbapadlock\`
`Database.accdb`	`Database.vbapadlock\`
`Presentation.pptm`	`Presentation.vbapadlock\`

⚠ Path rule

The `.vbapadlock` folder **must stay in the same directory** as its Office file. If separated, VBA Padlock cannot resolve the project correctly.

55.3 Standard Project Directory Content

```
▼ MyWorkbook.vbapadlock/
      project.xml     Project settings (managed by VBA Padlock Studio)
      Main.bas     Main script module (required)
      Module2.bas     Additional script module (optional)
      Helpers.bas     Additional script module (optional)
      mainicon.ico     Custom DLL icon (optional)
```



Office Source File

\Examples\Excel\01_HelloWorld\HelloWorld.xlsm

Office File Properties

Output DLL Filename: HelloWorld.dll

Application Title: Hello World

File Description:

Version Number: 1.0.0.0

The **Security Code** is a unique numeric value required to initialize the DLL. It must match the value passed in your VBA bridge code to ensure the DLL is not replaced by another one. **The Project GUID** is used to identify your DLL and store local settings on end users' computers.

Security Code:

Project GUID: {DF95A302-C836-4CA1-9357-74368461FA15}

Localization

Default Locale File: locale_en-US.json

[Info] File watcher started for external script files.

55.3.1 `project.xml`

`project.xml` is the central settings file for your project (managed internally by VBA Padlock Studio). It stores:

- Application metadata (title, version, company, description)
- Compilation and publishing settings
- Licensing configuration (activation/deactivation/trial/EULA/online validation)
- Hardware ID and key generator parameters

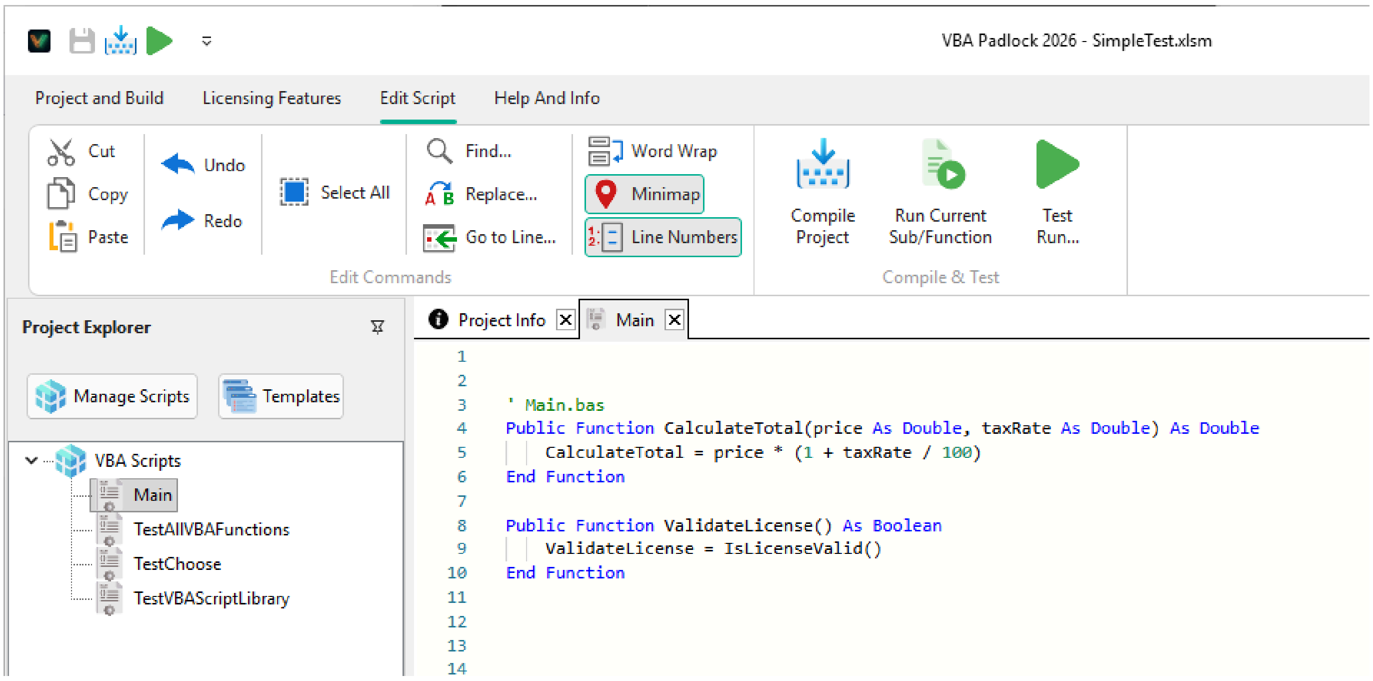
⚠ **Do not edit manually**

Always update settings from the Studio UI. Manual edits can desynchronize configuration and cause build/runtime issues.

55.3.2 Script modules (`.bas`)

Each `.bas` file corresponds to a VBA module that will be compiled into your protected DLL.

File	Role	Description
`Main.bas`	`main`	Required entry module for compiled code execution.
`*.bas`	custom	Optional additional modules (helpers, business logic, APIs).



```vba


' Main.bas
Public Function CalculateTotal(price As Double, taxRate As Double) As Double
    CalculateTotal = price * (1 + taxRate / 100)
End Function

Public Function ValidateLicense() As Boolean
    ValidateLicense = IsLicenseValid()
End Function
```

Recommended format:

- Encoding: **UTF-8**
- Line endings: CRLF or LF (both supported)
- Content: Standard VBA + built-ins from script functions

Example `Main.bas`:

```
' Main module - entry point for the compiled application
Public Function Initialize() As String
    Initialize = "Application initialized successfully"
End Function

Public Function CalculatePrice(quantity As Long, unitPrice As Double, taxRate
As Double) As Double
    CalculatePrice = quantity * unitPrice * (1 + taxRate)
End Function
```

55.3.3 `mainicon.ico` (optional)

If present, `mainicon.ico` is embedded into generated DLL resources.

- Format: Windows ICO
- Recommended multi-size icon: 16, 32, 48, 256 px

55.4 Project Format Variants

Format	Storage	Status
External (current)	Individual `.bas` files in `.vbapadlock`	Recommended for all new projects
Legacy	Inline scripts in `.vbaplproj`	Supported for backward compatibility

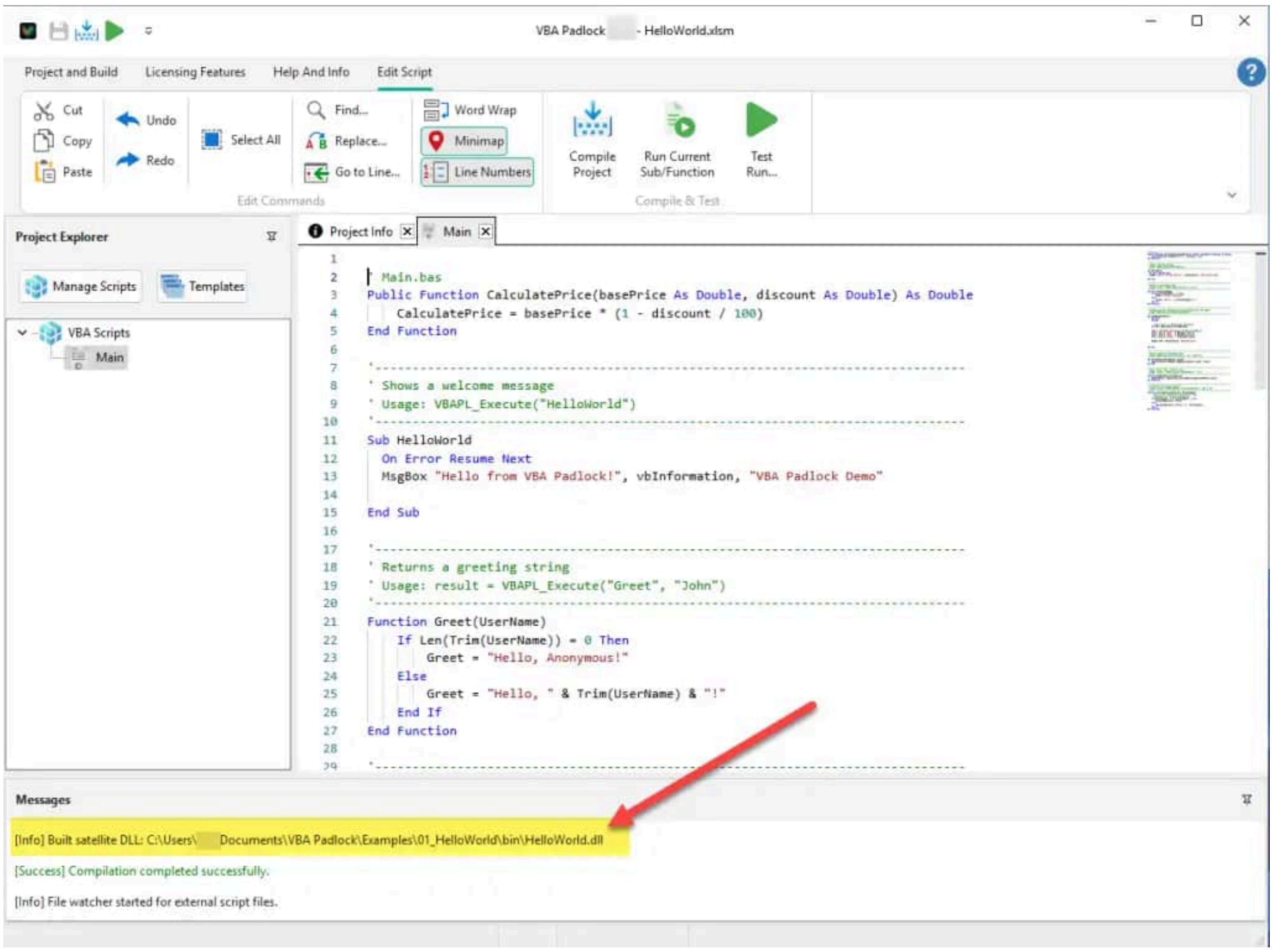
Legacy migration

Older projects are detected automatically and can be migrated to the external format through VBA Padlock Studio.

55.5 Build Output Layout (`bin/`)

After compilation/publish, VBA Padlock creates runtime artifacts in `bin/` next to the Office file:

```
    MyWorkbook.xlsm    Protected Office file (contains VBA Bridge)
  ▼ bin/
        MyWorkbookrun32.dll    32-bit runtime (for 32-bit Office)
        MyWorkbookrun64.dll    64-bit runtime (for 64-bit Office)
        MyWorkbook.dll    Satellite DLL (compiled bytecode)
```



```
' Main.bas
Public Function CalculatePrice(basePrice As Double, discount As Double) As Double
    CalculatePrice = basePrice * (1 - discount / 100)
End Function

'------------------------------------------------------------------------------
' Shows a welcome message
' Usage: VBAPL_Execute("HelloWorld")
'------------------------------------------------------------------------------
Sub HelloWorld
  On Error Resume Next
  MsgBox "Hello from VBA Padlock!", vbInformation, "VBA Padlock Demo"

End Sub

'------------------------------------------------------------------------------
' Returns a greeting string
' Usage: result = VBAPL_Execute("Greet", "John")
'------------------------------------------------------------------------------
Function Greet(UserName)
    If Len(Trim(UserName)) = 0 Then
        Greet = "Hello, Anonymous!"
    Else
        Greet = "Hello, " & Trim(UserName) & "!"
    End If
End Function

'------------------------------------------------------------------------------
```

```
[Info] Built satellite DLL: C:\Users\[redacted]Documents\VBA Padlock\Examples\01_HelloWorld\bin\HelloWorld.dll
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

File	Description	Required for deployment
`MyWorkbookrun32.dll`	32-bit runtime loader	Yes
`MyWorkbookrun64.dll`	64-bit runtime loader	Yes
`MyWorkbook.dll`	Satellite DLL containing protected bytecode	Yes

> ⚠ **Deployment integrity**
>
> Distribute the Office file and the complete `bin/` content together. Missing one DLL will break execution.

55.6 Complete Example Layout

```
▼ C:\Projects\MyApp\
      MyWorkbook.xlsm    Office file with VBA Bridge injected
    ▼ MyWorkbook.vbapadlock/
          project.xml    All project settings
          Main.bas    Main compiled module
          Utilities.bas    Helper functions module
          mainicon.ico    Custom DLL icon
    ▼ bin/
          MyWorkbookrun32.dll    32-bit runtime
          MyWorkbookrun64.dll    64-bit runtime
          MyWorkbook.dll    Satellite DLL
```

55.7 Supported Office File Types

Extension	Application	Description
`.xlsm`	Excel	Macro-enabled workbook
`.xlsb`	Excel	Binary workbook (macro-enabled)
`.xla` / `.xlam`	Excel	Add-in
`.docm`	Word	Macro-enabled document
`.dotm`	Word	Macro-enabled template
`.accdb`	Access	Database
`.accde`	Access	Execute-only database
`.pptm`	PowerPoint	Macro-enabled presentation
`.ppam`	PowerPoint	Add-in

55.8 Version Control Guidelines

Track source and configuration, ignore generated artifacts.

Include in repository:

- Office file (`.xlsm`, `.docm`, etc.)
- `.vbapadlock/` directory (`project.xml`, `.bas` modules, optional `mainicon.ico`)

Exclude from repository:

- `bin/` output folder
- Temporary `.vbaplcode` files

