A New Licensing Tool for Compiled COMSOL Applications

Daniel Ericsson[1], Ali Abida[2]

[1] Deflexional, Stockholm, Sweden
[2] Abida Consulting, Stockholm, Sweden

Contents

About Us

- Deflexional
  - 25+ years with COMSOL Multiphysics®
  - Certified Consultant, specialized in COMSOL App development
- Abida Consulting
  - 25+ years in coding
  - Extensive expertise in Java development
- Our partnership combines COMSOL expertise with advanced software development

Background

- The *Application Programming Guide* explains how to password protect standalone compiled apps
  - Limitation: Each new password or MAC-address requires recompiling the app
- The software HETT^{22} was released in 2022[1]
  - Certain parts of the code needed protection
  - Creating separate compiled apps for thousands of users was impossible
- A new solution was needed
  - Development of the Deflexional License Manager
  - Now available to all COMSOL app developers

1. D. Ericsson et al., COMSOL Conference 2023 Munich; www.comsol.com/paper/hett22-a-comsol-app-to-accurately-simulate-plan-and-monitor-concrete-castings-122611

deflexional

Deflexional License Manager

Provides COMSOL App developers with a powerful yet simple way to integrate licensing directly into their applications

deflexional.com/license-manager

Key Benefits

- Generate license keys via a web interface
- Lock license keys to MAC addresses
- Select license options: time-limited, trial, paid, version, …
- Serve all users with a single compiled app

Key Benefits

- Validate license keys directly within a COMSOL App
- Store license keys securely in the cloud
- Protect sensitive data with file and text encryption
- Follow detailed setup instructions and examples
- Test all features at no cost with a free account

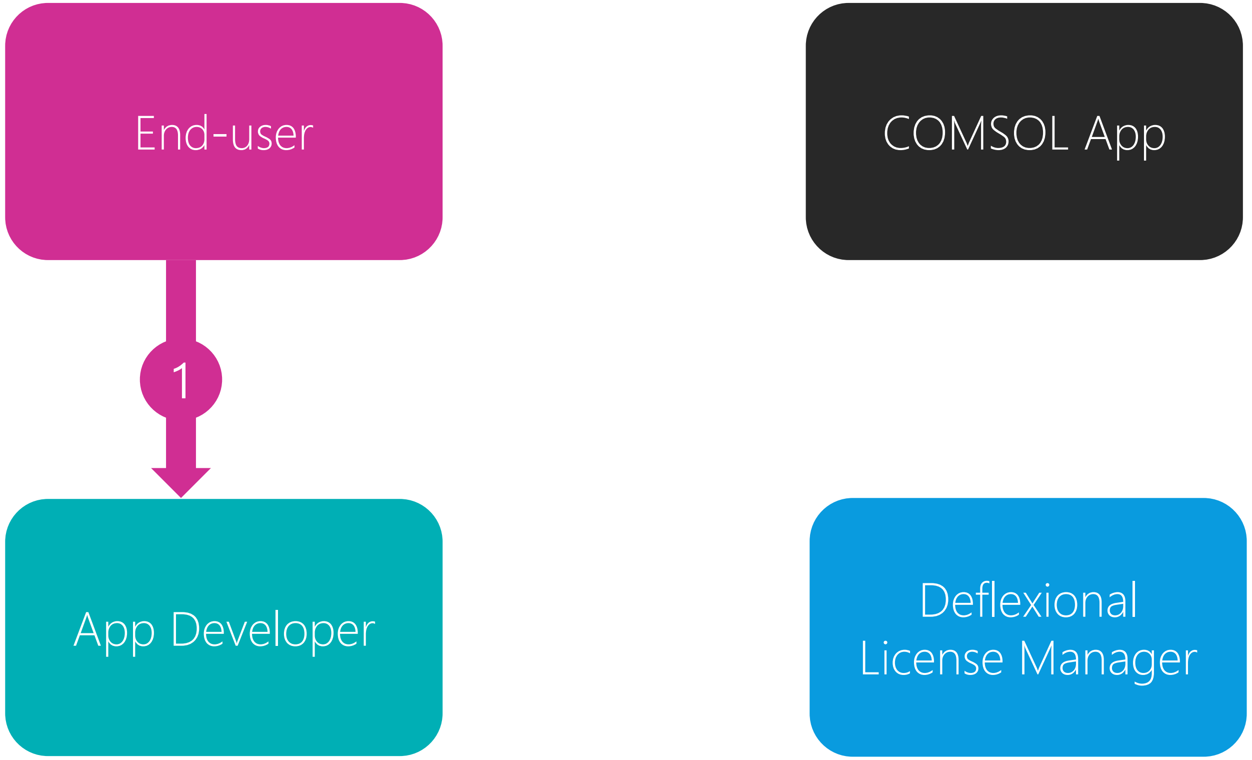
License Keys Workflow

Definitions

COMSOL App	A stand-alone application built using COMSOL Multiphysics® and COMSOL Compiler™
App Developer	The user of COMSOL Multiphysics® who develops and compiles the COMSOL Apps
End-user	The individual utilizing the compiled COMSOL App
Deflexional License Manager	The platform, where App Developers create and manage license keys for End-users

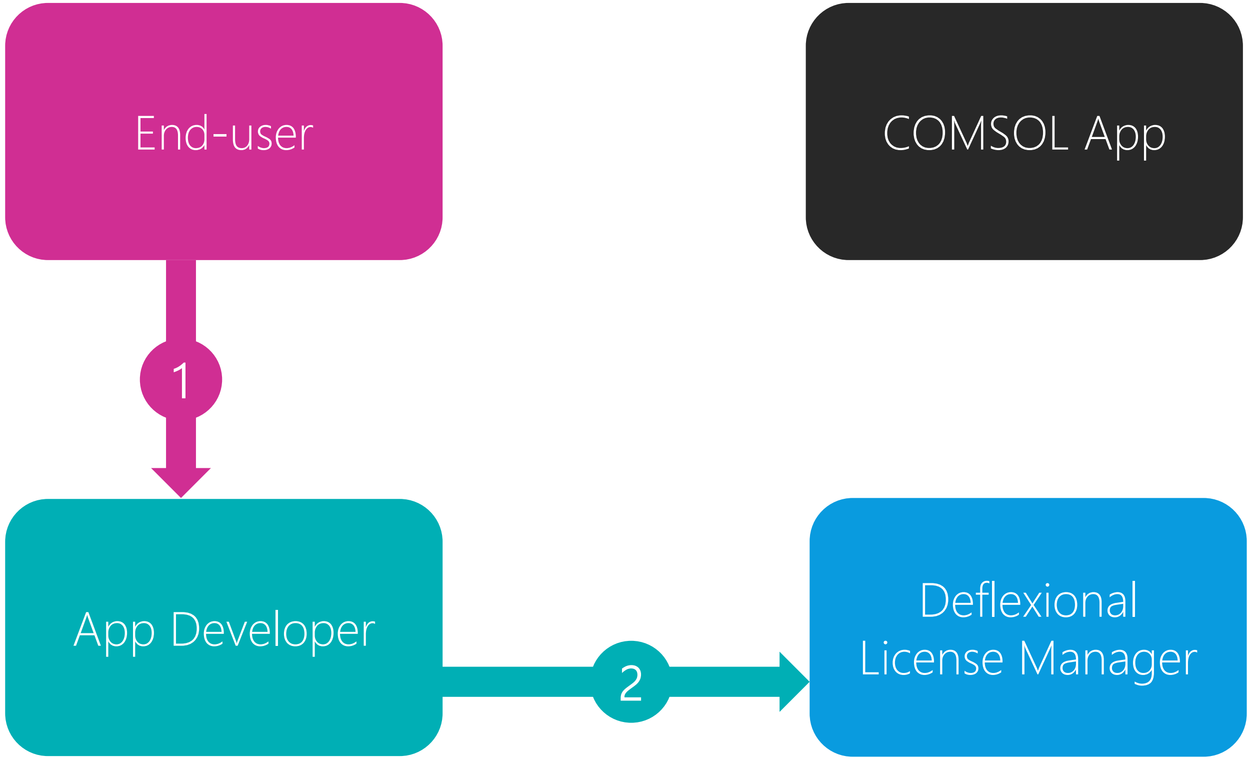
License Key Workflow

1. The End-user shares the MAC address of the computer where the COMSOL App will be used with the App Developer



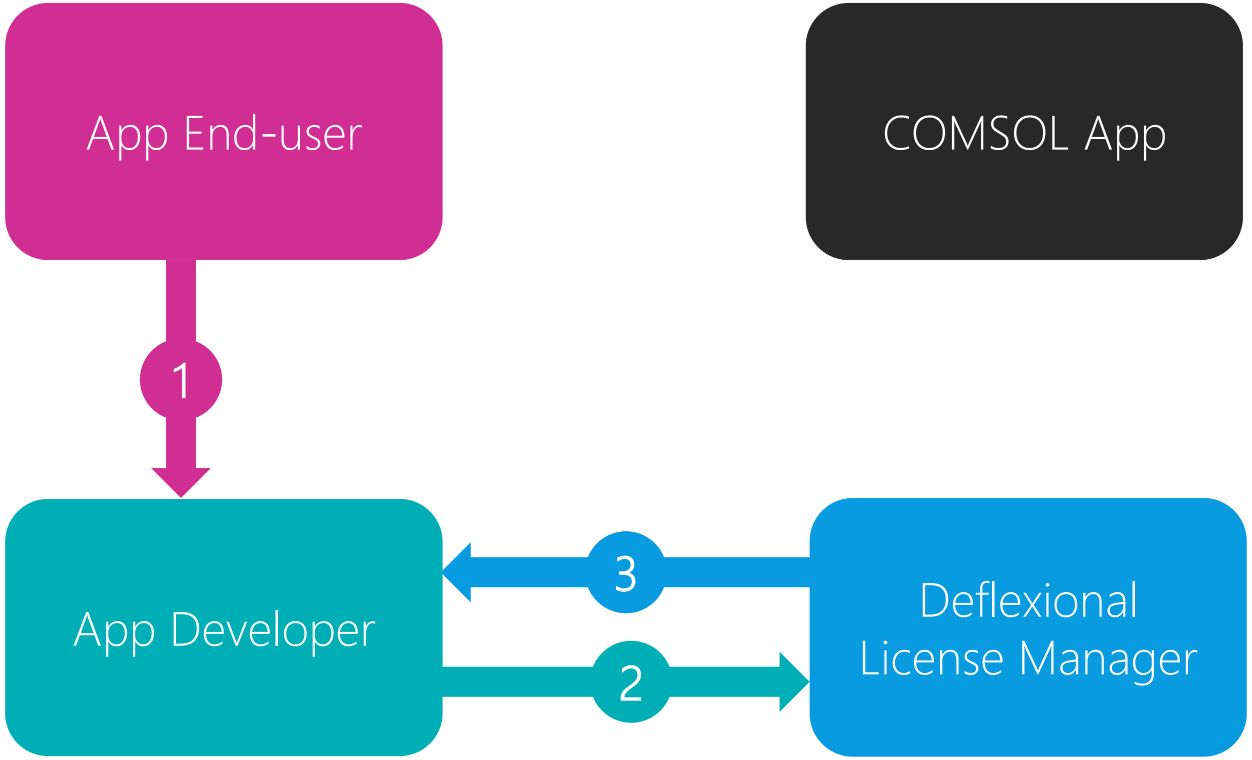
License Key Workflow

2 The App Developer logs into the Deflexional License Manager platform and enters the required details to generate a license key