```
# VBA Padlock build output
bin/
*.vbaplcode
```

55.9 See Also

Creating Your First Project

Step-by-step setup from opening a document to first compilation. Open guide →

Batch Compilation & CI/CD

Automate builds with command-line switches and pipelines. Open guide →

Distribution

Package and ship your Office file with all required runtime files. Open reference →

56. Troubleshooting

If something is not working as expected, start here. This page groups the most common VBA Padlock issues by area, with practical fixes you can apply immediately.

Compilation →

Errors during build or runtime mismatches after a successful compile.

Bridge & DLL Loading →

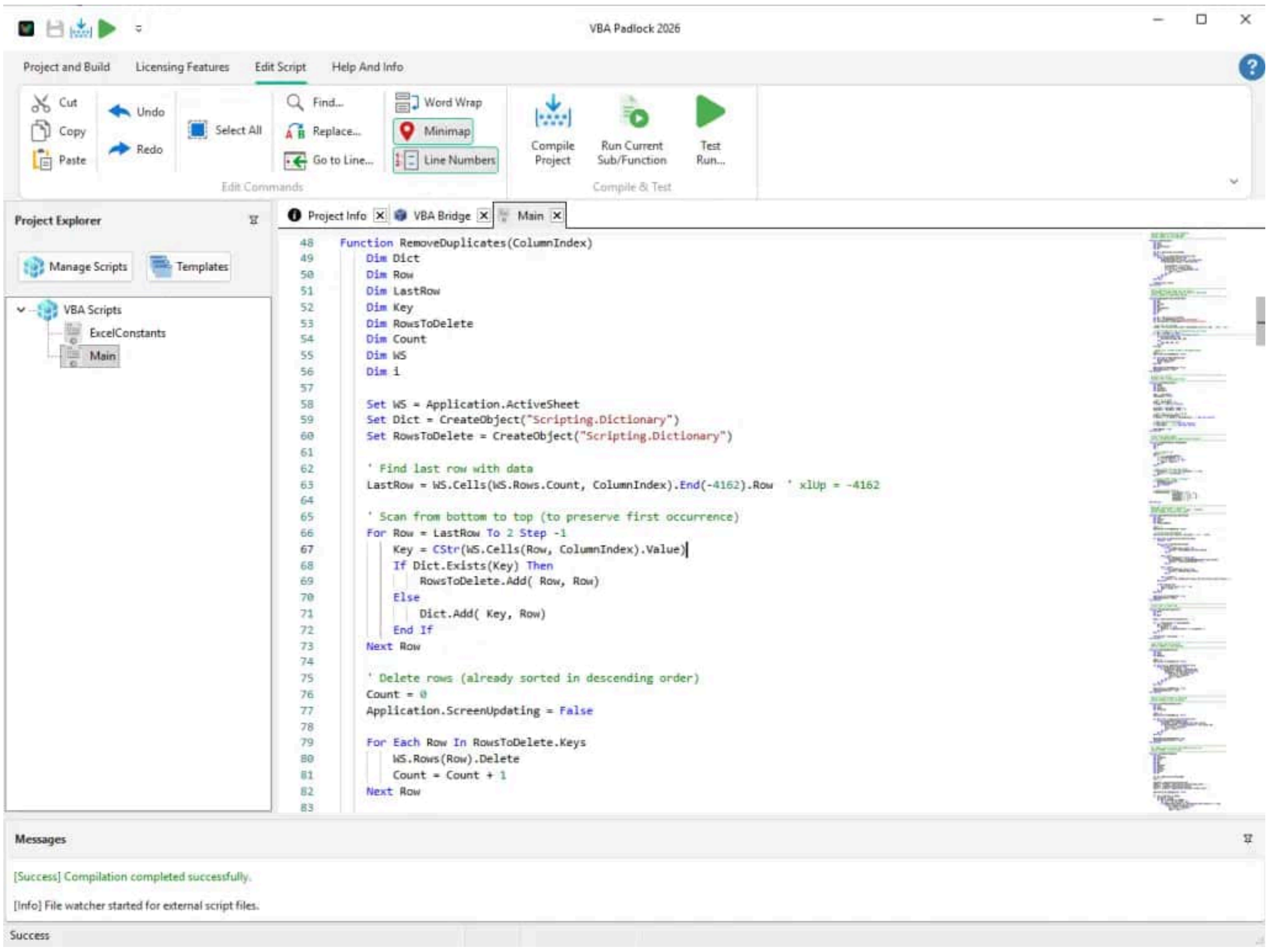
Injection problems, missing files, signature/security-code mismatches.

Licensing →

Trial, key validation, Hardware ID, and online activation issues.

56.1 Quick Diagnosis Flow

1. **Compile first** with `F5` and fix every error in the Messages panel.
2. **Validate scripts in Test Runner** before testing from Office.
3. **Re-inject VBA Bridge** after any Security Code / project changes.
4. **Check `bin/` deployment** (all expected DLLs present, not blocked).
5. **If licensing is involved**, verify activation settings and key/project match.



```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Dim Row
    Dim LastRow
    Dim Key
    Dim RowsToDelete
    Dim Count
    Dim WS
    Dim i

    Set WS = Application.ActiveSheet
    Set Dict = CreateObject("Scripting.Dictionary")
    Set RowsToDelete = CreateObject("Scripting.Dictionary")

    ' Find last row with data
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp = -4162

    ' Scan from bottom to top (to preserve first occurrence)
    For Row = LastRow To 2 Step -1
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Dict.Exists(Key) Then
            RowsToDelete.Add( Row, Row)
        Else
            Dict.Add( Key, Row)
        End If
    Next Row

    ' Delete rows (already sorted in descending order)
    Count = 0
    Application.ScreenUpdating = False

    For Each Row In RowsToDelete.Keys
        WS.Rows(Row).Delete
        Count = Count + 1
    Next Row
```

```
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
Success
```

Fastest way to isolate issues

When possible, test in this order: **Compile → Test Runner → Office macro call**. This quickly tells you whether the issue is in script logic, bridge integration, or Office environment/configuration.

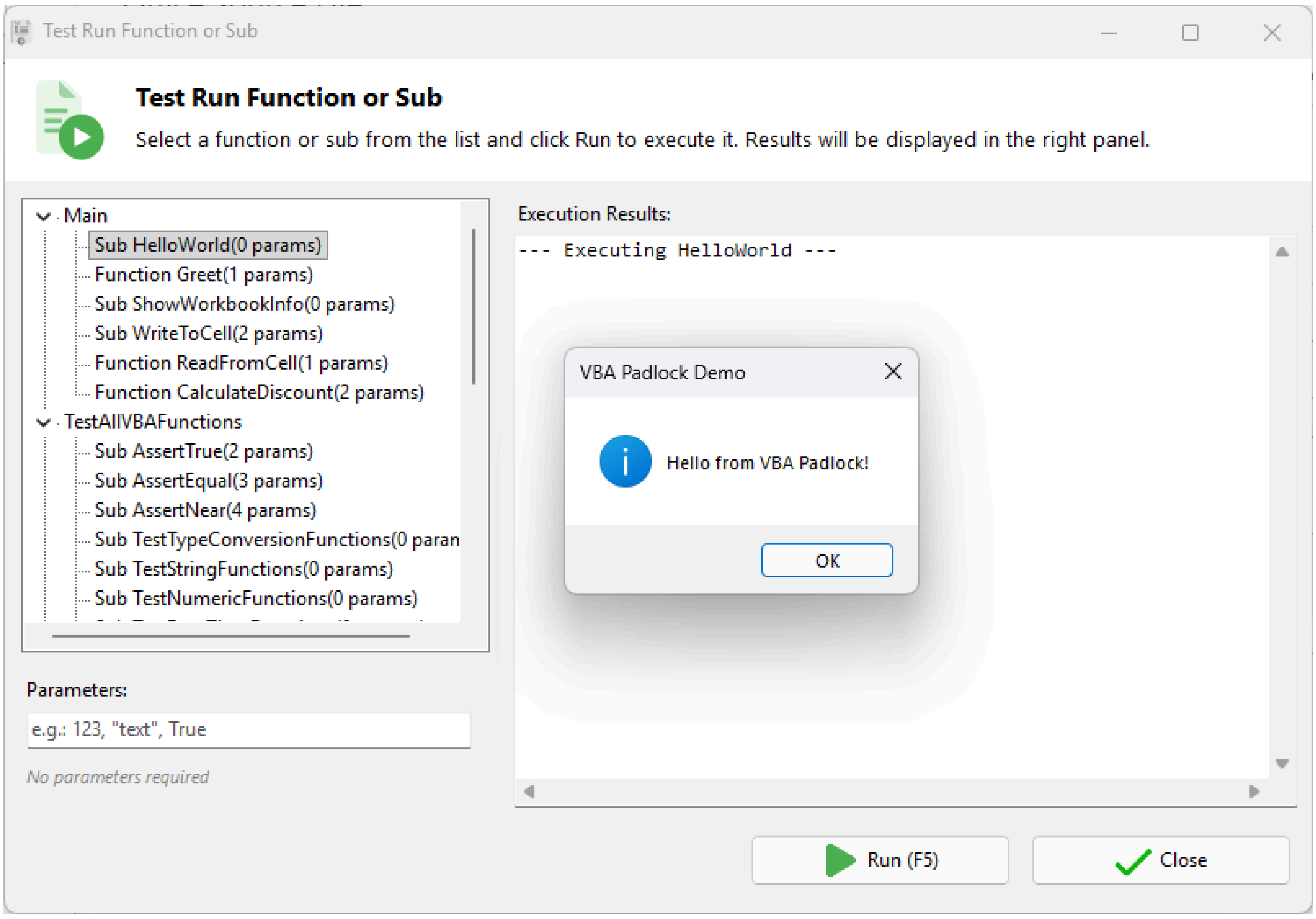
56.2 Compilation Issues

56.2.1 Compilation fails with errors

Cause: The VBA Padlock script contains syntax errors, unsupported functions, or references to undefined variables.

Solution:

- Check the Messages panel for specific error messages and line numbers.
- Review VBA Compatibility for unsupported features (e.g., no class modules, no `Declare` statements, no Office object model access inside scripts).
- Ensure all functions used are either user-defined or from the built-in script libraries.


Test Run Function or Sub

Test Run Function or Sub

Select a function or sub from the list and click Run to execute it. Results will be displayed in the right panel.

- Main
  - Sub HelloWorld(0 params)
  - Function Greet(1 params)
  - Sub ShowWorkbookInfo(0 params)
  - Sub WriteToCell(2 params)
  - Function ReadFromCell(1 params)
  - Function CalculateDiscount(2 params)
- TestAllVBAFunctions
  - Sub AssertTrue(2 params)
  - Sub AssertEqual(3 params)
  - Sub AssertNear(4 params)
  - Sub TestTypeConversionFunctions(0 paran
  - Sub TestStringFunctions(0 params)
  - Sub TestNumericFunctions(0 params)

Parameters:

e.g.: 123, "text", True

No parameters required

Execution Results:

```
--- Executing HelloWorld ---
```

VBA Padlock Demo

Hello from VBA Padlock!

56.2.2 Compilation succeeds but functions return errors at runtime

Cause: The compiled function exists but encounters a runtime issue (wrong parameter count, type mismatch, etc.).

Solution:

- Test your function in the Test Runner before injecting into Office.
- Verify you are passing the correct number and types of parameters to match the compiled function's signature.
- Check the return value — `VBAPL_Execute` returns a `Variant` that may contain an error code.

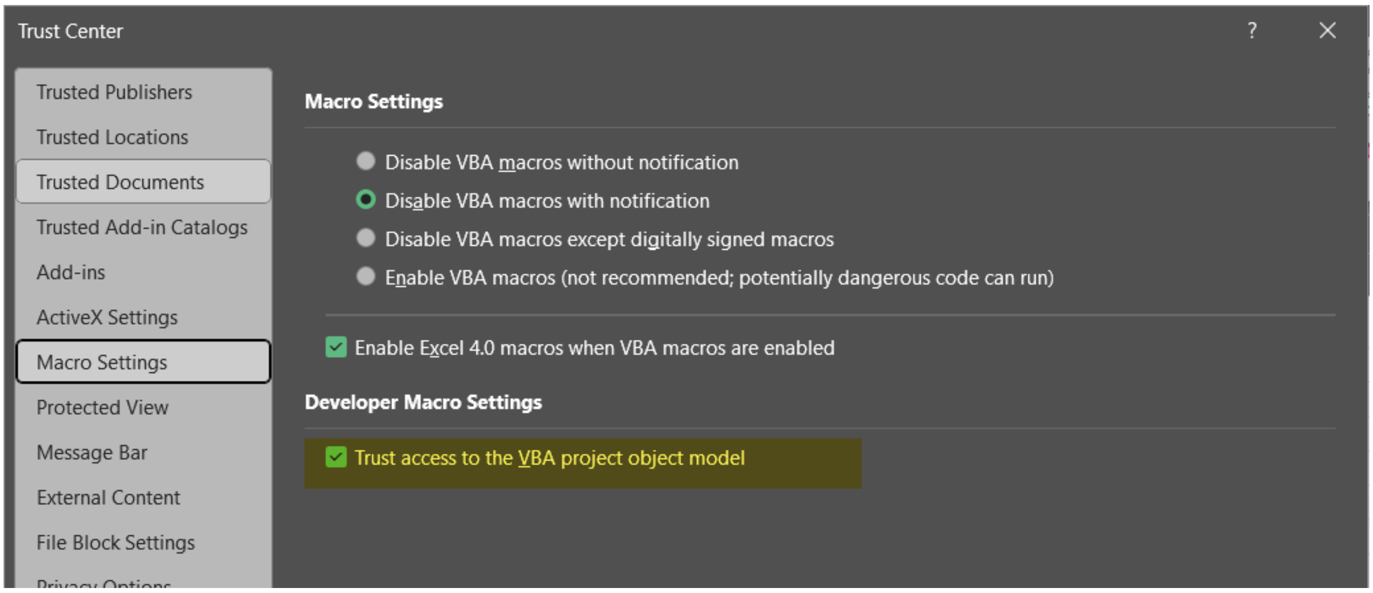
56.3 VBA Bridge Injection Issues

56.3.3 Injection fails or fails to create the VBA Bridge in Office

Cause: VBA Padlock requires programmatic access to the Office VBA project to automate the injection of the bridge module. This access is disabled by default in Office for security reasons.

Solution: You must enable "Trust access to the VBA project object model" in your Office application settings:

1. Open the Office application (Excel, Word, etc.).
2. Go to **File** > **Options**.
3. Select **Trust Center** in the left menu.
4. Click the **Trust Center Settings...** button.
5. Select **Macro Settings** in the left menu.
6. Under **Developer Macro Settings**, check the **Trust access to the VBA project object model** box.
7. Click **OK** and restart the Office application.



Macro Settings

- Disable VBA macros without notification
- [✓ Disable VBA macros with notification]
- Disable VBA macros except digitally signed macros
- Enable VBA macros (not recommended; potentially dangerous code can run)

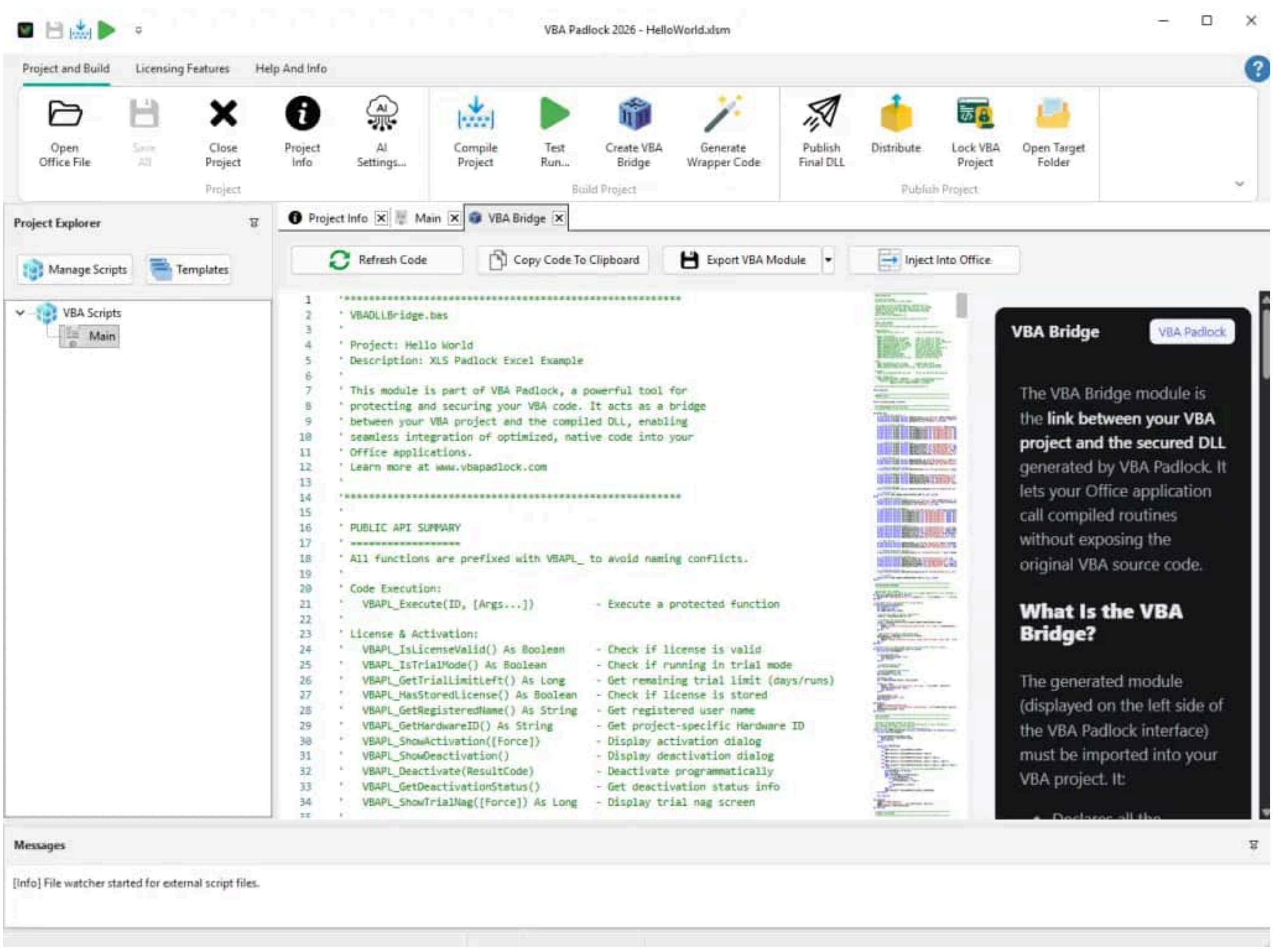
✓ Enable Excel 4.0 macros when VBA macros are enabled

Developer Macro Settings

✓ Trust access to the VBA project object model

Note

This setting is specific to each Office application. If you are protecting both Excel and Word files, you must enable it in both applications.



```
'************************************************************
' VBADLLBridge.bas
'
' Project: Hello World
' Description: XLS Padlock Excel Example
'
' This module is part of VBA Padlock, a powerful tool for
' protecting and securing your VBA code. It acts as a bridge
' between your VBA project and the compiled DLL, enabling
' seamless integration of optimized, native code into your
' Office applications.
' Learn more at www.vbapadlock.com
'
'************************************************************
'
' PUBLIC API SUMMARY
' ==================
' All functions are prefixed with VBAPL_ to avoid naming conflicts.
'
' Code Execution:
'   VBAPL_Execute(ID, [Args...])          - Execute a protected function
'
' License & Activation:
'   VBAPL_IsLicenseValid() As Boolean     - Check if license is valid
'   VBAPL_IsTrialMode() As Boolean        - Check if running in trial mode
'   VBAPL_GetTrialLimitLeft() As Long     - Get remaining trial limit (days/runs)
'   VBAPL_HasStoredLicense() As Boolean   - Check if license is stored
'   VBAPL_GetRegisteredName() As String   - Get registered user name
'   VBAPL_GetHardwareID() As String       - Get project-specific Hardware ID
'   VBAPL_ShowActivation([Force])         - Display activation dialog
'   VBAPL_ShowDeactivation()              - Display deactivation dialog
'   VBAPL_Deactivate(ResultCode)          - Deactivate programmatically
'   VBAPL_GetDeactivationStatus()         - Get deactivation status info
'   VBAPL_ShowTrialNag([Force]) As Long   - Display trial nag screen
'
```

VBA Bridge VBA Padlock

The VBA Bridge module is the **link between your VBA project and the secured DLL** generated by VBA Padlock. It lets your Office application call compiled routines without exposing the original VBA source code.

What Is the VBA Bridge?

The generated module (displayed on the left side of the VBA Padlock interface) must be imported into your VBA project. It:

```
[Info] File watcher started for external script files.
```

56.4 DLL Loading Issues

56.4.4 "DLL not found" or "File not found" error

Cause: The Office file cannot find the DLL files.

Solution:

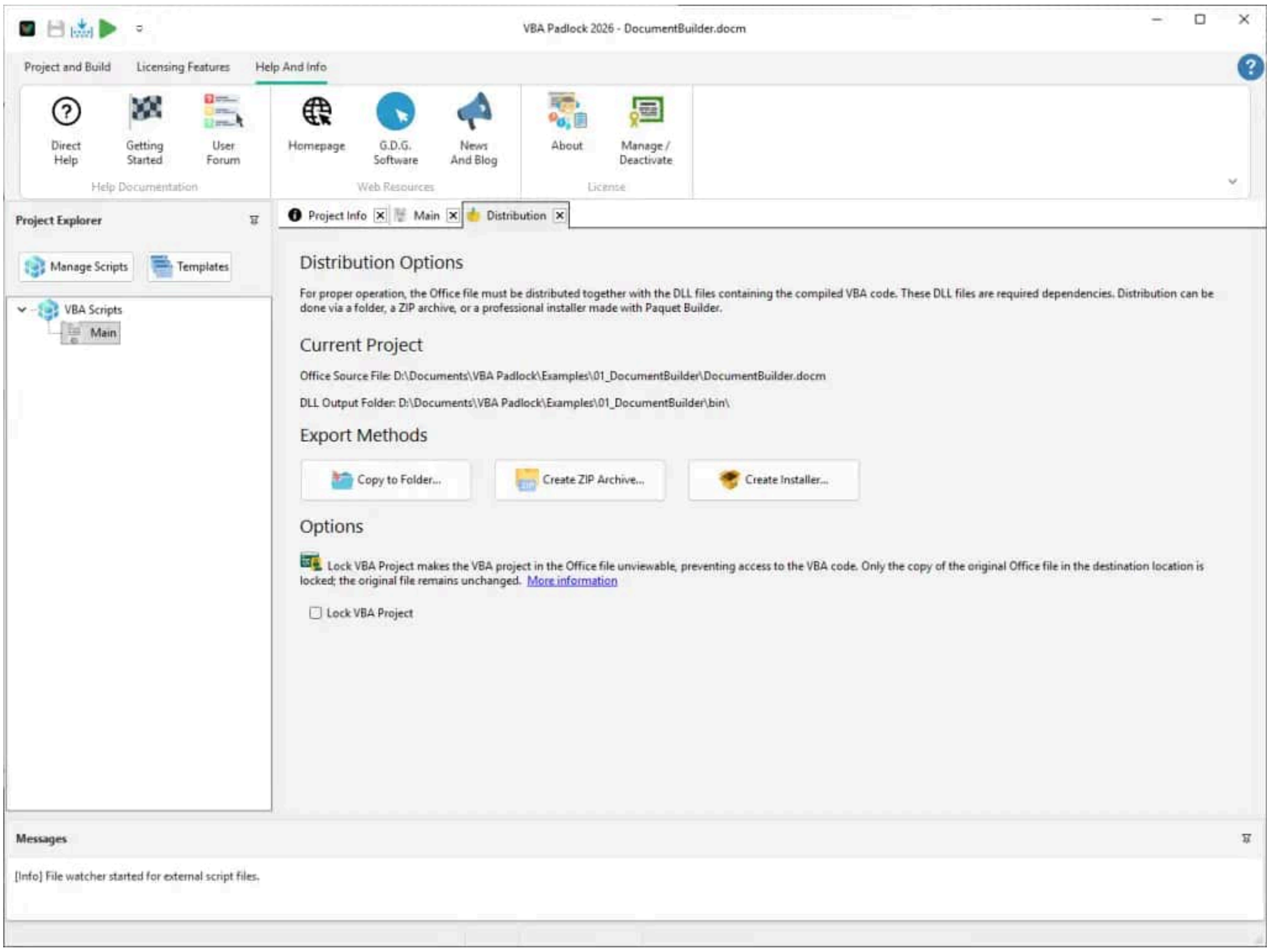
- All three DLLs must be in a `bin/` subdirectory relative to the Office file:

```
MyWorkbook.xlsm
bin\
├── MyWorkbookrun32.dll
├── MyWorkbookrun64.dll
└── MyWorkbook.dll
```

- Verify the `bin/` directory exists and contains all three DLLs.
- If the files were downloaded from the internet, right-click each DLL → **Properties** → check **Unblock**.

Tip

Use the Distribution dialog to package files correctly. It creates the proper directory structure automatically.



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

[Info] File watcher started for external script files.

56.4.5 "Verification failed" or Security Code mismatch

Cause: The Security Code in the VBA Bridge does not match the one in the satellite DLL. This happens when you recompile without re-injecting the VBA Bridge.

Solution:

1. In VBA Padlock, click **Create VBA Bridge → Inject Into Office**.
2. Recompile with `F5`.
3. Use **Publish Final DLL** to build the final version.

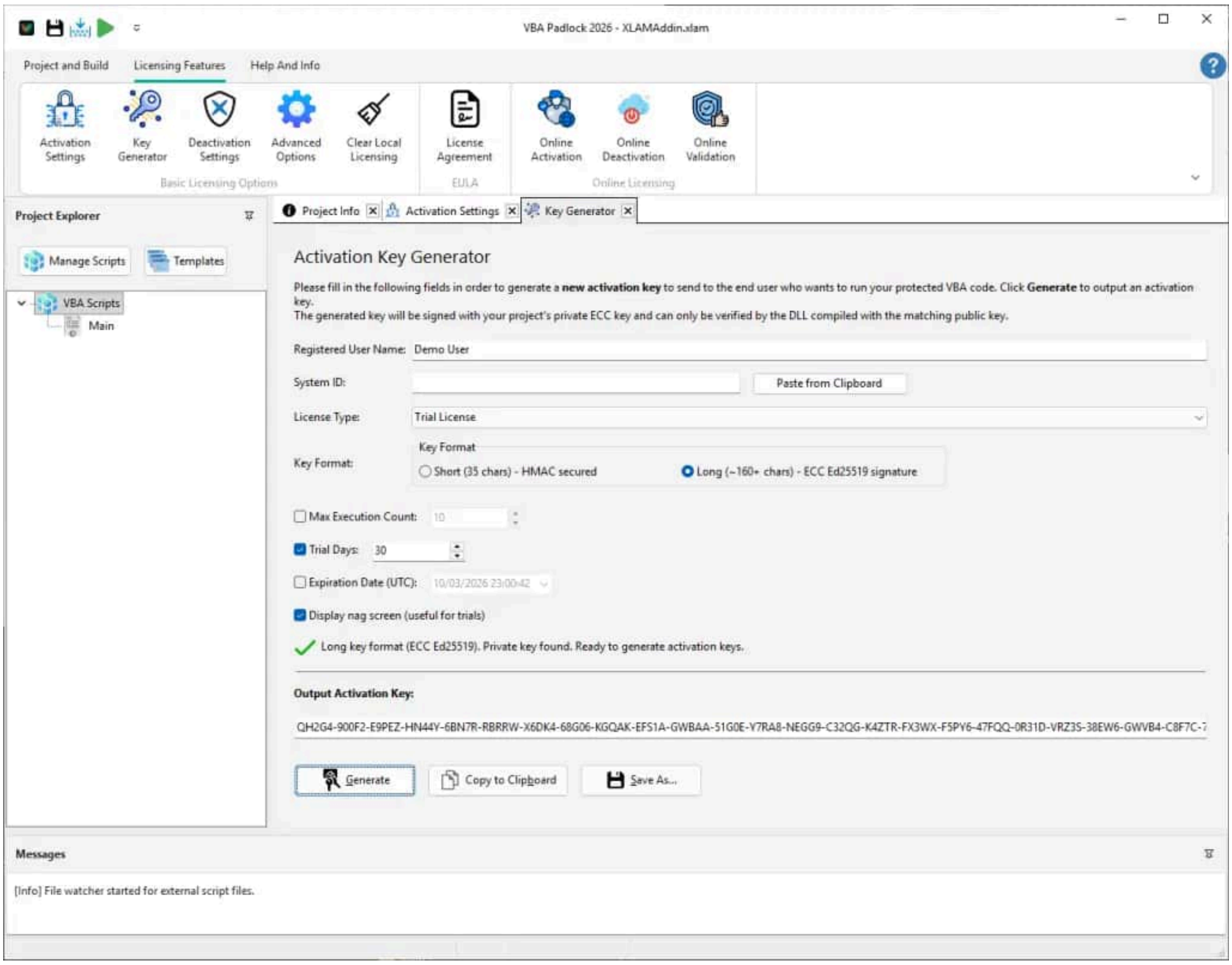
56.4.6 DLL signature verification failed

Cause: A DLL file was modified after compilation, or a file is corrupted.

Solution:

- Recompile the project from VBA Padlock.
- Do not modify or patch the DLL files after compilation — they are integrity-verified.
- If the issue persists, try deleting the `bin/` directory and rebuilding.

56.5 Licensing Issues



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

License Type: Trial License

Key Format: Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

[Info] File watcher started for external script files.

56.5.7 Trial period not starting

Cause: License storage permissions or conflicting installations.

Solution:

- Ensure the user has write access to `HKEY_CURRENT_USER` (registry storage) or to the application folder (`.LIC` file storage).
- Check the storage mode in Activation Settings.
- If using portable mode, verify the `.LIC` file can be created next to the Office file.

56.5.8 License key rejected

Cause: The key format does not match, the key was generated for a different project, or the Hardware ID does not match.

Solution:

- Verify the key was generated with the same project (matching Security Code and ECC keys).
- Check for typos — common confusions: `0` vs `O`, `1` vs `l`, `I` vs `l`.
- If using hardware-locked keys, verify the user's System ID matches the one used to generate the key.
- Use the Key Generator to regenerate the key if needed.

56.5.9 Hardware ID changed unexpectedly

Cause: A hardware component used for the Hardware ID was replaced or updated (new hard drive, network adapter, etc.).

Solution:

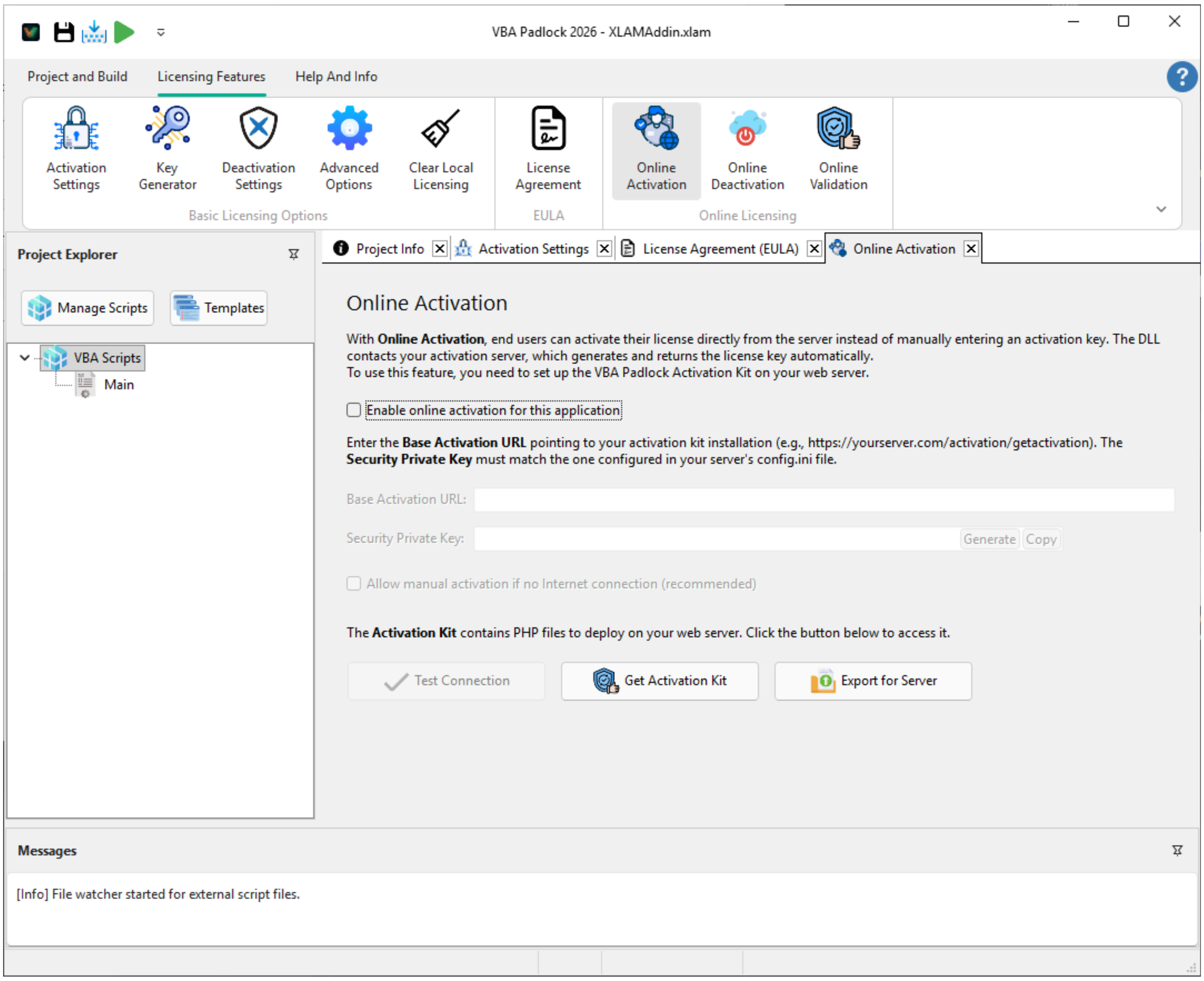
- Check which components are selected in Hardware ID Options.
- Remove volatile components (e.g., MAC address) if users frequently change hardware.
- Generate a new key for the user's updated Hardware ID. See Key Generation guide.

56.5.10 Online activation/deactivation fails

Cause: Network issues, incorrect server URL, or server configuration problems.

Solution:

- Verify the activation URL in Online Activation Settings.
- Test the server connection from VBA Padlock using the **Test Connection** button.
- Ensure the server uses HTTPS and the PHP activation kit is correctly deployed. See Activation Server guide.
- Check the server error logs for details.



Online Activation

With **Online Activation**, end users can activate their license directly from the server instead of manually entering an activation key. The DLL contacts your activation server, which generates and returns the license key automatically.
To use this feature, you need to set up the VBA Padlock Activation Kit on your web server.

☐ Enable online activation for this application

Enter the **Base Activation URL** pointing to your activation kit installation (e.g., https://yourserver.com/activation/getactivation). The **Security Private Key** must match the one configured in your server's config.ini file.

Base Activation URL:

Security Private Key: Generate Copy

☐ Allow manual activation if no Internet connection (recommended)

The **Activation Kit** contains PHP files to deploy on your web server. Click the button below to access it.

Test Connection | Get Activation Kit | Export for Server

[Info] File watcher started for external script files.

56.6 Runtime Issues

56.6.11 Functions work in Test Runner but fail in Office

Cause: The Office file may have an outdated Bridge module, or macros may be disabled.

Solution:

- Re-inject the VBA Bridge: **Create VBA Bridge** → **Inject Into Office**.
- Verify macros are enabled in Office Trust Center settings.

56.6.12 Anti-virus false positive

Cause: Some antivirus software flags DLLs with certain protection patterns as suspicious.

Solution:

- This is a false positive. VBA Padlock runtime DLLs are digitally signed by G.D.G. Software.

- Submit the DLL to your antivirus vendor for whitelisting.
- Add the `bin/` directory to your antivirus exclusion list.

> **Note**
>
> VBA Padlock runtime DLLs are signed with a trusted Authenticode certificate. Most antivirus programs recognize signed DLLs from known publishers as safe. You can also sign the satellite DLL with your own Authenticode certificate if needed.