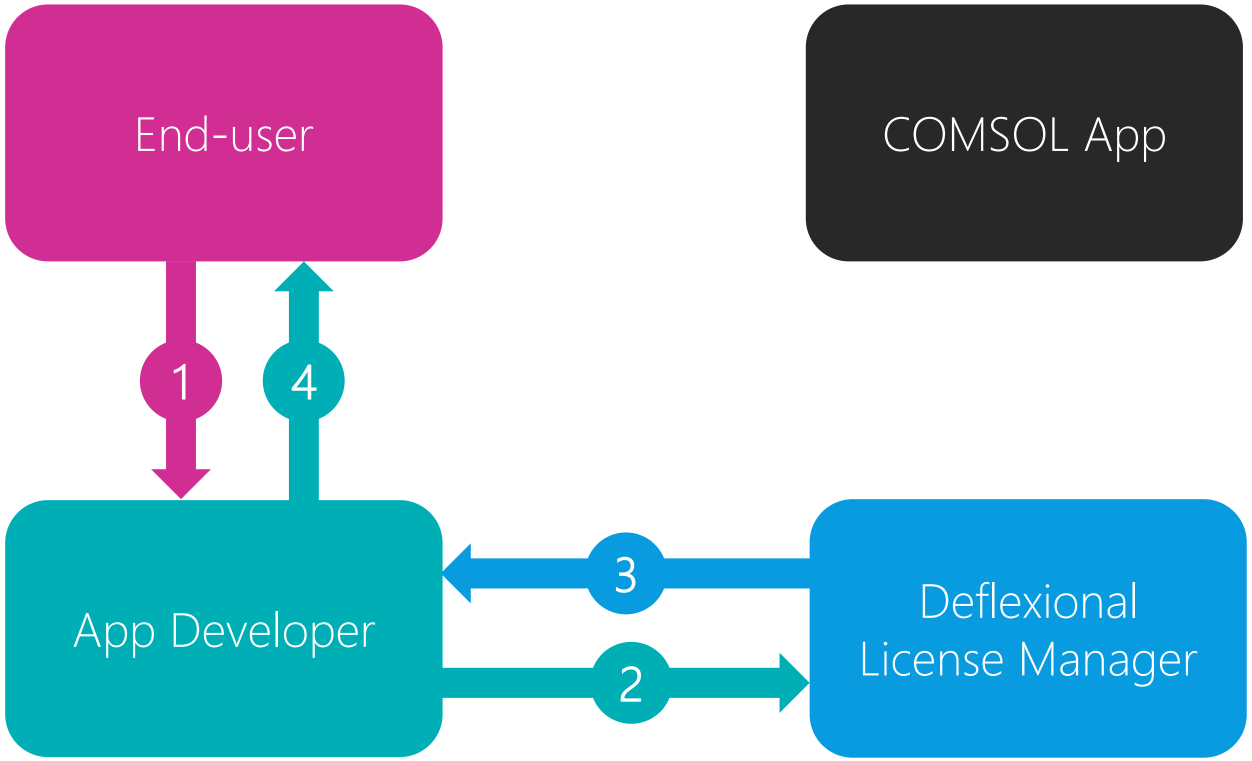
License Key Workflow

3 Using a secret key and the App End-user's details, the Deflexional License Manager generates license keys



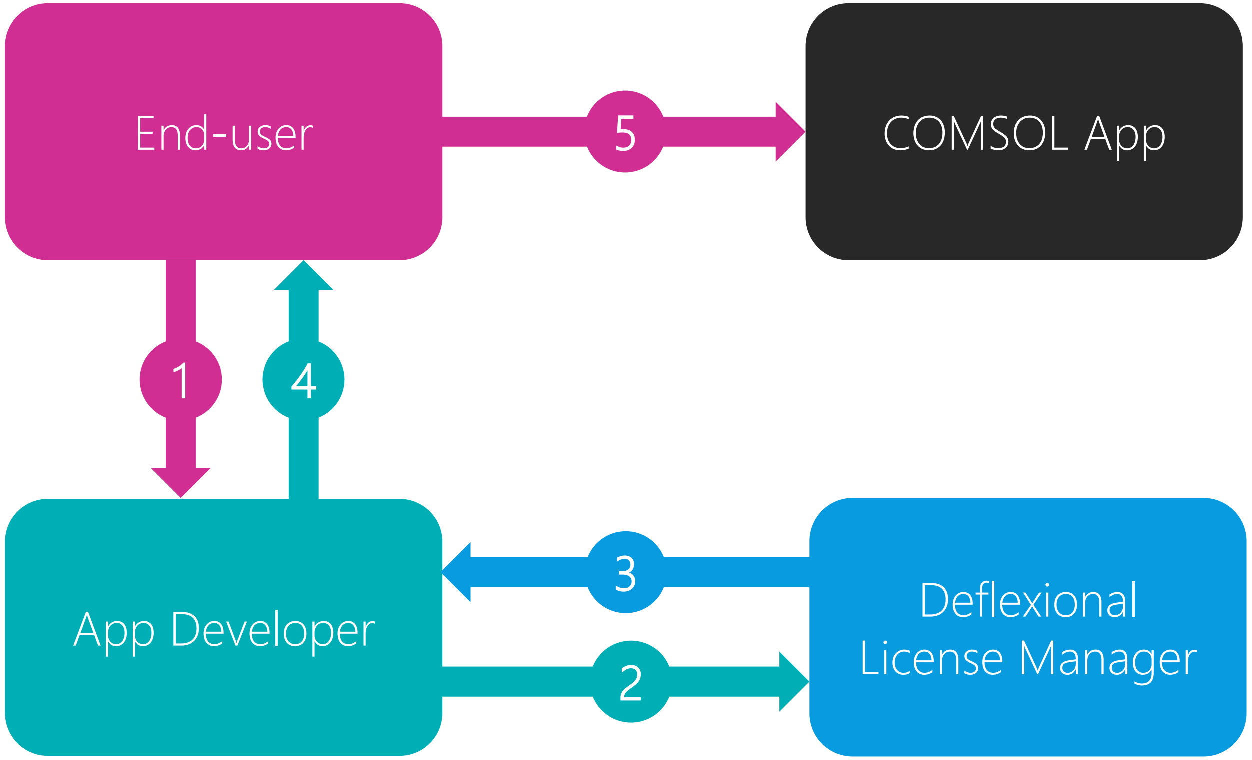
License Key Workflow

4 The App Developer sends the license key to the End-user



License Key Workflow

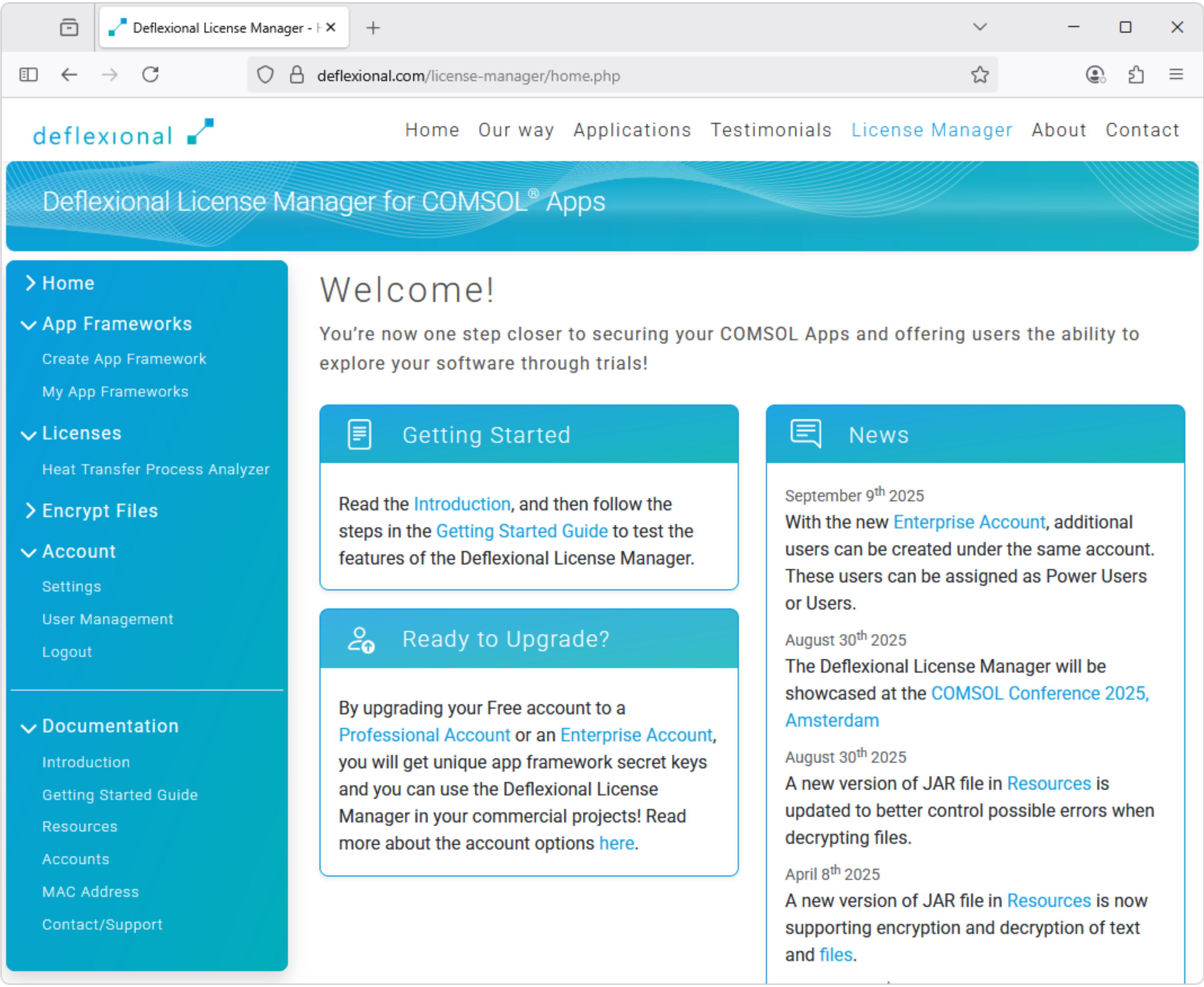
5 The End-user inputs the license key into the COMSOL App to unlock access, and the COMSOL App then verifies the validity of the license key



COMSOL App Integration

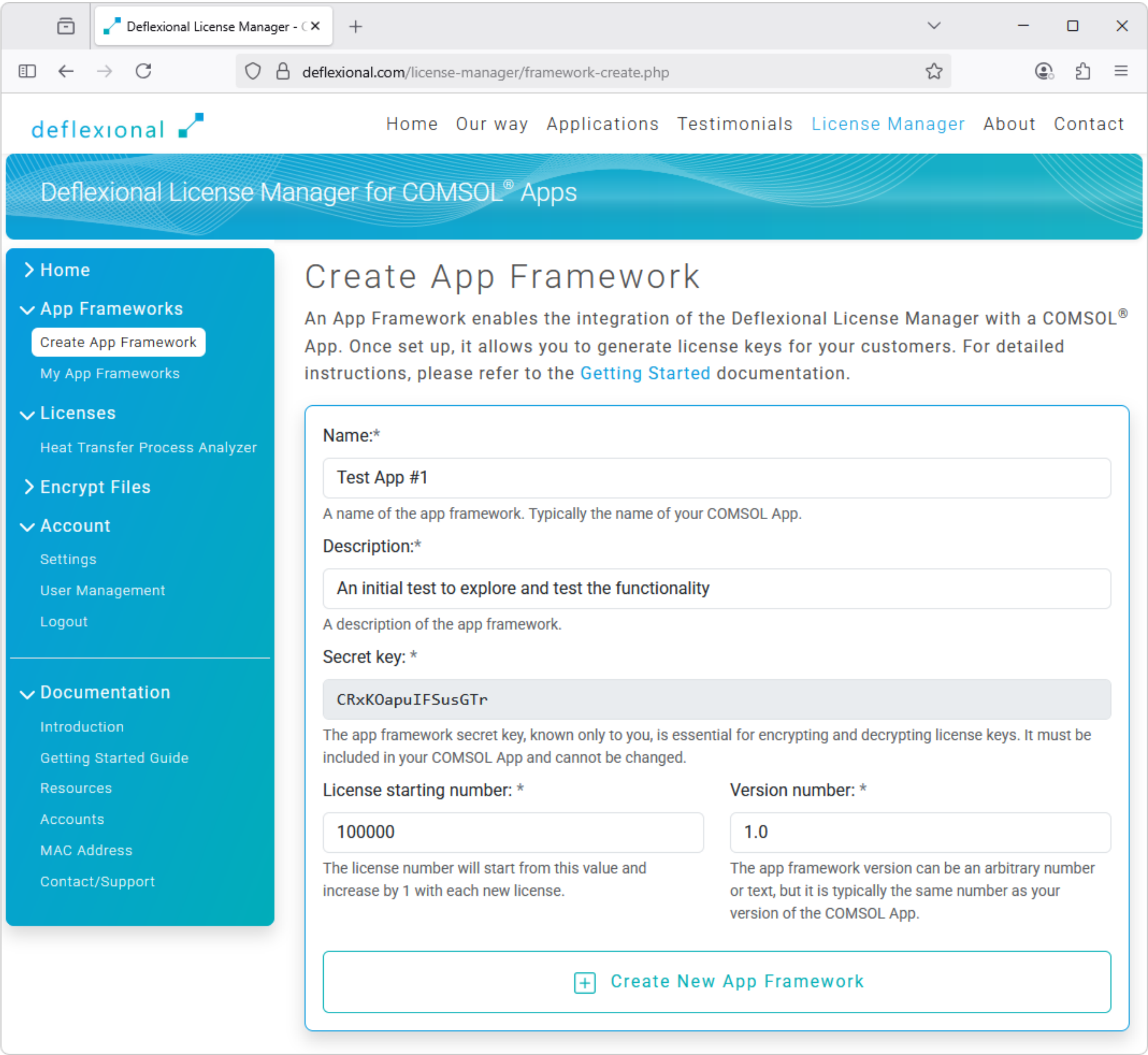
Getting Started

1. Create an account at: deflexional.com/license-manger
2. Login and explore the documentation
   - Introduction
   - Getting Started Guide
3. Download resources and try example apps
4. Integrate with your own COMSOL Apps