56.7 Office Compatibility

56.7.13 32-bit vs 64-bit Office

VBA Padlock generates both 32-bit and 64-bit runtime DLLs. The VBA Bridge automatically detects the Office architecture and loads the correct DLL. No configuration is needed.

56.7.14 Office version compatibility

Compiled DLLs are compatible with Office 2016 and later, including Microsoft 365 desktop apps. The VBA Bridge uses `Declare PtrSafe` syntax for 64-bit compatibility.

56.7.15 Multiple Office files sharing the same DLL

Each Office file needs its own satellite DLL (compiled with its own Security Code). The runtime DLLs (`{DLLName}run32.dll` and `{DLLName}run64.dll`) are also project-specific, named after the project's DLL name.

56.7.16 Support for Windows ARM

Issue: Office ARM is installed, and VBA Padlock DLLs fail to load.

Cause: Native ARM versions of Office cannot load standard x86 or x64 DLLs.

Solution:

- On Windows ARM (like **Surface Pro 11, Surface Laptop 7, or Snapdragon X Elite** devices), uninstall the ARM-native version of Office and install **Office 64-bit** instead.
- Office 64-bit on Windows ARM uses Microsoft's **Prism** emulation. This allows the standard x64 versions of VBA Padlock DLLs to load and execute seamlessly without any modifications.

> **Important compatibility note**
>
> For ARM devices, the key point is the **Office build architecture**, not just the CPU architecture. Use Office 64-bit (x64 emulated by Prism), not Office ARM-native.

56.8 Getting Help

If your issue is not covered here:

- Check the License Return Codes for error code meanings
- Review the VBA Compatibility page for supported features
- Visit the FAQ for additional answers
- Contact support for personalized assistance

When contacting support

Include: Office version/build, 32/64-bit architecture, exact error message, and whether the issue appears in **Test Runner** or only in Office.

57. Frequently Asked Questions

Find quick answers to the most common questions about VBA Padlock: compatibility, compilation, licensing, distribution, and security.

General →

What VBA Padlock is, supported Office apps, trial limitations, ARM support.

Compilation →

What can be compiled, function parameters, and common build behavior.

Licensing →

Trial keys, hardware locking, key formats, and transfers.

Distribution & Security →

Required files, signatures, installer, and protection model.

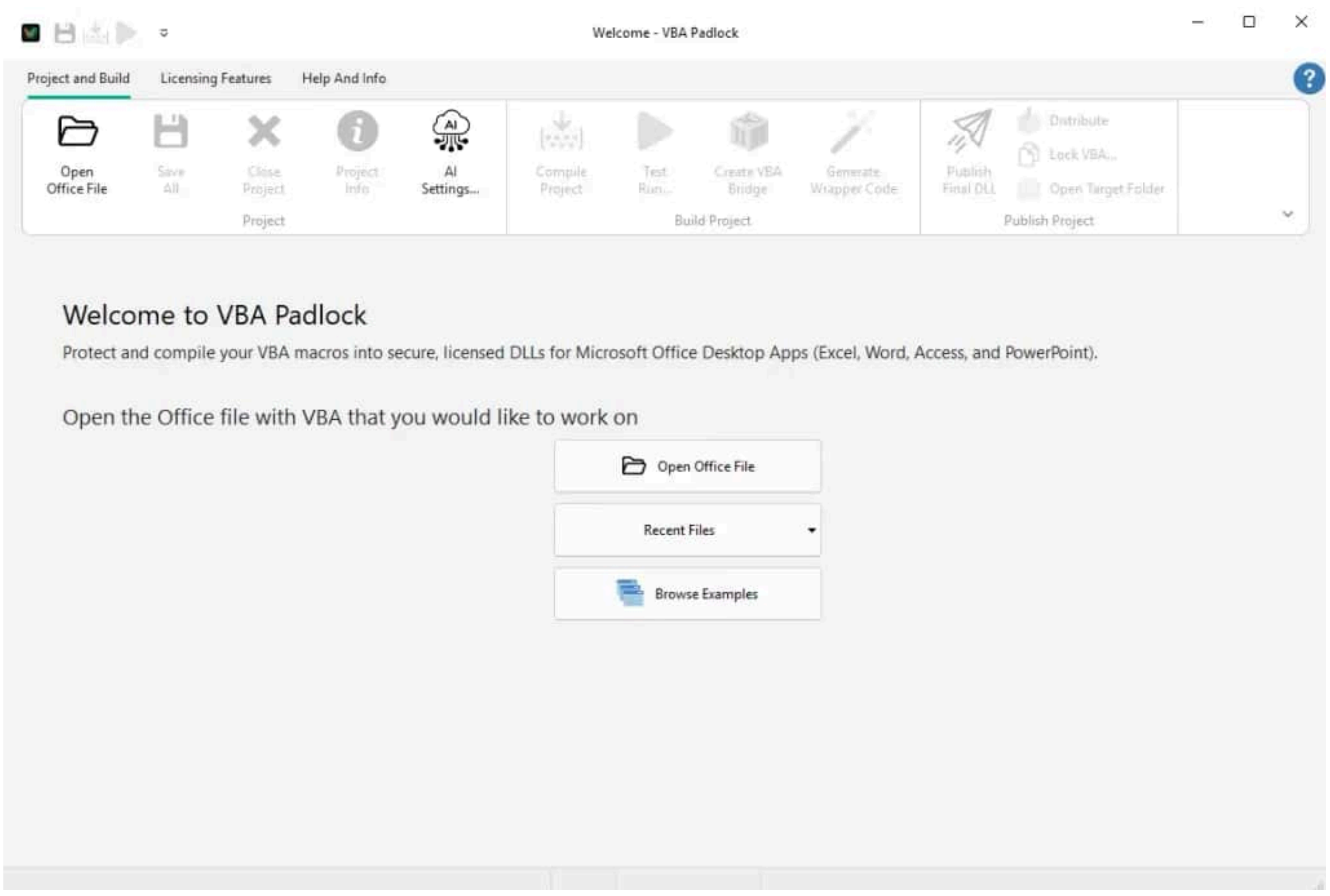
Online Activation →

Server requirements and offline fallback options.

Need a practical walkthrough?

If you prefer a guided setup instead of browsing Q&A, start with the Quick Start and then jump to Troubleshooting when needed.

57.1 General



Welcome to VBA Padlock

Protect and compile your VBA macros into secure, licensed DLLs for Microsoft Office Desktop Apps (Excel, Word, Access, and PowerPoint).

Open the Office file with VBA that you would like to work on

57.1.1 What is VBA Padlock?

VBA Padlock is a development tool that compiles your VBA source code into protected DLLs. Your Office applications (Excel, Word, Access, PowerPoint) call the compiled functions through a VBA Bridge module, keeping your business logic hidden and tamper-resistant.

57.1.2 Which Office applications are supported?

VBA Padlock supports:

- **Excel** — `.xlsm`, `.xlsb`, `.xlam`
- **Word** — `.docm`, `.dotm`
- **Access** — `.accdb`
- **PowerPoint** — `.pptm`, `.ppam`

57.1.3 What are the limitations of the evaluation version?

The free evaluation version includes all features of VBA Padlock but adds a notification screen when your protected VBA code runs.

Important: DLLs compiled with the evaluation version automatically **expire after a few days**. These DLLs are intended only for local testing and evaluation on your machine; they are **not permitted for distribution**. To create permanent, distributable DLLs, you must purchase a license.

57.1.4 Does VBA Padlock require .NET Framework?

No. VBA Padlock is a native Windows application with no external runtime dependencies. The compiled DLLs also do not require .NET.

57.1.5 Does it work with both 32-bit and 64-bit Office?

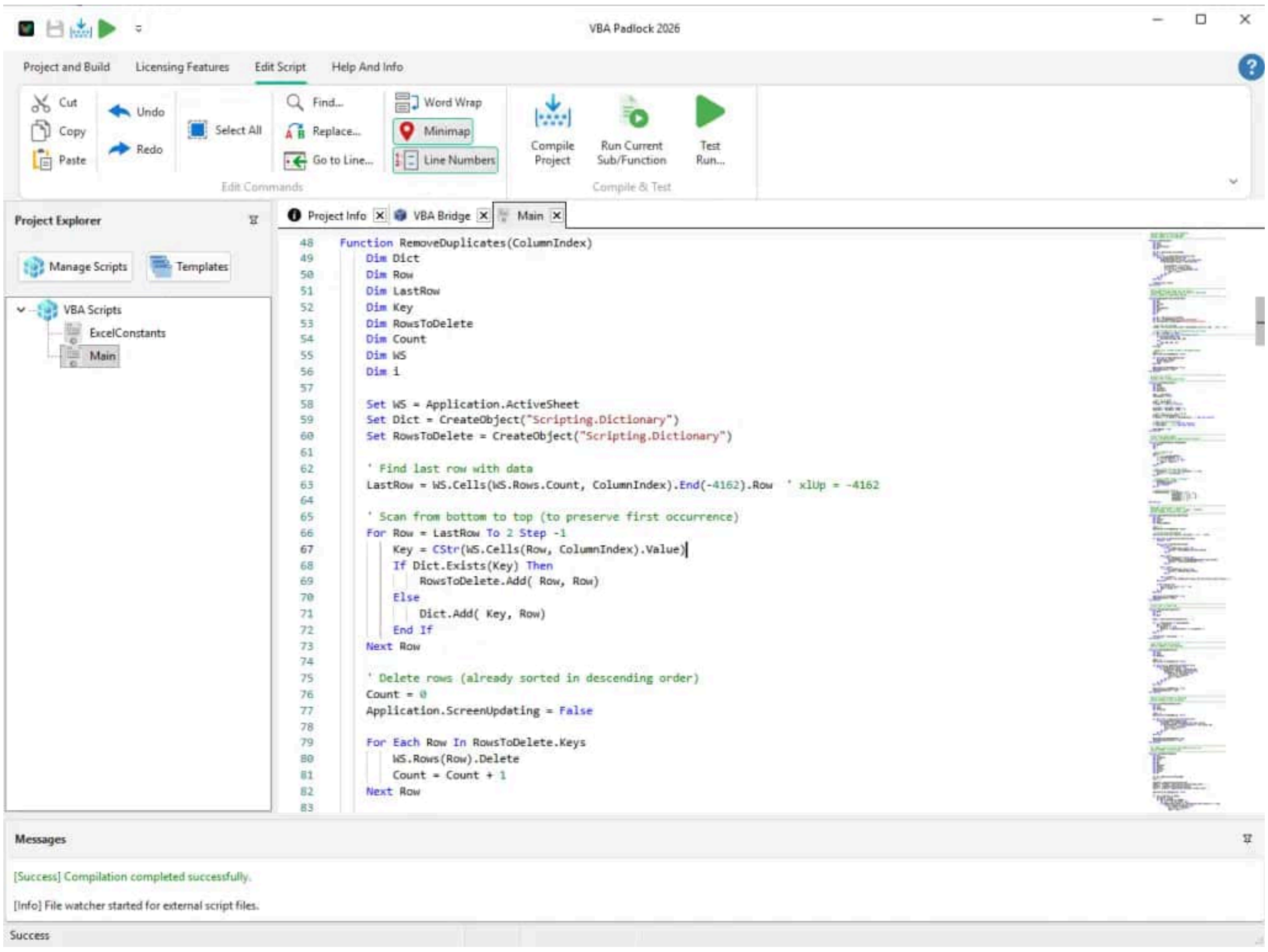
Yes. VBA Padlock produces two runtime DLLs (32-bit and 64-bit). The VBA Bridge automatically loads the correct one based on the Office architecture. No configuration is needed.

57.1.6 Does VBA Padlock support Windows ARM or Office ARM?

Native Office ARM is not supported for now. However, on **Windows ARM** devices (e.g., **Surface Pro 11, Surface Laptop 7, or Snapdragon X Elite** models), we recommend installing **Office 64-bit**. It runs via Microsoft's **Prism** emulation layer, which allows VBA Padlock's x64 DLLs to function perfectly.

> ⚠ **ARM compatibility rule**
>
> For ARM devices, what matters is your **Office build architecture**. Use **Office 64-bit** (x64 through Prism emulation), not ARM-native Office.

57.2 Compilation