Create a New App Framework

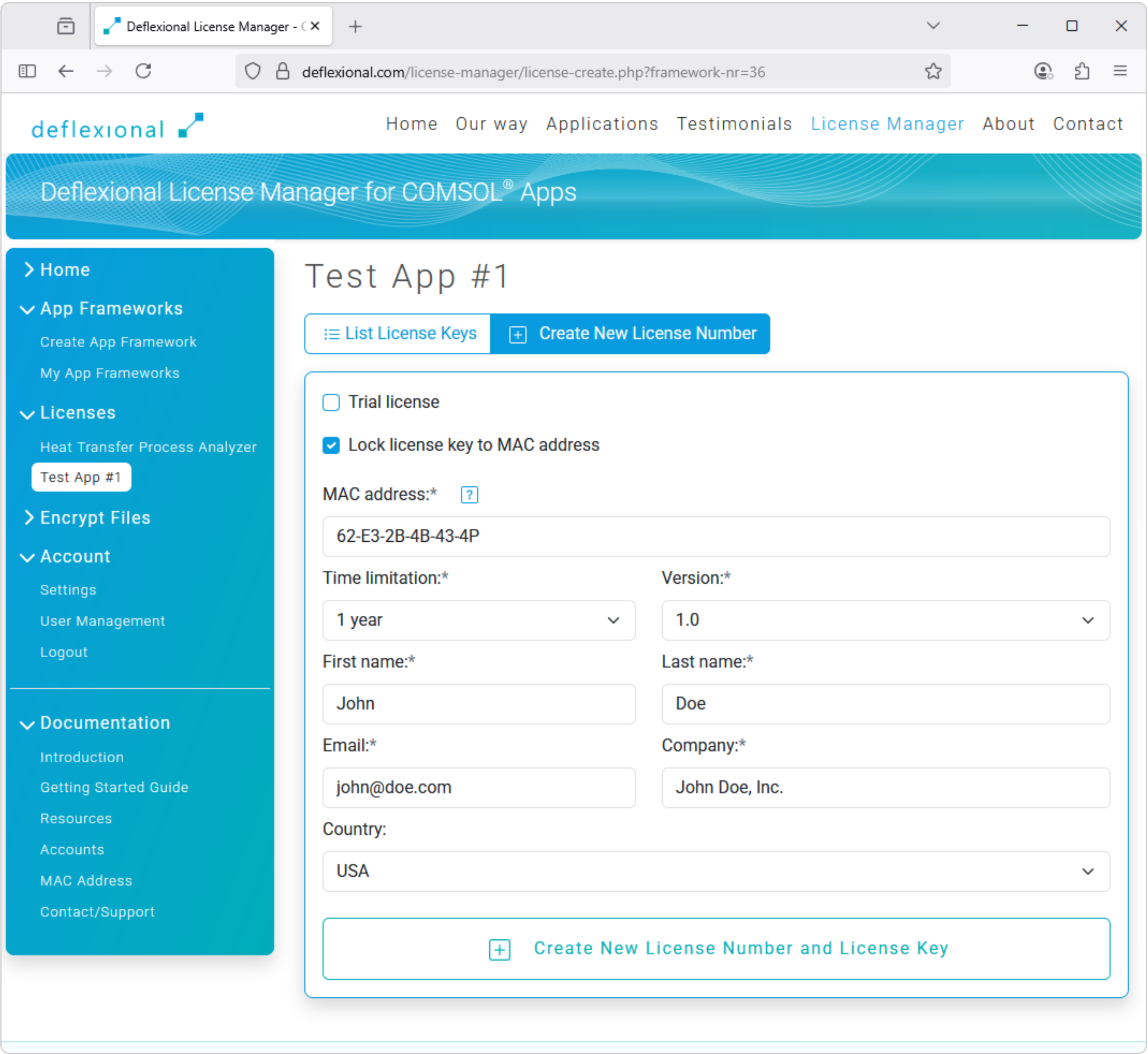
- Fill in the required details and submit the form
- The specific Secret key must later be included in the COMSOL App
- Note: Free Accounts always use the Secret key 11111111111



deflexional

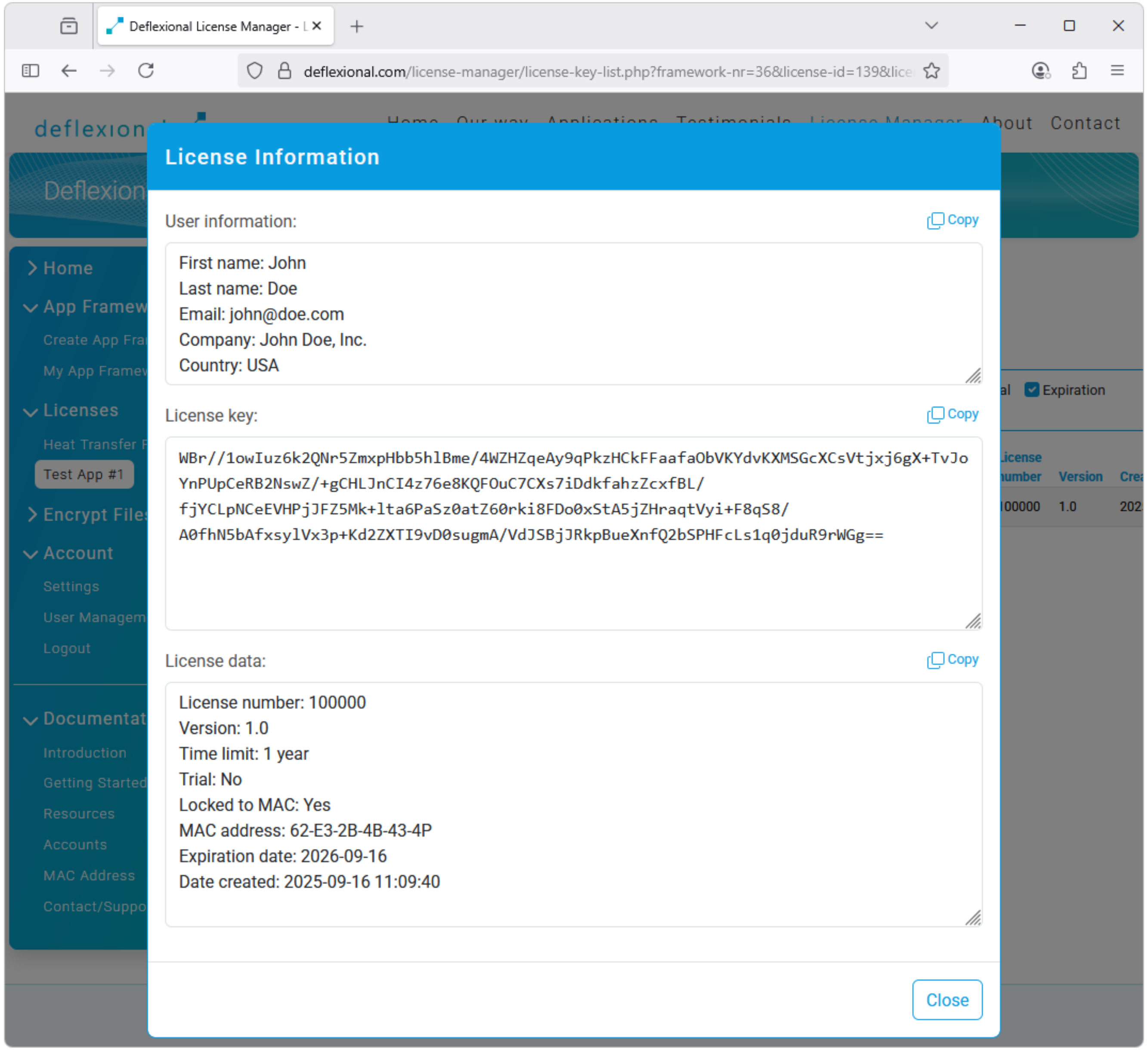
Create a New License Key

- Fill in the required details and submit the form



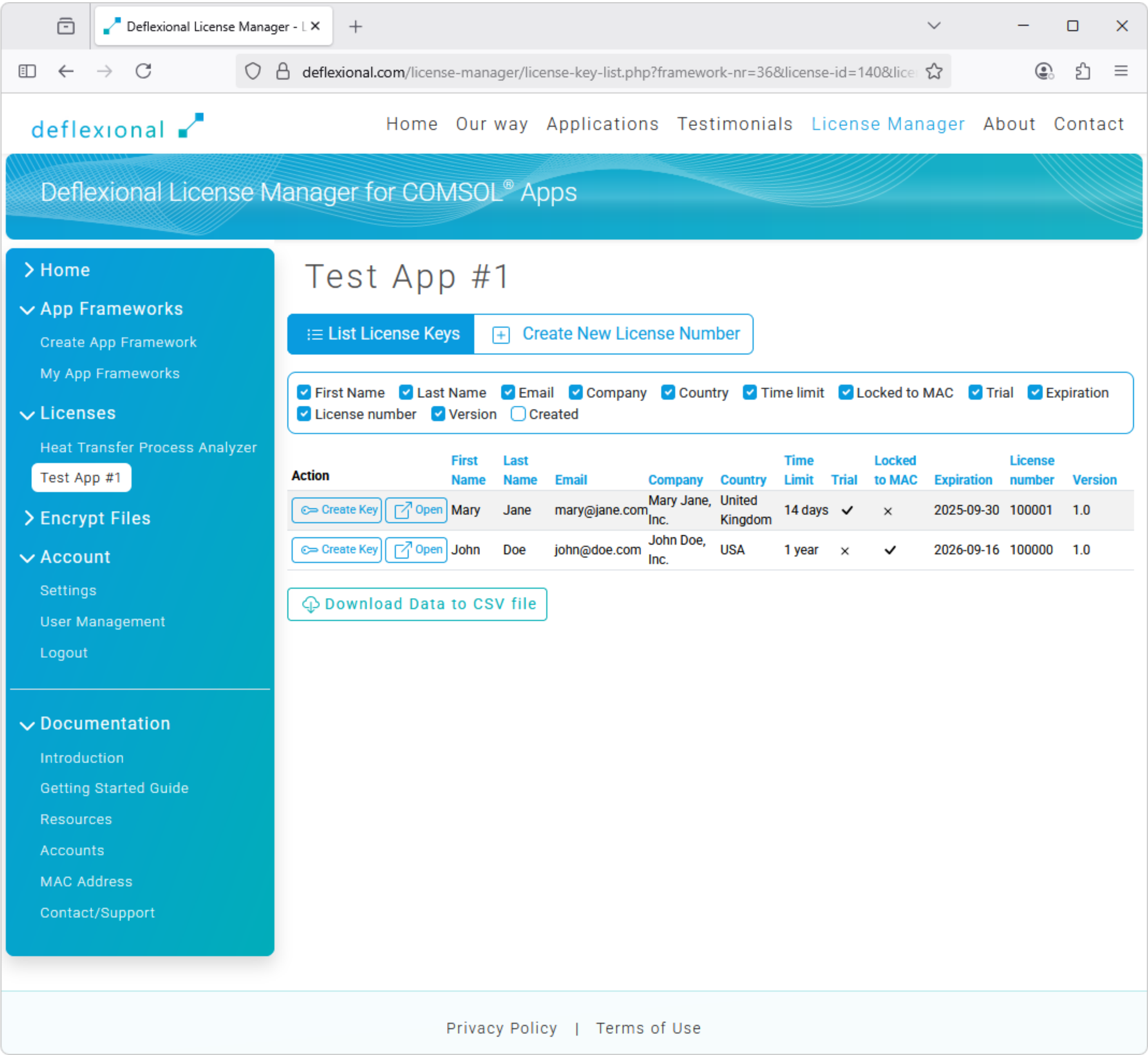
Create a New License Key

- Copy the generated license key



List License Keys

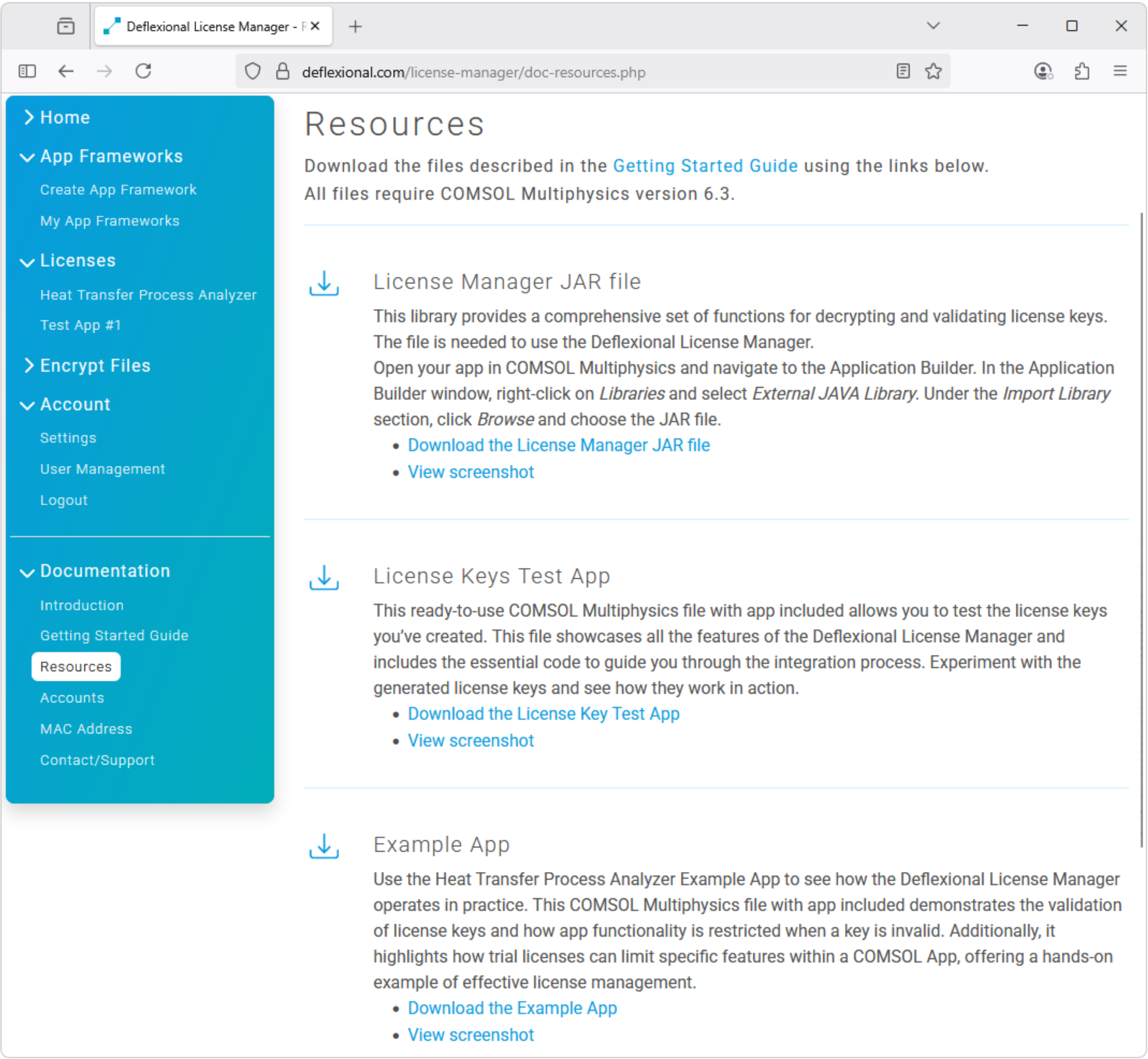
- Sort and filter
- Create new License numbers (new users)
- Create new License keys to an existing License number
- Download a local copy of the database information



deflexional

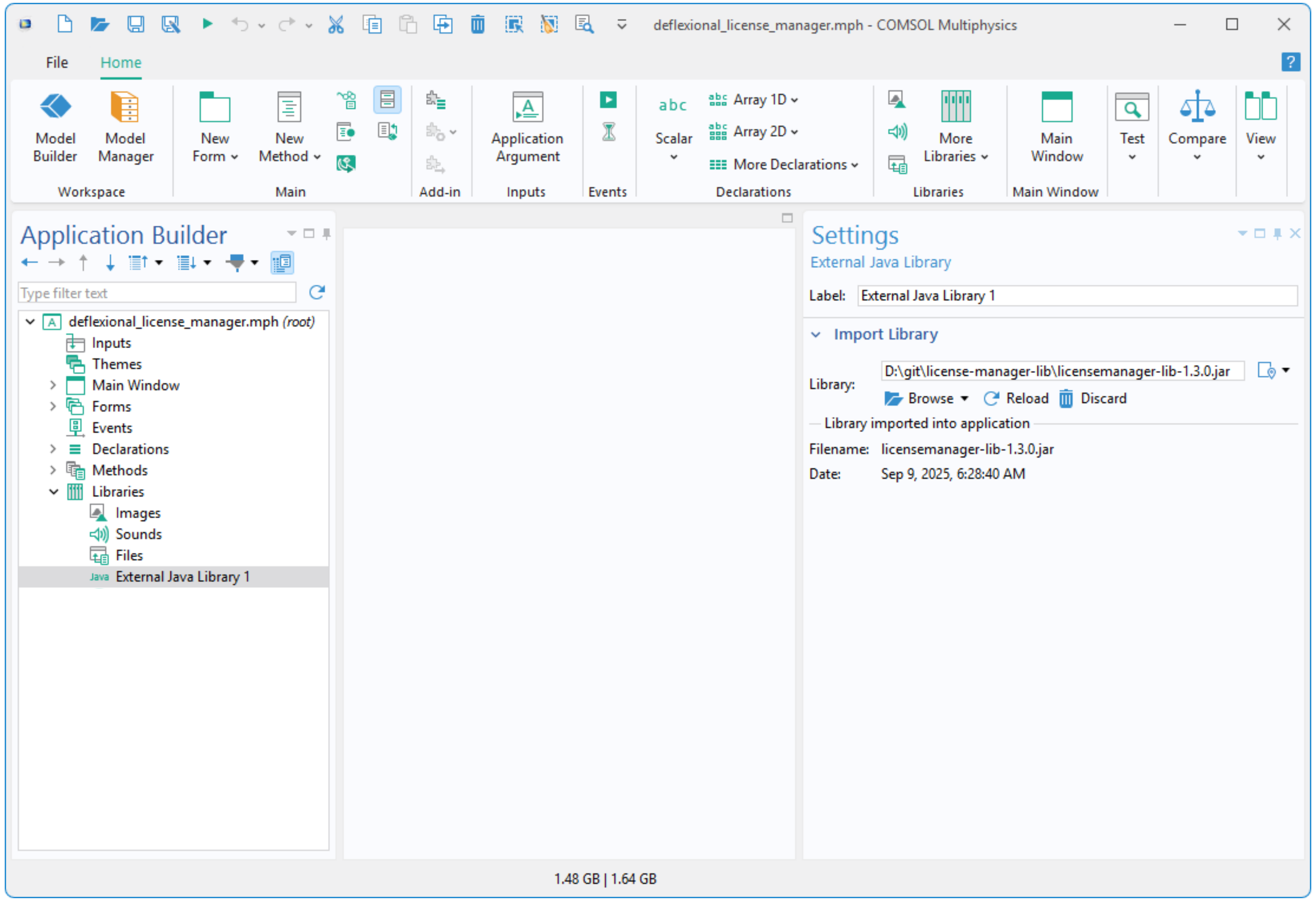
Resources

- License Manager JAR file
- License Keys Test App
- Example App



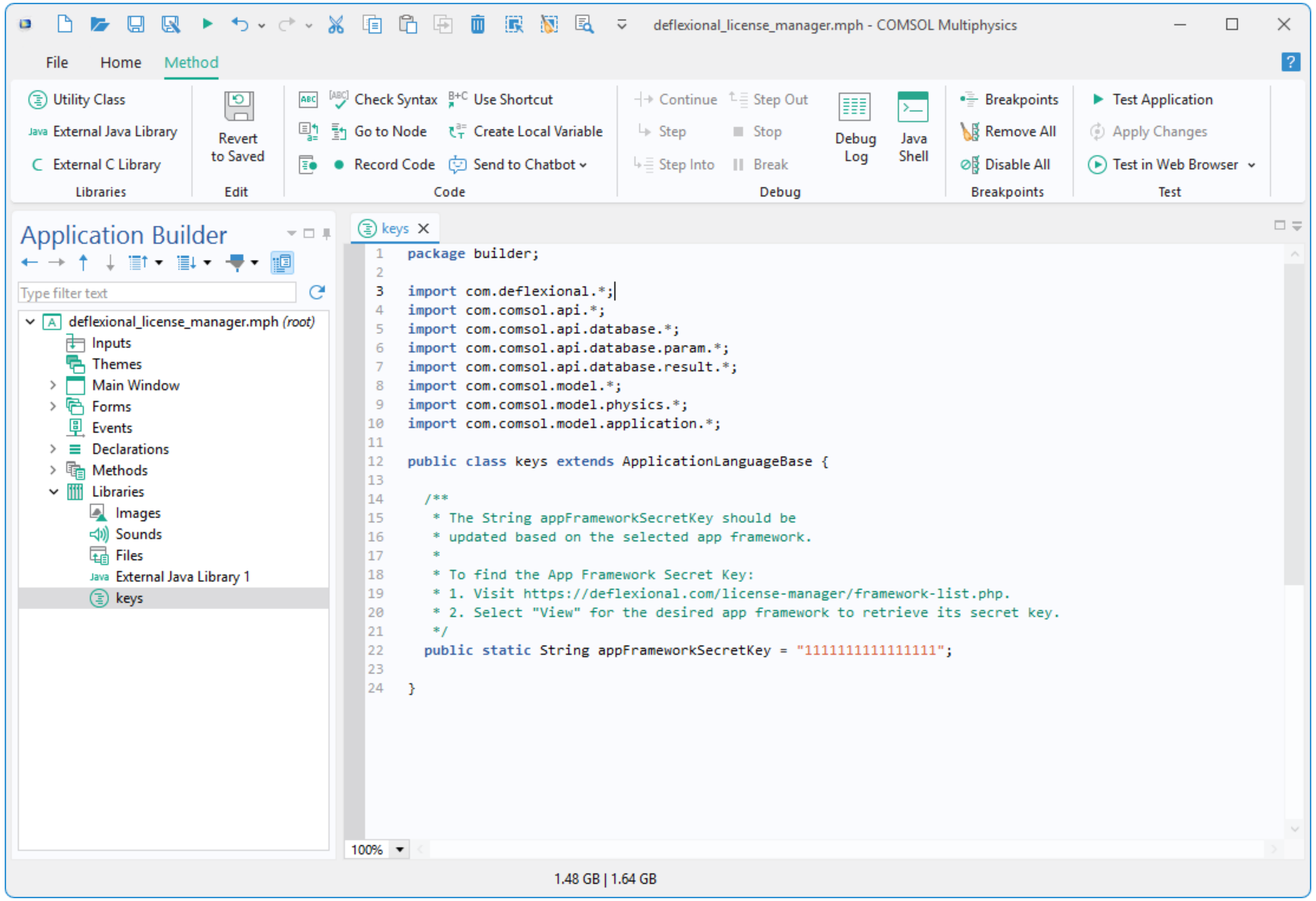
Integrate the App Framework – JAR File

- Import the License Manager JAR file as an *External Java Library* in COMSOL Multiphysics®
- Once the library has been imported, you'll have access to several useful methods in the *LicenseManager* and *EncryptionManager* classes