```
Function RemoveDuplicates(ColumnIndex)
    Dim Dict
    Dim Row
    Dim LastRow
    Dim Key
    Dim RowsToDelete
    Dim Count
    Dim WS
    Dim i

    Set WS = Application.ActiveSheet
    Set Dict = CreateObject("Scripting.Dictionary")
    Set RowsToDelete = CreateObject("Scripting.Dictionary")

    ' Find last row with data
    LastRow = WS.Cells(WS.Rows.Count, ColumnIndex).End(-4162).Row  ' xlUp = -4162

    ' Scan from bottom to top (to preserve first occurrence)
    For Row = LastRow To 2 Step -1
        Key = CStr(WS.Cells(Row, ColumnIndex).Value)
        If Dict.Exists(Key) Then
            RowsToDelete.Add( Row, Row)
        Else
            Dict.Add( Key, Row)
        End If
    Next Row

    ' Delete rows (already sorted in descending order)
    Count = 0
    Application.ScreenUpdating = False

    For Each Row In RowsToDelete.Keys
        WS.Rows(Row).Delete
        Count = Count + 1
    Next Row

```

```
[Success] Compilation completed successfully.
[Info] File watcher started for external script files.
```

57.2.7 What VBA features can I compile?

VBA Padlock compiles standard VBA modules (`.bas` files). Compiled scripts have access to 60+ built-in functions for license checks, hashing, date/time calculations, and more. See VBA Compatibility for supported features and limitations.

57.2.8 Can I use Office objects (Worksheets, Ranges, Documents) inside compiled scripts?

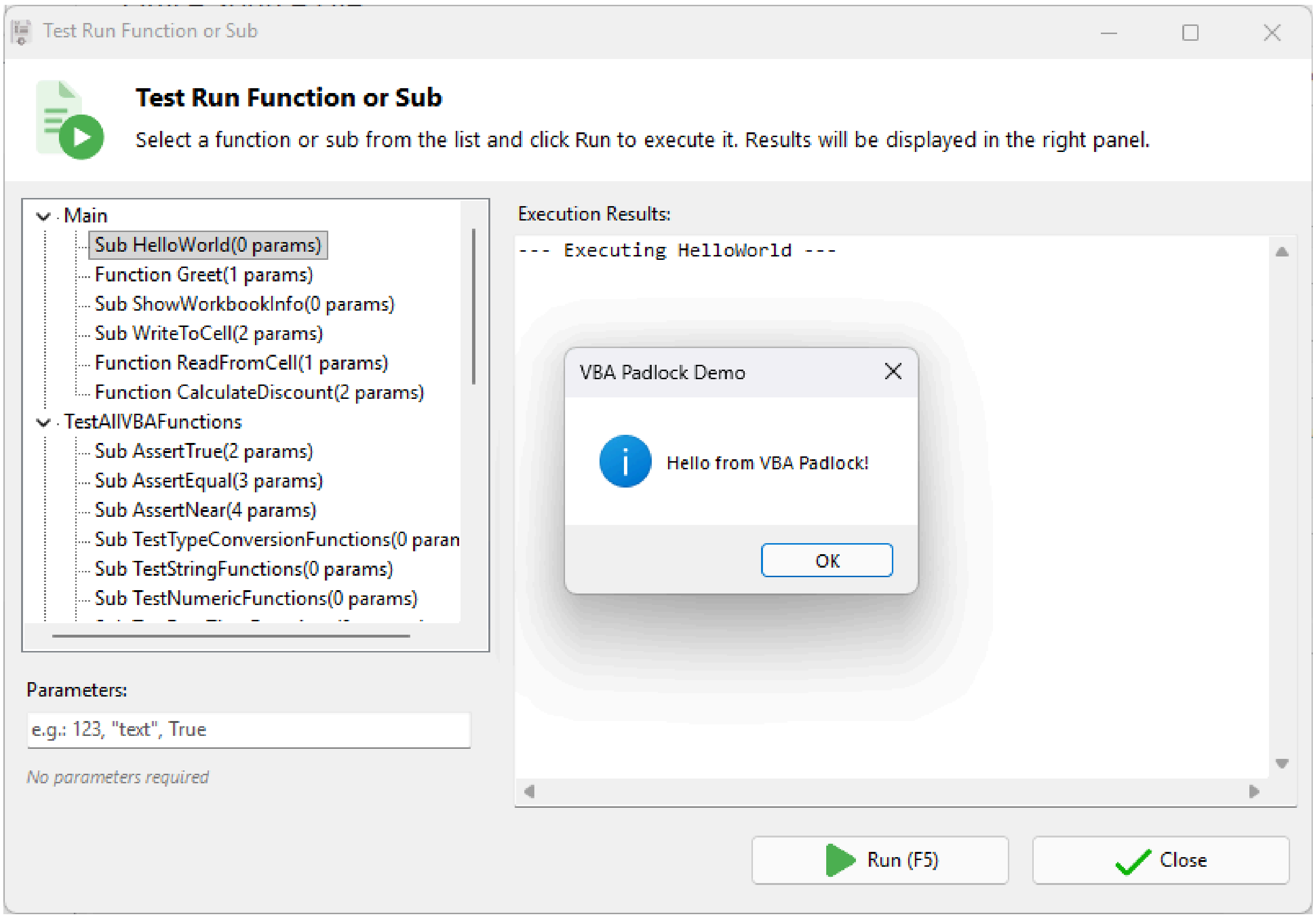
Yes. Compiled scripts have full access to the host Office application through the `Application` object. You can read and write cell values, manipulate documents, create slides, query databases, and use `CreateObject()` for COM objects — just like regular VBA. See VBA Compatibility for details and examples.

57.2.9 How many parameters can a function accept?

`VBAPL_Execute` supports **any number of parameters** — there is no limit. Internally, calls with 0 to 4 arguments use optimized DLL exports, and calls with 5 or more arguments are automatically packed into an array. This is handled transparently by the VBA Bridge.

57.2.10 What happens if my script has a compilation error?

The Messages panel shows the error location and description. Fix the error in the Code Editor and press `F5` to recompile. You can also use the Test Runner to verify individual functions.


Test Run Function or Sub
Test Run Function or Sub
Select a function or sub from the list and click Run to execute it. Results will be displayed in the right panel.
Main
Sub HelloWorld(0 params)
Function Greet(1 params)
Sub ShowWorkbookInfo(0 params)
Sub WriteToCell(2 params)
Function ReadFromCell(1 params)
Function CalculateDiscount(2 params)
TestAllVBAFunctions
Sub AssertTrue(2 params)
Sub AssertEqual(3 params)
Sub AssertNear(4 params)
Sub TestTypeConversionFunctions(0 paran
Sub TestStringFunctions(0 params)
Sub TestNumericFunctions(0 params)
Parameters:
e.g.: 123, "text", True
No parameters required
Execution Results:
VBA Padlock Demo
OK
Run (F5)
Close

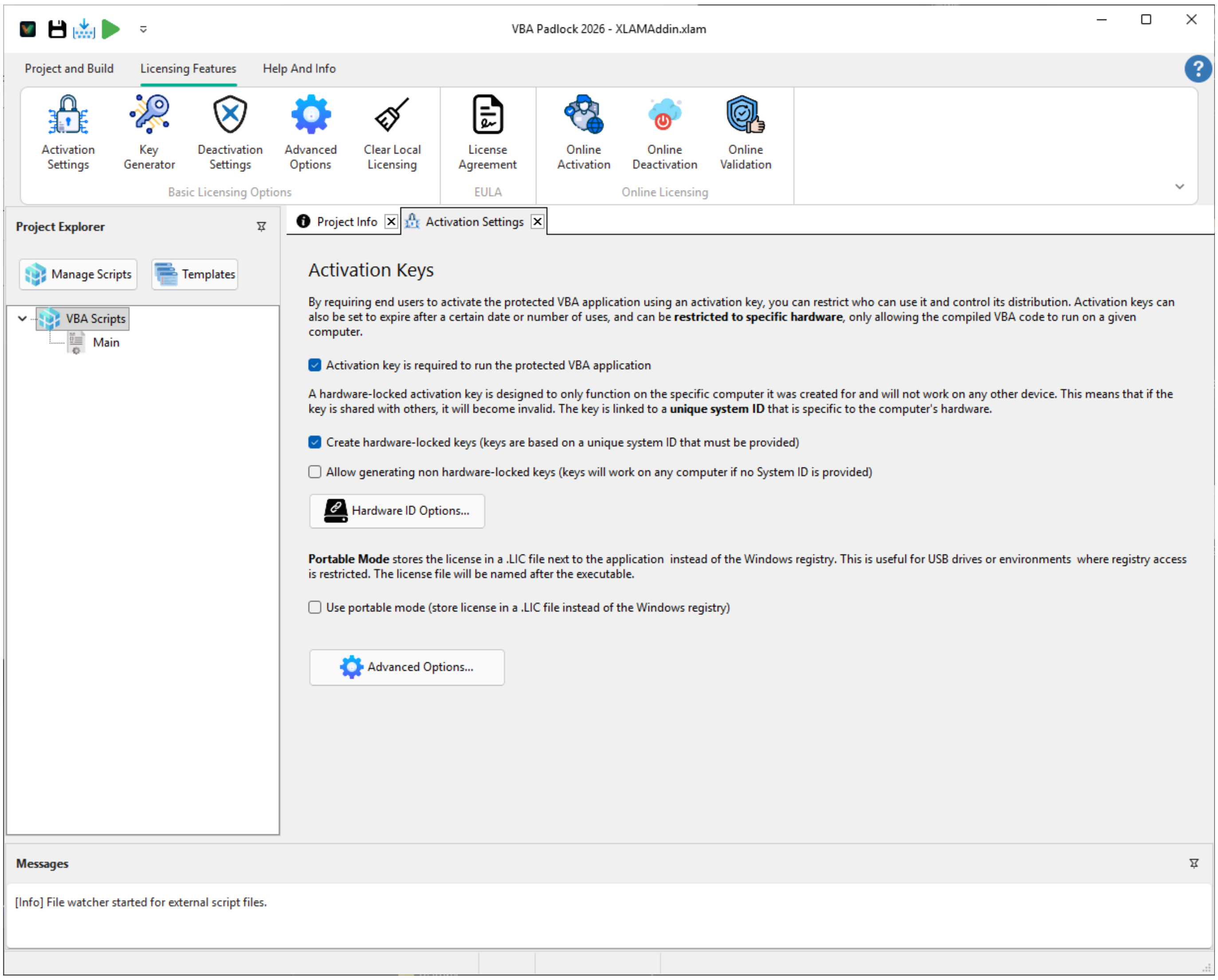
```
--- Executing HelloWorld ---
```

Hello from VBA Padlock!

Tip

Fast workflow for diagnostics: **Compile** → **Test Runner** → **Office macro call**.

57.3 Licensing



Activation Keys

By requiring end users to activate the protected VBA application using an activation key, you can restrict who can use it and control its distribution. Activation keys can also be set to expire after a certain date or number of uses, and can be **restricted to specific hardware**, only allowing the compiled VBA code to run on a given computer.

[✓] Activation key is required to run the protected VBA application

A hardware-locked activation key is designed to only function on the specific computer it was created for and will not work on any other device. This means that if the key is shared with others, it will become invalid. The key is linked to a **unique system ID** that is specific to the computer's hardware.

[✓] Create hardware-locked keys (keys are based on a unique system ID that must be provided)