Integrate the App Framework – Secret Key

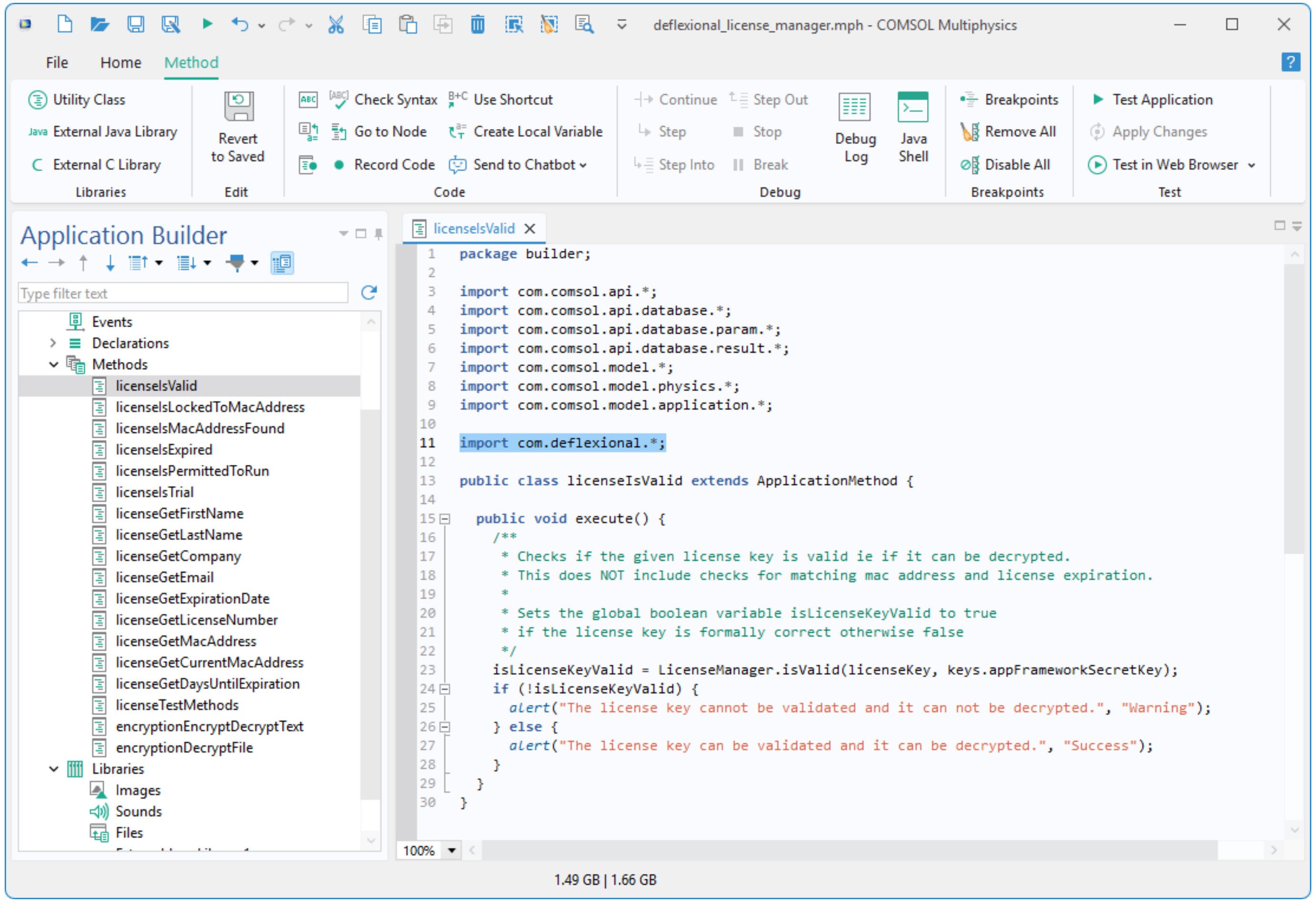
- Add the App Framework Secret key to the COMSOL App
- This key will be used as input to the methods in the *LicenseManager* class



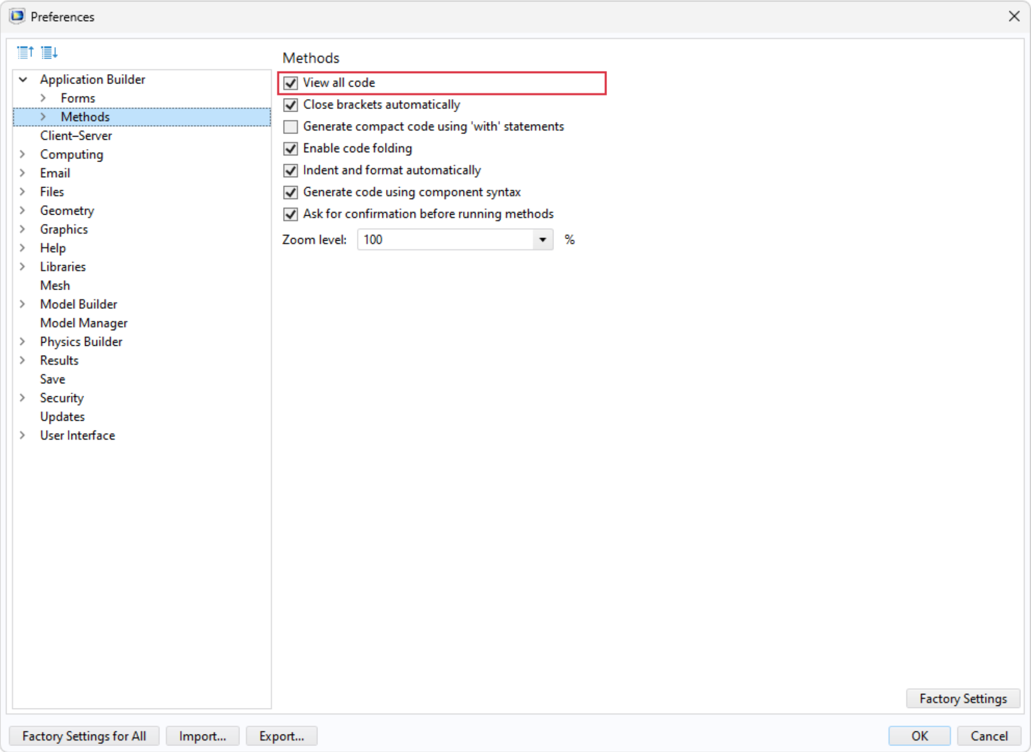
Integrate the App Framework – Code

- Create methods in the Application Builder that calls the *LicenseManager* class

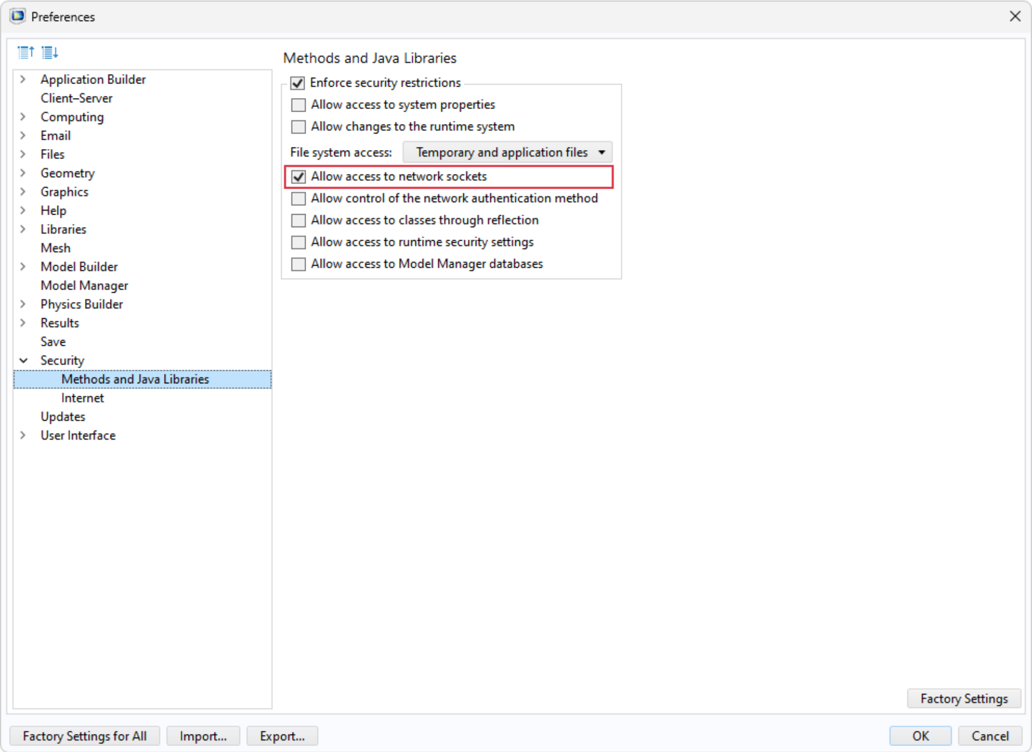
- Ensure the method includes code to import the *LicenseManager* class



Update the Preferences in COMSOL®



Enable the *View all code* option to allow the use of import statements

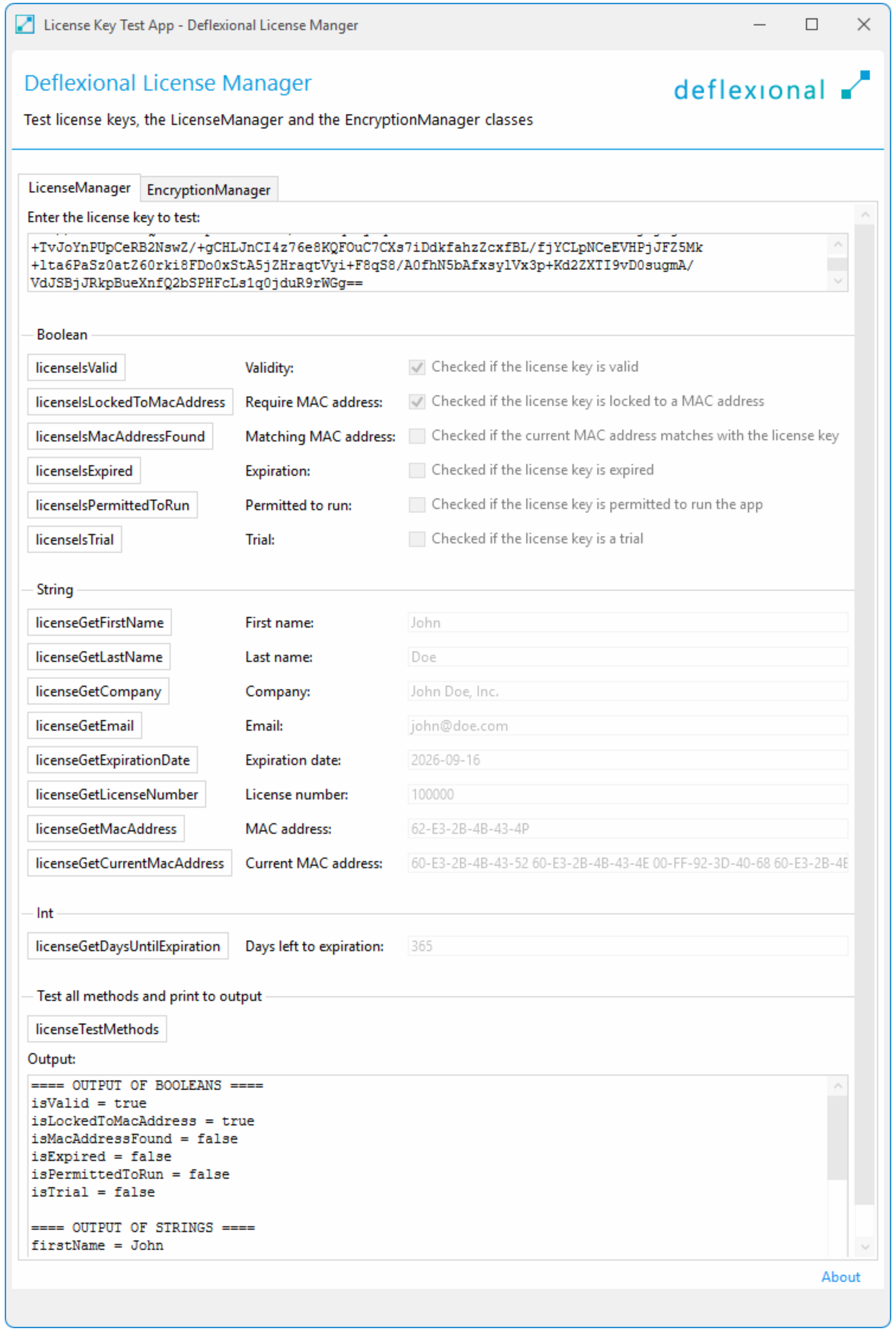


Enable *Allow access to network sockets* to permit MAC address verification when running applications

deflexional

License Key Test App

- Public methods of the *LicenseManager* class are documented and tested in the License Key Test App



LicenseManager Methods

Boolean

- isValid
- isLockedToMacAddress
- isValid
- isPermittedToRun
- isTrial

String

- firstName
- lastName
- company
- email
- expirationDate
- licenseNumber
- macAddresses
- currentMacAddresses

Integer

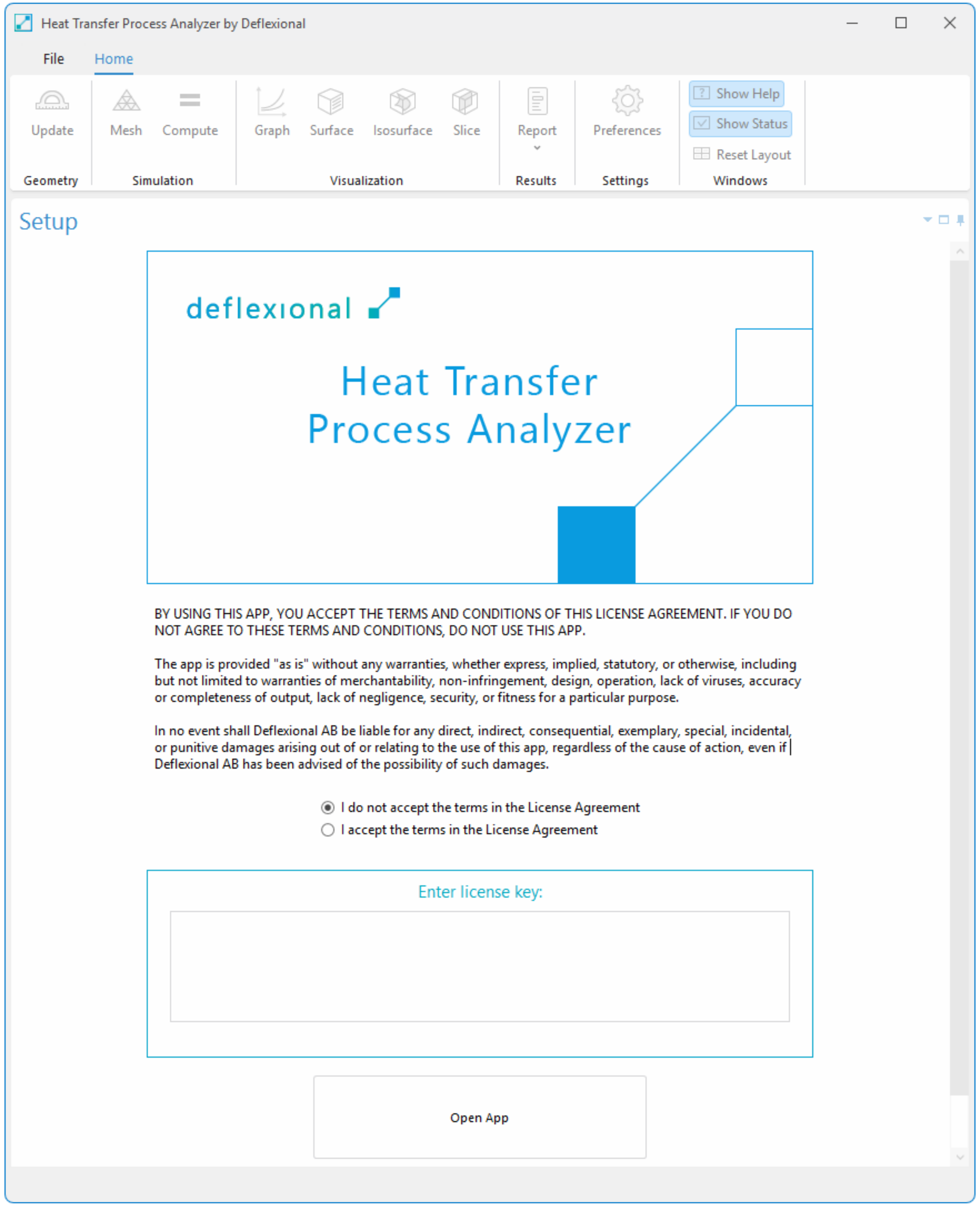
- daysUntilExpiration

Example:

```
Boolean isLicensePermittedToRun = LicenseManager.isPermittedToRun(licenseKey, keys.appFrameworkSecretKey);
```

Example App

- Demonstrates license key validation and how app functionality is restricted for invalid keys
- Highlights feature limitations with trial licenses
- Decrypts sensitive material data from an encrypted file using the *EncryptionManager* class



deflexional

Account Options

Accounts - Free

FREE	PROFESSIONAL	ENTERPRISE
€0/year	€495/year	€995/year
✓ Generate license keys	✓ Generate license keys	✓ Generate license keys
✓ Bind keys to MAC addresses	✓ Bind keys to MAC addresses	✓ Bind keys to MAC addresses
✓ Set expiration dates for keys	✓ Set expiration dates for keys	✓ Set expiration dates for keys
✓ Create trial or paid versions	✓ Create trial or paid versions	✓ Create trial or paid versions
✓ Secure cloud storage for keys	✓ Secure cloud storage for keys	✓ Secure cloud storage for keys
✓ Validate keys in COMSOL Apps	✓ Validate keys in COMSOL Apps	✓ Validate keys in COMSOL Apps
✓ Encrypt/decrypt text and files	✓ Encrypt/decrypt text and files	✓ Encrypt/decrypt text and files
✓ Access integration JAR-file	✓ Access integration JAR-file	✓ Access integration JAR-file
✓ Instructions and example files	✓ Instructions and example files	✓ Instructions and example files
✗ Internal testing use only	✓ Professional use	✓ Professional use
✗ No custom *secret key*	✓ Unlimited *secret keys*	✓ Unlimited *secret keys*
✗ Only one user	✗ Only one user	✓ Multiple users

The app framework secret key is fixed at "1111111111111111", making it easily decryptable.
License keys are intended for internal testing only.

Accounts - Professional

FREE	PROFESSIONAL	ENTERPRISE
€0/year	€495/year	€995/year
✓ Generate license keys	✓ Generate license keys	✓ Generate license keys
✓ Bind keys to MAC addresses	✓ Bind keys to MAC addresses	✓ Bind keys to MAC addresses
✓ Set expiration dates for keys	✓ Set expiration dates for keys	✓ Set expiration dates for keys
✓ Create trial or paid versions	✓ Create trial or paid versions	✓ Create trial or paid versions
✓ Secure cloud storage for keys	✓ Secure cloud storage for keys	✓ Secure cloud storage for keys
✓ Validate keys in COMSOL Apps	✓ Validate keys in COMSOL Apps	✓ Validate keys in COMSOL Apps
✓ Encrypt/decrypt text and files	✓ Encrypt/decrypt text and files	✓ Encrypt/decrypt text and files
✓ Access integration JAR-file	✓ Access integration JAR-file	✓ Access integration JAR-file
✓ Instructions and example files	✓ Instructions and example files	✓ Instructions and example files
✗ Internal testing use only	✓ Professional use	✓ Professional use
✗ No custom *secret key*	✓ Unlimited *secret keys*	✓ Unlimited *secret keys*
✗ Only one user	✗ Only one user	✓ Multiple users

The secret key is known only to the developer,
making it suitable for commercial projects.

Accounts - Enterprise

FREE

€0/year

- ✓ Generate license keys
- ✓ Bind keys to MAC addresses
- ✓ Set expiration dates for keys
- ✓ Create trial or paid versions
- ✓ Secure cloud storage for keys
- ✓ Validate keys in COMSOL Apps
- ✓ Encrypt/decrypt text and files
- ✓ Access integration JAR-file
- ✓ Instructions and example files

- ✗ Internal testing use only
- ✗ No custom *secret key*
- ✗ Only one user

PROFESSIONAL

€495/year

- ✓ Generate license keys
- ✓ Bind keys to MAC addresses
- ✓ Set expiration dates for keys
- ✓ Create trial or paid versions
- ✓ Secure cloud storage for keys
- ✓ Validate keys in COMSOL Apps
- ✓ Encrypt/decrypt text and files
- ✓ Access integration JAR-file
- ✓ Instructions and example files

- ✓ Professional use
- ✓ Unlimited *secret keys*
- ✗ Only one user

ENTERPRISE

€995/year

- ✓ Generate license keys
- ✓ Bind keys to MAC addresses
- ✓ Set expiration dates for keys
- ✓ Create trial or paid versions
- ✓ Secure cloud storage for keys
- ✓ Validate keys in COMSOL Apps
- ✓ Encrypt/decrypt text and files
- ✓ Access integration JAR-file
- ✓ Instructions and example files

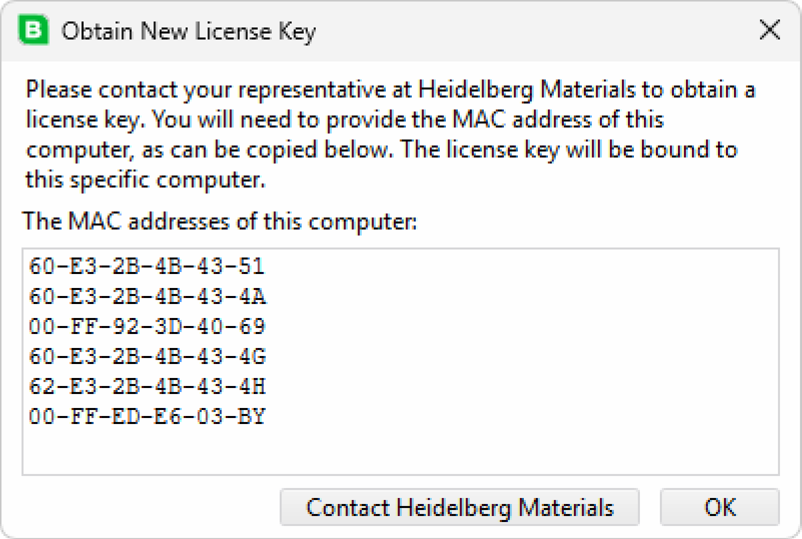
- ✓ Professional use
- ✓ Unlimited *secret keys*
- ✓ Multiple users