[ ] Allow generating non hardware-locked keys (keys will work on any computer if no System ID is provided)

Hardware ID Options...

Portable Mode stores the license in a .LIC file next to the application instead of the Windows registry. This is useful for USB drives or environments where registry access is restricted. The license file will be named after the executable.

[ ] Use portable mode (store license in a .LIC file instead of the Windows registry)

Advanced Options...

[Info] File watcher started for external script files.

57.3.11 How does the licensing system work?

VBA Padlock can require a license key before your compiled functions execute. You configure activation in the **Licensing Features** tab and generate keys with the Key Generator. See Licensing System for a full overview.

57.3.12 Can I offer a trial period?

Yes. You can create trial keys with time limits (e.g., 30 days), run limits, or expiration dates. A nag screen reminds users to purchase during the trial. See Trial Mode.

57.3.13 What is hardware locking?

Hardware locking ties a license key to a specific computer using a **Hardware ID** derived from hardware components (CPU, hard drive, etc.). This prevents key sharing. See Hardware Locking.

57.3.14 Can users transfer their license to a new computer?

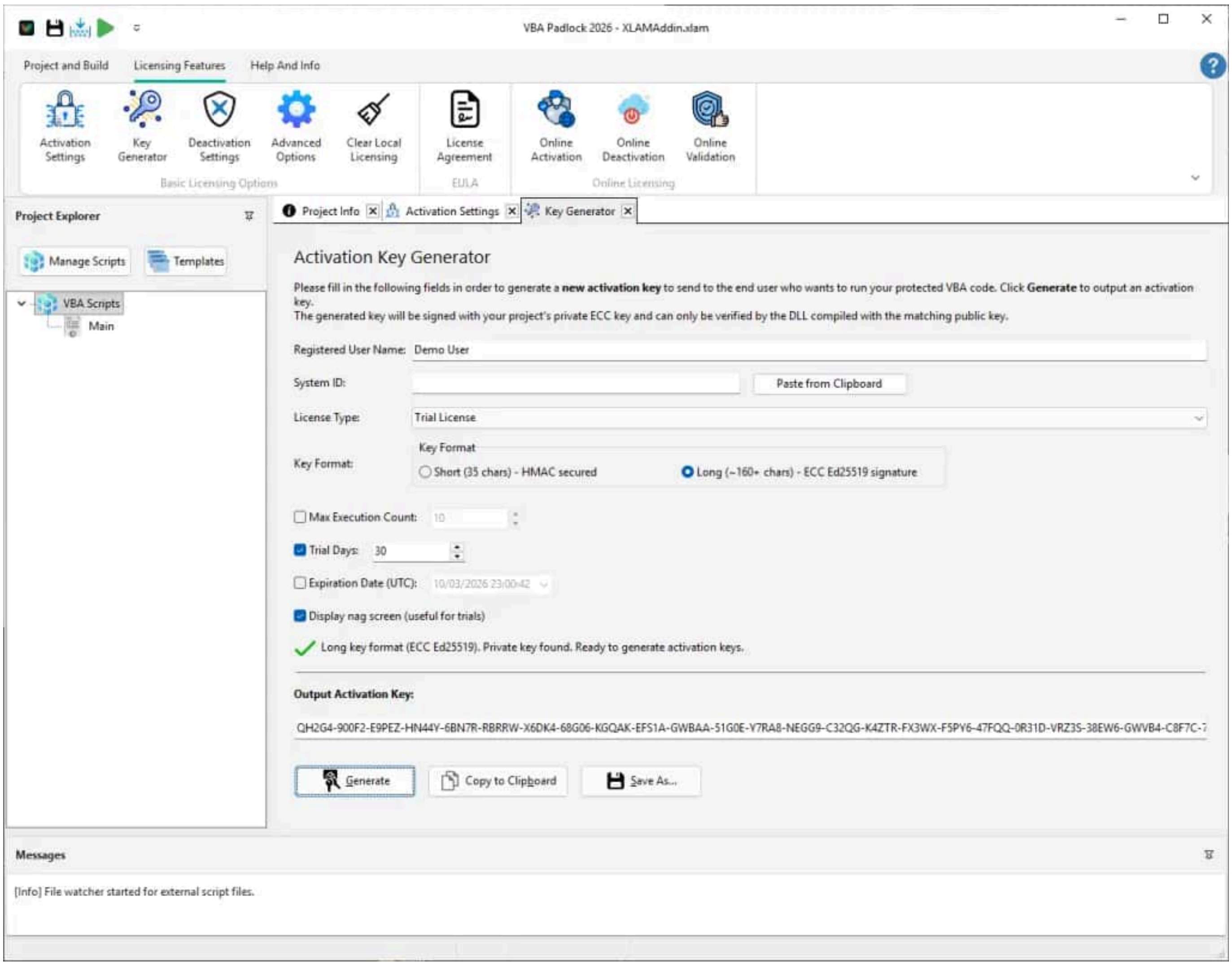
Yes, through deactivation. The user deactivates on the old machine (online or manually), and you issue a new key for the new machine. With online activation, this process is fully automated.

57.3.15 What key formats are available?

- HMAC-based and easy to type manually.
- Supports expiration dates, trial days, and hardware locking.
- Best for simple activation workflows and lower support friction.

- Ed25519-signed (ECC), maximum cryptographic security.
- Supports all options, including **Max Execution Count** and nag-screen based trials.
- Recommended for advanced licensing models.

Run-limited trials require the **Long** format.



Activation Key Generator

Please fill in the following fields in order to generate a **new activation key** to send to the end user who wants to run your protected VBA code. Click **Generate** to output an activation key.
The generated key will be signed with your project's private ECC key and can only be verified by the DLL compiled with the matching public key.

Registered User Name: Demo User

System ID:

Paste from Clipboard

License Type: Trial License

Key Format: Key Format ○ Short (35 chars) - HMAC secured [✓ Long (~160+ chars) - ECC Ed25519 signature]

☐ Max Execution Count: 10

☑ Trial Days: 30

☐ Expiration Date (UTC): 10/03/2026 23:00:42

☑ Display nag screen (useful for trials)

✓ Long key format (ECC Ed25519). Private key found. Ready to generate activation keys.

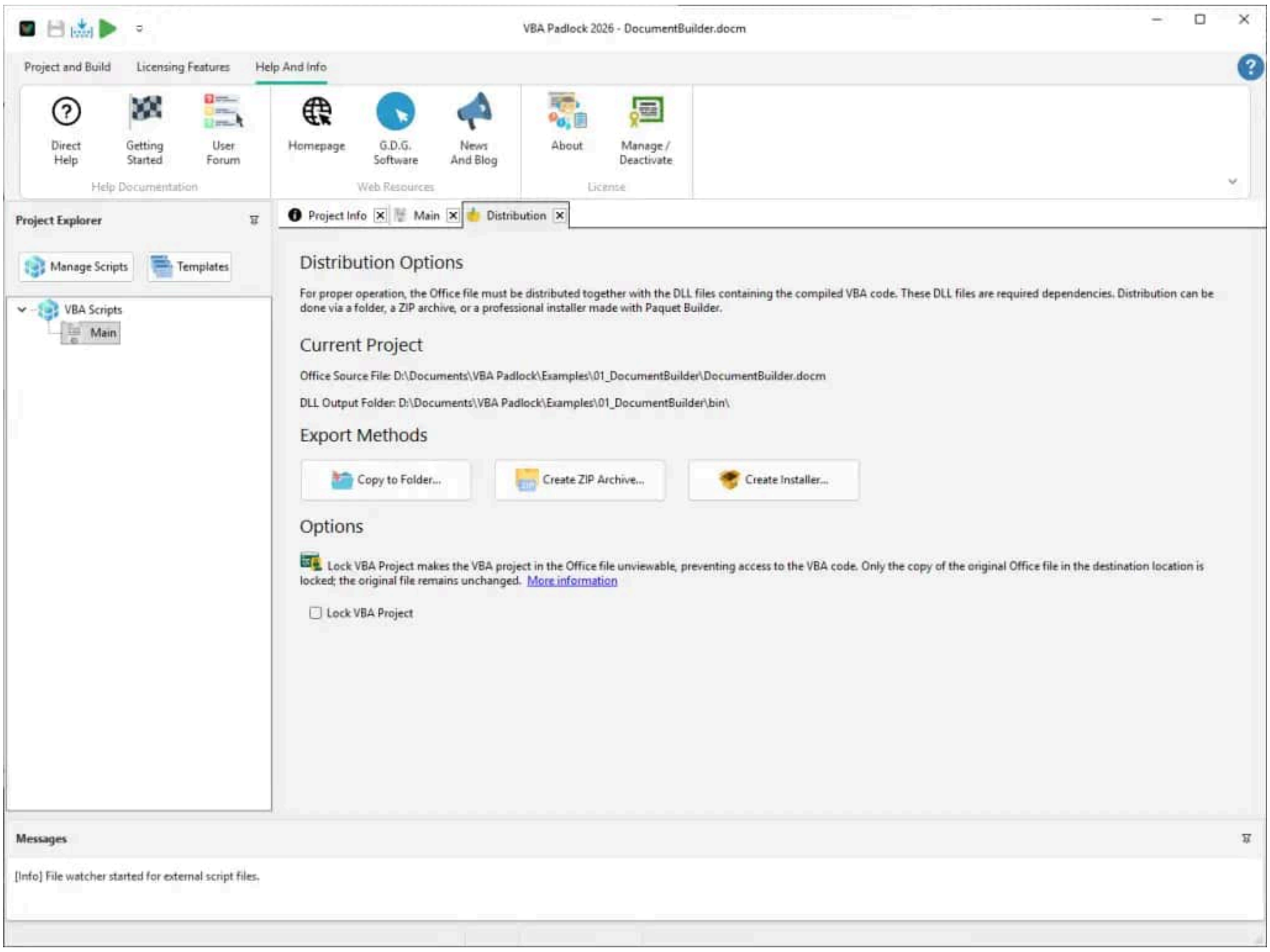
Output Activation Key:

QH2G4-900F2-E9PEZ-HN44Y-6BN7R-RBRRW-X6DK4-68G06-KGQAK-EFS1A-GWBAA-51G0E-Y7RA8-NEGG9-C32QG-K4ZTR-FX3WX-F5PY6-47FQQ-0R31D-VRZ3S-38EW6-GWVB4-C8F7C-7

Generate Copy to Clipboard Save As...

[Info] File watcher started for external script files.

57.4 Distribution



Distribution Options

For proper operation, the Office file must be distributed together with the DLL files containing the compiled VBA code. These DLL files are required dependencies. Distribution can be done via a folder, a ZIP archive, or a professional installer made with Paquet Builder.

Current Project

Office Source File: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\DocumentBuilder.docm

DLL Output Folder: D:\Documents\VBA Padlock\Examples\01_DocumentBuilder\bin\

Export Methods

Options

Lock VBA Project makes the VBA project in the Office file unviewable, preventing access to the VBA code. Only the copy of the original Office file in the destination location is locked; the original file remains unchanged. More information

☐ Lock VBA Project

[Info] File watcher started for external script files.

57.4.16 What files do I need to distribute?

Your distribution must include the Office file and the `bin/` directory with three DLLs:

```
MyWorkbook.xlsm
bin\
├── MyWorkbookrun32.dll
├── MyWorkbookrun64.dll
└── MyWorkbook.dll
```

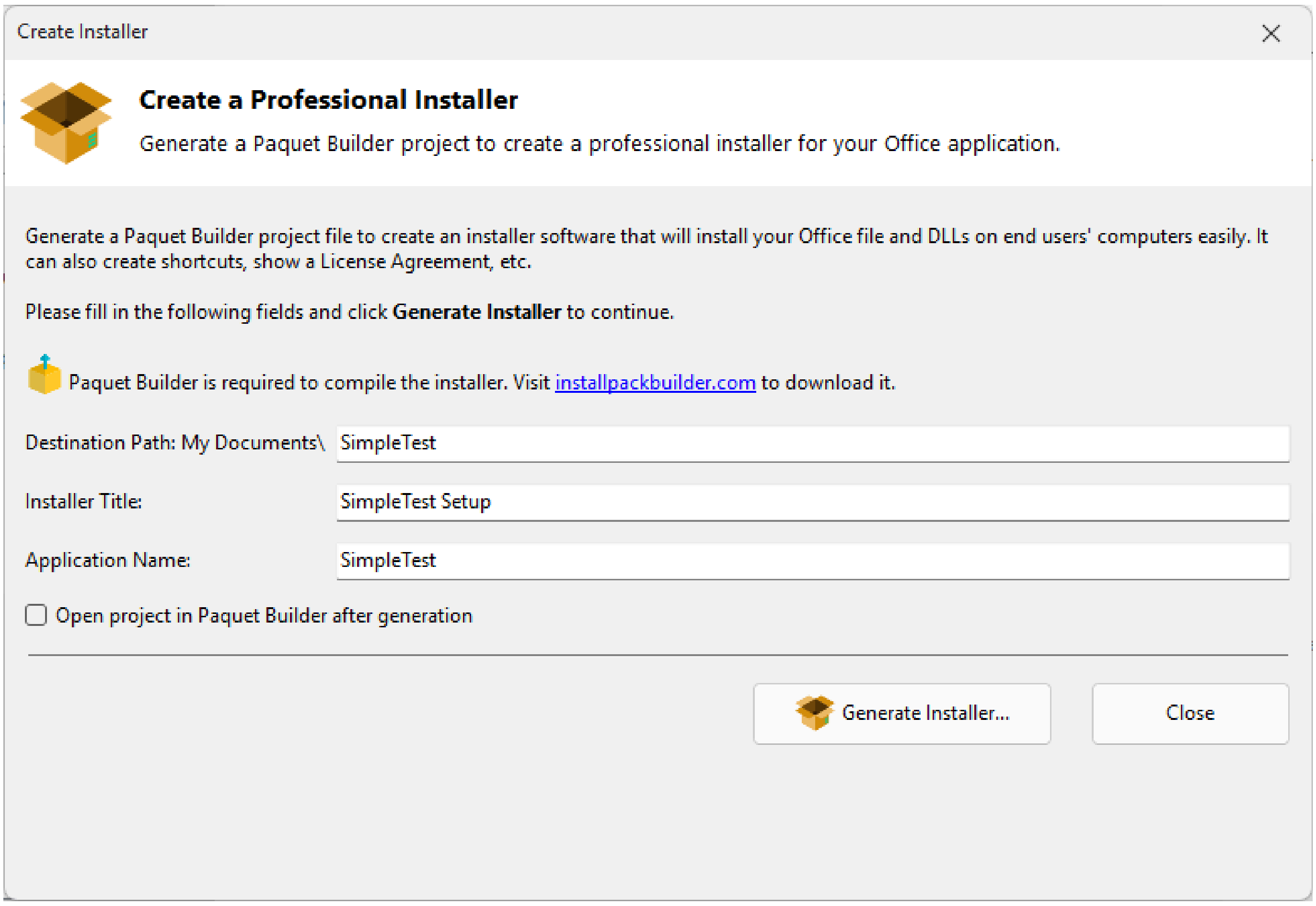
Use the Distribution dialog to package files automatically.

57.4.17 Are the DLLs digitally signed?

The runtime DLLs are signed by G.D.G. Software using Authenticode certificates. Windows SmartScreen and antivirus software recognize them as legitimate. The satellite DLL is not Authenticode-signed by G.D.G. Software but is integrity-verified by an internal cryptographic signature. You can optionally sign it with your own Authenticode certificate.

57.4.18 Can I create an installer for my application?

Yes. The Create Installer dialog generates installer scripts using Paquet Builder.



Create a Professional Installer

Generate a Paquet Builder project to create a professional installer for your Office application.

Generate a Paquet Builder project file to create an installer software that will install your Office file and DLLs on end users' computers easily. It can also create shortcuts, show a License Agreement, etc.

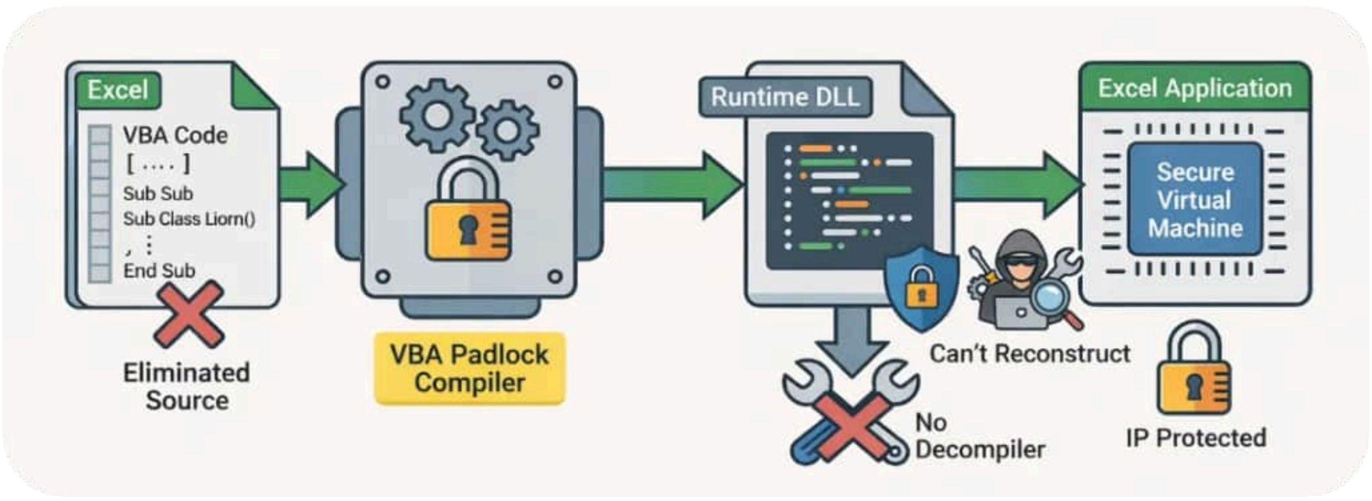
Please fill in the following fields and click **Generate Installer** to continue.

Paquet Builder is required to compile the installer. Visit installpackbuilder.com to download it.

Destination Path: My Documents\	SimpleTest
Installer Title:	SimpleTest Setup
Application Name:	SimpleTest

☐ Open project in Paquet Builder after generation

57.5 Security



57.5.19 How secure is the protection?

VBA Padlock applies multiple defense layers: compilation into bytecode, digital signatures, anti-debugging, and integrity checks. No protection is unbreakable, but VBA Padlock raises the difficulty of reverse engineering significantly. See Security.

57.5.20 My antivirus flags the DLL as suspicious. Is it safe?

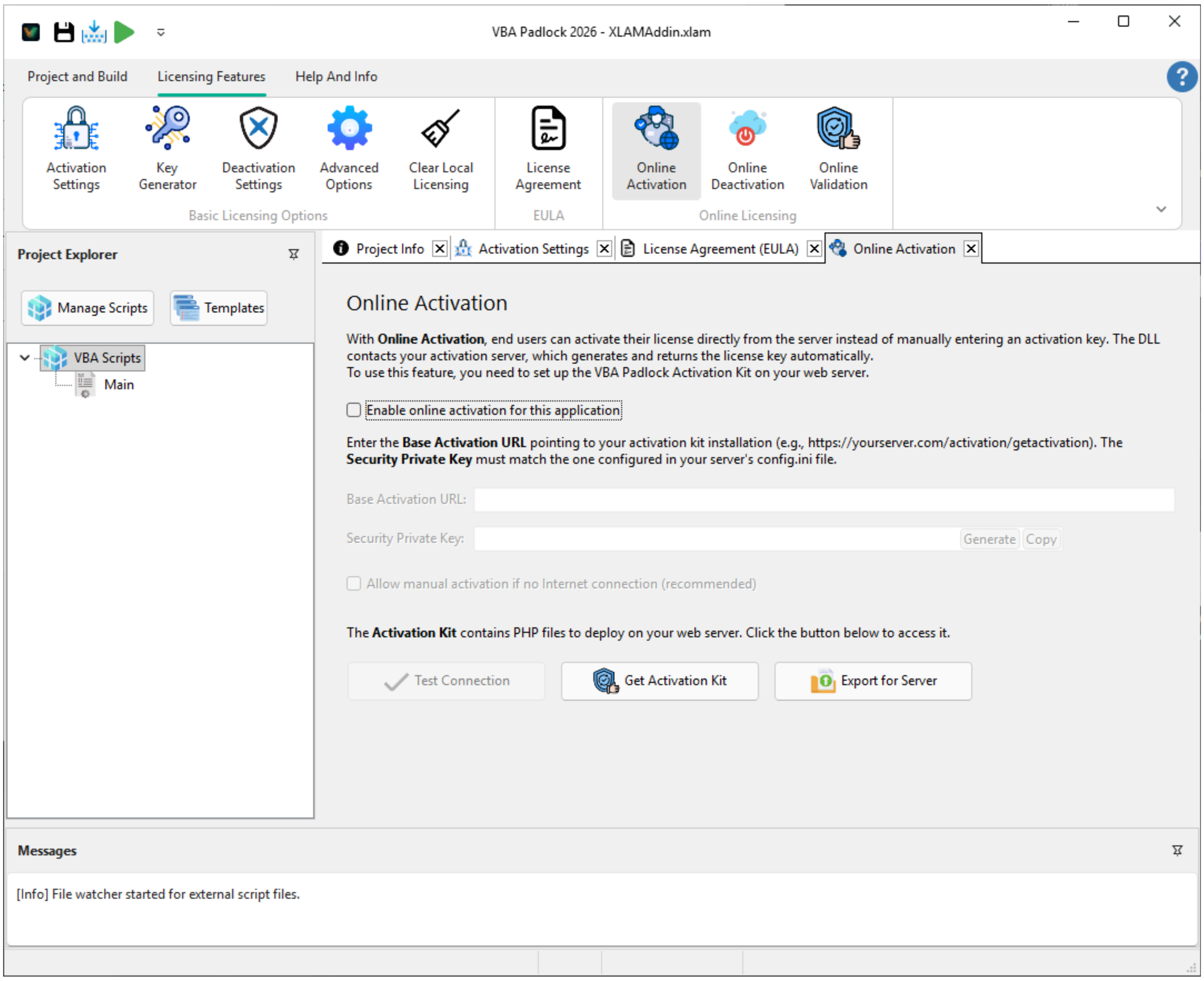
This is a false positive caused by protection patterns. VBA Padlock runtime DLLs are digitally signed by G.D.G. Software. Submit the DLL to your antivirus vendor for whitelisting, or add the `bin/` directory to your exclusion list.

57.5.21 What is the Security Code?

The Security Code is a unique identifier that links your Office file to the compiled DLL. It prevents someone from substituting a DLL from a different project. You configure it in Project Info.

> **Security best practice**
>
> After changing your Security Code or DLL name, always re-inject the VBA Bridge before final testing and distribution.

57.6 Online Activation



Online Activation

With **Online Activation**, end users can activate their license directly from the server instead of manually entering an activation key. The DLL contacts your activation server, which generates and returns the license key automatically.
To use this feature, you need to set up the VBA Padlock Activation Kit on your web server.

☐ Enable online activation for this application

Enter the **Base Activation URL** pointing to your activation kit installation (e.g., https://yourserver.com/activation/getactivation). The **Security Private Key** must match the one configured in your server's config.ini file.

Base Activation URL:

Security Private Key:

☐ Allow manual activation if no Internet connection (recommended)

The **Activation Kit** contains PHP files to deploy on your web server. Click the button below to access it.

[Info] File watcher started for external script files.

57.6.22 Do I need a web server?

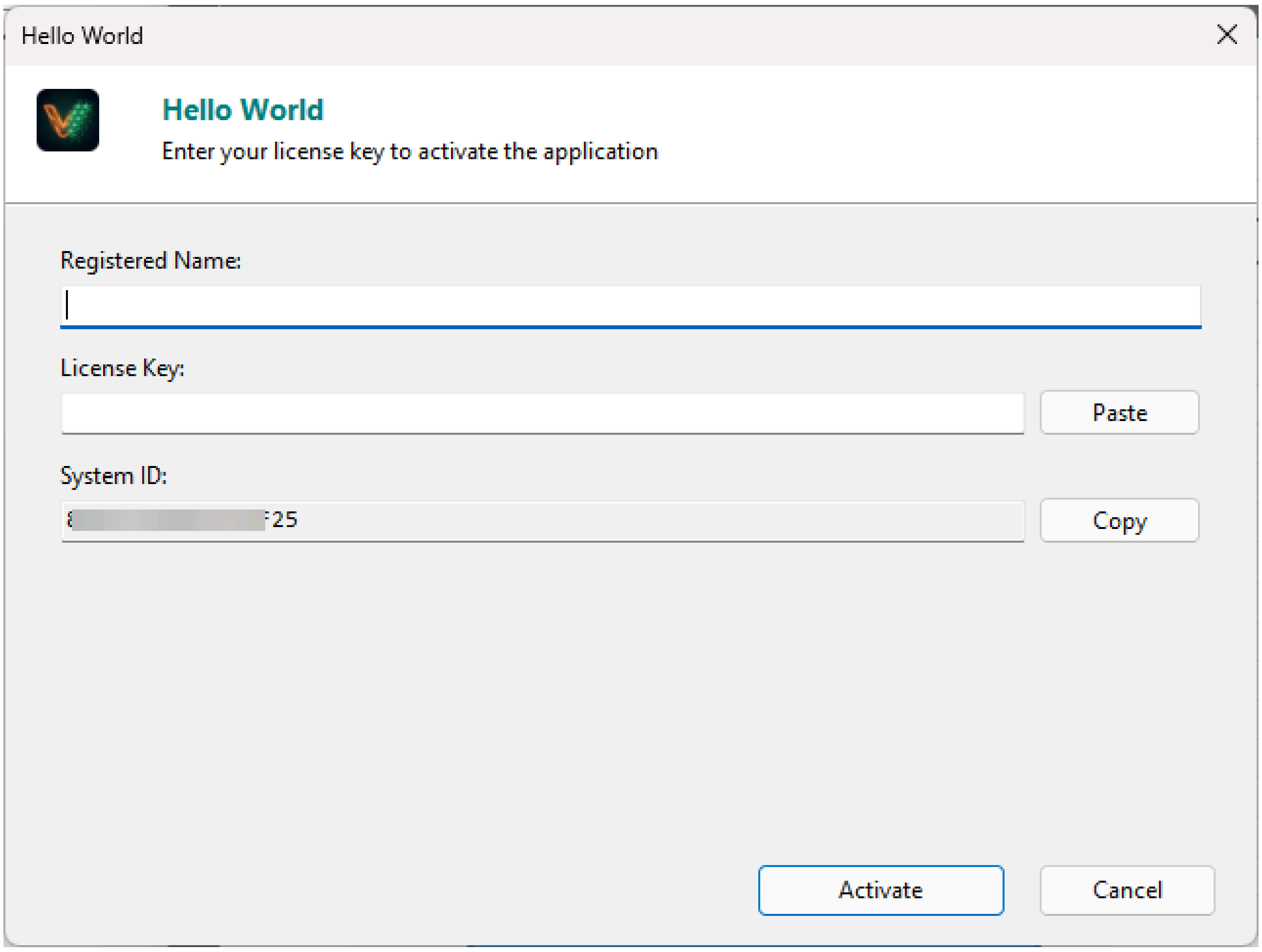
Only if you want automated online activation. You can also generate and distribute keys manually using the Key Generator. If you do want a server, VBA Padlock provides a PHP activation kit. See Deploy the Activation Server.

57.6.23 What are the server requirements?

- PHP 8.1+
- MySQL / MariaDB
- Apache with `mod_rewrite`
- Composer (for PHP dependencies)
- HTTPS (SSL certificate required)

57.6.24 Can users activate offline?

Yes. If online activation fails or is not available, users can use manual activation. They provide their Hardware ID, and you generate a key for them. See Key Generation guide.



Hello World

Hello World

Enter your license key to activate the application

Registered Name:

License Key:

System ID: [redacted]25

57.7 Still Have Questions?

⚠ Troubleshooting

Fix the most common issues quickly. Open Troubleshooting →

UI Reference

Browse every dialog and option in detail. Open UI Reference →

Support

Need direct help from our team? Contact support →

58. Contact Support

Need help with VBA Padlock? Here are the ways to reach us.

58.1 Support Options

Email Support

Send your question to our support team.

support@vbapalock.com

Documentation

Browse the complete documentation.

View documentation →

FAQ

Check common questions first.

Read the FAQ →

Troubleshooting

Solutions to common issues.

View troubleshooting →

58.2 When Contacting Support

To help us resolve your issue quickly, please include:

1. **VBA Padlock version** — found in VBA Padlock Studio under **Help** → **About**.
2. **Office version** — Excel/Word/Access/PowerPoint, 32-bit or 64-bit.
3. **Windows version** — e.g., Windows 10 22H2, Windows 11 23H2.
4. **Error messages** — exact text or screenshots of any error dialogs.
5. **Steps to reproduce** — what you did before the issue occurred.

Tip

If the issue is a compilation error, include the relevant Messages panel output. If it's a licensing issue, include the results of `VBAPL_IsLicenseValid()` and `VBAPL_IsTrialMode()`.

58.3 Useful Resources

Before contacting support, you may find your answer in:

- Troubleshooting — Common issues and solutions
- FAQ — Frequently asked questions
- License Return Codes — All error code meanings
- VBA Compatibility — Supported VBA features