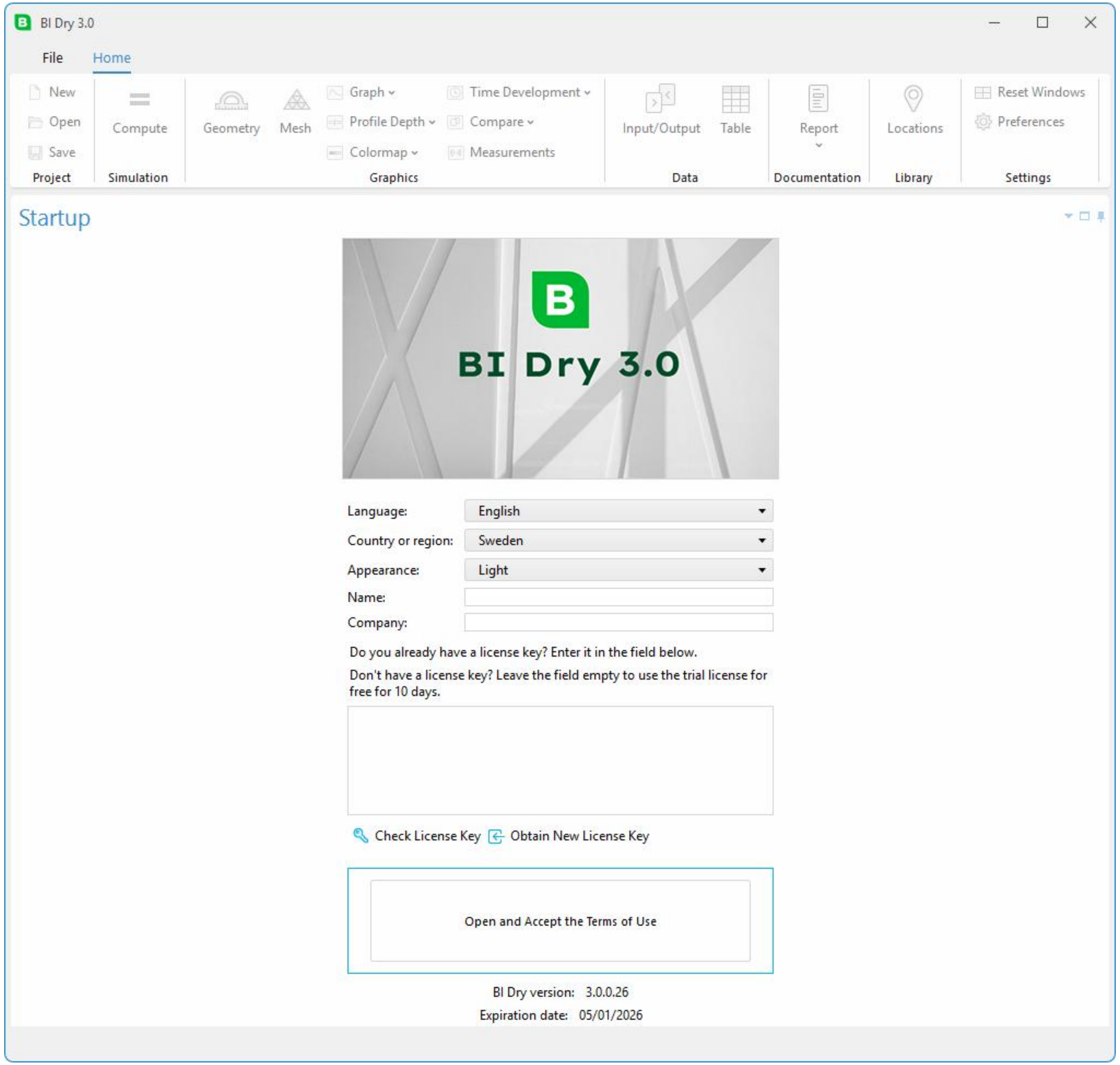
Includes all Professional features and supports multiple users.
The account Administrator can assign users with specific permissions.

Customer Examples

BI Dry 3.0

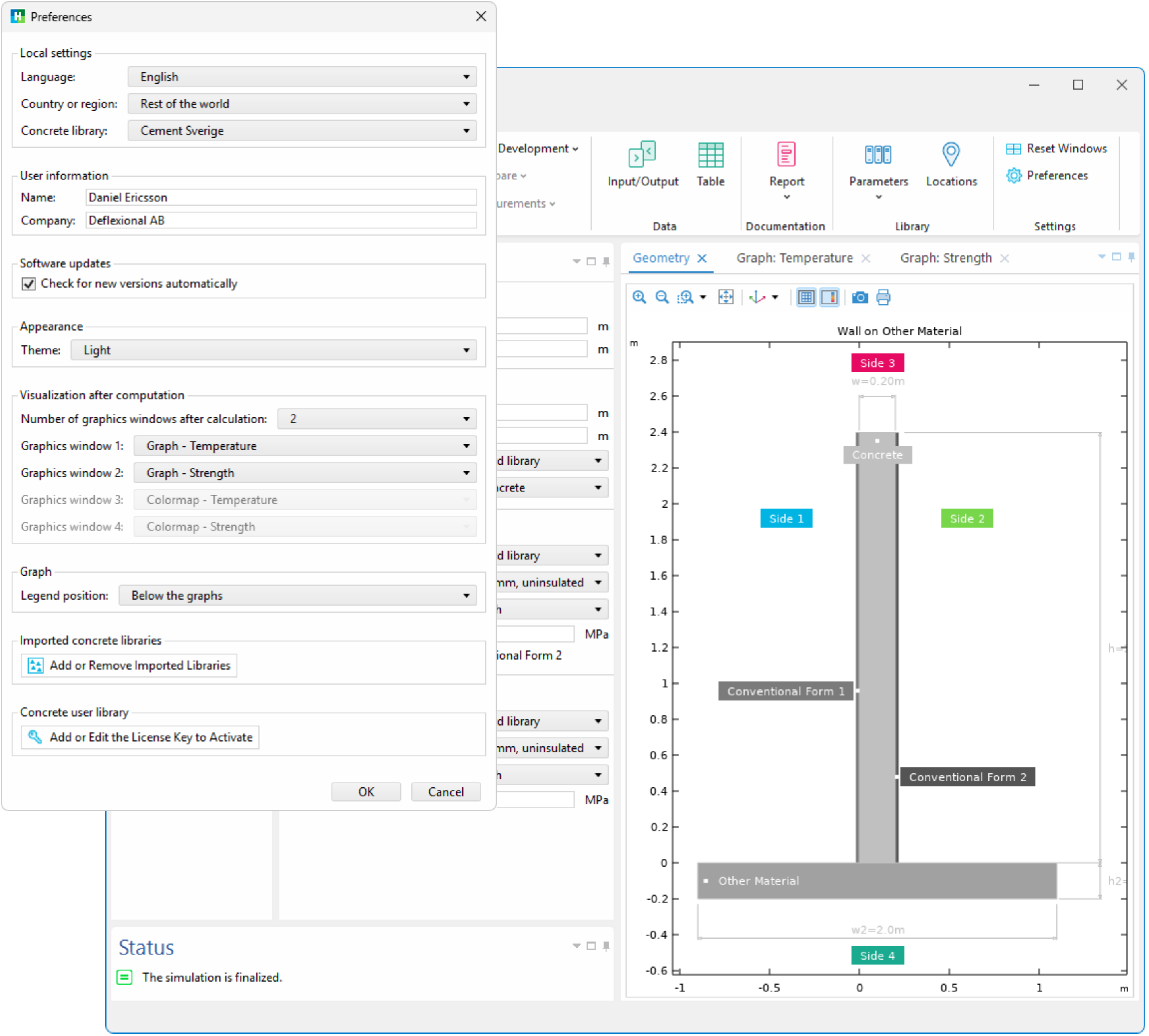
- Trial for 10 days by default
- Users obtain MAC addresses automatically





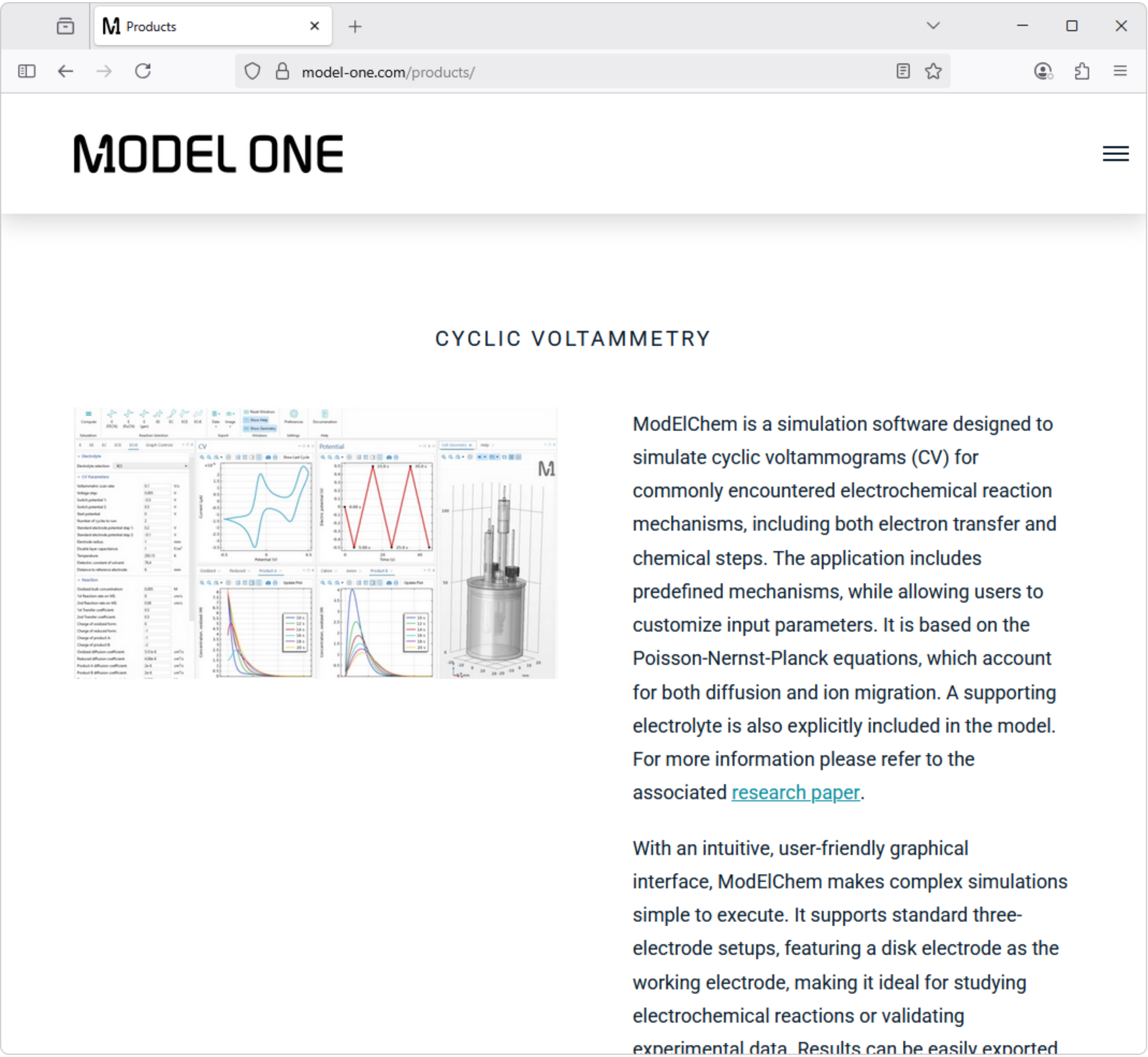
HETT^{22}

- Possibility to import encrypted files for material data with time limits
- A license key is required to access the Concrete user library



Model One

- Distributes its commercial COMSOL apps using the Deflexional License Manager



Summary

By providing a secure, flexible, and easy-to-implement licensing solution tailored for COMSOL developers, the Deflexional License Manager enables app creators to monetize and protect their intellectual property while ensuring full compatibility with the COMSOL ecosystem

deflexional

We maximize the benefits of engineering simulations

Daniel Ericsson
CEO & Founder

daniel.ericsson@deflexional.com
Phone: +46-70-586 00 98
www.deflexional.com

Extra Slides

App Framework

- An App Framework is created by the App Developer and contains the critical data required to integrate the Deflexional License Manager with a COMSOL App.
- While not mandatory, it is recommended to create a separate App Framework for each COMSOL App.
- Each App Framework includes at least one version, known as an *App Framework Version*, which contains a unique *Secret Key* that can be easily integrated into the COMSOL App.

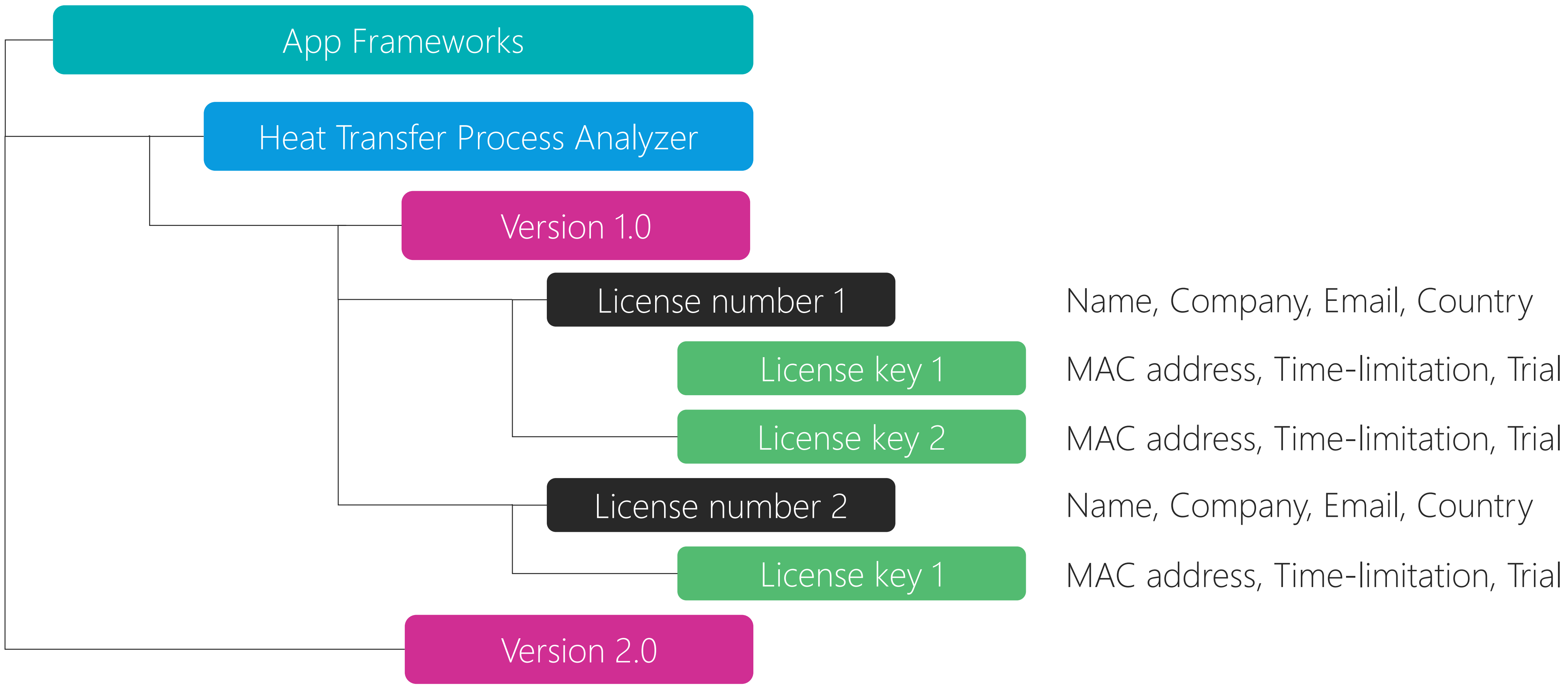
App Frameworks: Example Structure

- App Frameworks
  - Heat Transfer Process Analyzer — Name, Description, License starting number
    - Version 1.0 — Secret key, Version
    - Version 2.0 — Secret key, Version
  - Battery Simulator — Name, Description, License starting number
    - Version 2.0.01 — Secret key, Version
    - Version 2.0.02 — Secret key, Version

Licenses

- A *License Number* stores the end-user's information, including name, company, and email. Each License Number can be associated with multiple License Keys.
- A *License Key* contains details such as time limitations, the MAC address, and whether it's a trial version. It is generated based on the Secret Key from the corresponding App Framework Version.

Licenses: Example Structure



Account Permissions

- Administrator (all accounts)
  - For Free and Professional accounts, there is only one user with Administrator rights. In Enterprise accounts, the user who initially created the account remains the sole Administrator. The Administrator of an Enterprise account has access to User Management and can invite new users to the account and set account permission.
- Power User (Enterprise accounts only)
  - Has the same permissions as the Administrator, except that Power Users cannot add new users or modify account permissions.
- User (Enterprise accounts only)
  - Can create license keys but cannot view, edit, or add app framework secret keys.
- Disabled (Enterprise accounts only)
  - Has no access to the account.

LicenseManager Methods

Boolean

- isValid
- isLockedToMacAddress
- isValid
- isPermittedToRun
- isTrial

String

- firstName
- lastName
- company
- email
- expirationDate
- licenseNumber
- macAddresses
- currentMacAddresses

Integer

- daysUntilExpiration

LicenseManager.isValid

```java
public class licenseIsValid extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key is valid - if it can be decrypted.
     * This does NOT include checks for matching MAC addresses and license expiration.
     *
     * Sets the global boolean variable isLicenseKeyValid to true
     * if the license key is correct otherwise false
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      alert("The license key can be validated and it can be decrypted.", "Success");
    }

  }

}
```

LicenseManager.isLockedToMacAddress

```
public class licenseIsLockedToMacAddress extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key is locked to the user's MAC address.
     * If the license key is not locked to the MAC address, the license key can be used on any computer.
     *
     * Sets the global boolean variable isLicenseLockedToMacAddress to true
     * if the license key is locked to a specific MAC address or specific MAC Addresses.
     */
    isLicenseLockedToMacAddress = LicenseManager.isLockedToMacAddress(licenseKey, keys.appFrameworkSecretKey);
    if (isLicenseLockedToMacAddress) {
      alert("The license key is locked to a MAC address.", "Information");
    } else {
      alert("The license key is not locked to a MAC address.", "Information");
    }
  }

}
```

LicenseManager.isMacAddressFound

```
public class licenseIsMacAddressFound extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key is valid for the computer that executes the code.
     * Runs only if the license key is validated and that the license is locked to a MAC Address.
     *
     * Sets the global boolean variable isLicenseMacAddressFound to true
     * if the license key is valid for the current computer otherwise false
     */

    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      isLicenseLockedToMacAddress = LicenseManager.isLockedToMacAddress(licenseKey, keys.appFrameworkSecretKey);
      if (!isLicenseLockedToMacAddress) {
        alert("The license key is not locked to a MAC address. Cannot check for matching MAC addresses.", "Information");
      } else {
        isLicenseMacAddressFound = LicenseManager.isMacAddressFound(licenseKey, keys.appFrameworkSecretKey);
        if (!isLicenseMacAddressFound) {
          alert("The MAC address in the license key can not be found on this computer.", "Warning");
        } else {
          alert("The MAC address in the license key matches the MAC address of this computer.", "Success");
        }
      }
    }

  }

}
```

LicenseManager.isExpired

```
public class licenseIsExpired extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key has expired or not.
     *
     * Sets the global boolean variable isLicenseExpired to true
     * if the license key has expired otherwise false.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      isLicenseExpired = LicenseManager.isExpired(licenseKey, keys.appFrameworkSecretKey);
      if (isLicenseExpired) {
        alert("The license key has expired.", "Warning");
      } else {
        alert("The license key is not expired.", "Success");
      }
    }

  }

}
```

LicenseManager.isPermittedToRun

```java
public class licenseIsPermittedToRun extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key is valid for app usage or not.
     * Includes checks for validity, matching MAC address and expiration.
     *
     * Sets the global boolean variable isLicenseValid to true
     * if the license key is valid for app usage otherwise false.
     */
    isLicensePermittedToRun = LicenseManager.isPermittedToRun(licenseKey, keys.appFrameworkSecretKey);
    String errorMessage = "The license key is permitted to run.";
    if (!isLicensePermittedToRun) {
      errorMessage = "The license key is not valid. Either it is not a valid license key, "
        +"the matching MAC address is not found on this computer, or the license has expired.";
    }
    alert(errorMessage, "Information");

  }

}
```

LicenseManager.isTrial

```java
public class licenseIsTrial extends ApplicationMethod {

  public void execute() {
    /**
     * Checks if the given license key is a trial or not.
     *
     * Sets the global boolean variable isLicenseTrial to true
     * if the license key is a trial, otherwise to false.
     */
    isLicenseTrial = LicenseManager.isTrial(licenseKey, keys.appFrameworkSecretKey);
    if (isLicenseTrial) {
      alert("The license key is a trial.", "Information");
    } else {
      alert("The license key is not a trial.", "Information");
    }

  }

}
```

LicenseManager.firstName

```
public class licenseGetFirstName extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the first name that the license key is created for.
     *
     * Sets the global string variable licenseFirstName.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseFirstName = LicenseManager.firstName(licenseKey, keys.appFrameworkSecretKey);
      if (licenseFirstName.isEmpty()) {
        alert("The First name can not be decrypted from the license key.", "Warning");
      } else {
        alert("The First name in the encrypted license key is: "+licenseFirstName, "Success");
      }
    }

  }

}
```

LicenseManager.lastName

```
public class licenseGetLastName extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the last name that the license key is created for.
     *
     * Sets the global string variable licenseLastName.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseLastName = LicenseManager.lastName(licenseKey, keys.appFrameworkSecretKey);
      if (licenseLastName.isEmpty()) {
        alert("The Last name can not be decrypted from the license key.", "Warning");
      } else {
        alert("The Last name in the license key is: "+licenseLastName, "Success");
      }
    }

  }

}
```

LicenseManager.company

```
public class licenseGetCompany extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the company name that the license key is created for.
     *
     * Sets the global string variable licenseCompany.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseCompany = LicenseManager.company(licenseKey, keys.appFrameworkSecretKey);
      if (licenseCompany.isEmpty()) {
        alert("The Company name can not be decrypted from the license key.", "Warning");
      } else {
        alert("The Company name in the license key is: "+licenseCompany, "Success");
      }
    }

  }

}
```

LicenseManager.email

```java
public class licenseGetEmail extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the email that the license key is created for.
     *
     * Sets the global string variable licenseEmail.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseEmail = LicenseManager.email(licenseKey, keys.appFrameworkSecretKey);
      if (licenseEmail.isEmpty()) {
        alert("The email can not be decrypted from the license key.", "Warning");
      } else {
        alert("The email in the license key is: "+licenseEmail, "Success");
      }
    }

  }

}
```

LicenseManager.expirationDate

```
public class licenseGetExpirationDate extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the expiration date that the license key is created for.
     *
     * Sets the global string variable licenseExpirationDate.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseExpirationDate = LicenseManager.expirationDate(licenseKey, keys.appFrameworkSecretKey);
      if (licenseExpirationDate.isEmpty()) {
        alert("The expiration date can not be decrypted from the license key.", "Warning");
      } else {
        alert("The expiration date in the license key is: "+licenseExpirationDate, "Success");
      }
    }

  }

}
```

LicenseManager.licenseNumber

```
public class licenseGetLicenseNumber extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the expiration date that the license key is created for.
     *
     * Sets the global string variable licenseLicenseNumber.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseLicenseNumber = LicenseManager.licenseNumber(licenseKey, keys.appFrameworkSecretKey);
      if (licenseLicenseNumber.isEmpty()) {
        alert("The license number can not be decrypted from the license key.", "Warning");
      } else {
        alert("The license number in the license key is: "+licenseLicenseNumber, "Success");
      }
    }

  }

}
```

LicenseManager.macAddresses

```java
public class licenseGetMacAddress extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the MAC address(es) that the license key is created for.
     *
     * Sets the global string variable licenseMacAddress.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      String[] licenseMacAddresses = LicenseManager.macAddresses(licenseKey, keys.appFrameworkSecretKey);
      licenseMacAddress = String.join(" ", licenseMacAddresses);
      if (licenseMacAddress.isEmpty()) {
        alert("The MAC address can not be decrypted from the license key.", "Warning");
      } else {
        alert("The MAC address in the license key is: "+licenseMacAddress, "Success");
      }
    }

  }

}
```

deflexional

LicenseManager.currentMacAddresses

```
public class licenseGetCurrentMacAddress extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the MAC address(es) of this computer.
     *
     * Sets the global string variable currentMacAddress.
     */
    String[] currentMacAddresses = LicenseManager.currentMacAddresses();
    licenseCurrentMacAddress = String.join(" ", currentMacAddresses);
  }

}
```

LicenseManager.daysUntilExpiration

```
public class licenseGetDaysUntilExpiration extends ApplicationMethod {

  public void execute() {
    /**
     * Gets the number of days until expiration.
     *
     * Sets the global int variable licenseDaysUntilExpiration.
     */
    isLicenseKeyValid = LicenseManager.isValid(licenseKey, keys.appFrameworkSecretKey);
    if (!isLicenseKeyValid) {
      alert("The license key cannot be validated and it can not be decrypted.", "Warning");
    } else {
      licenseDaysUntilExpiration = LicenseManager.daysUntilExpiration(licenseKey, keys.appFrameworkSecretKey);
      if (licenseDaysUntilExpiration == 0) {
        alert("The license key has expired.", "Warning");
      } else {
        alert("The license key is not expired, and the number of days until expiration is: "+licenseDaysUntilExpiration, "Success");
      }
    }

  }

}
```

EncryptionManager Methods

String

- encryptText
- decryptText

File

- decryptFile

EncryptionManager.encryptText EncryptionManager.decryptText

```
public class encryptionEncryptDecryptText extends ApplicationMethod {

  public void execute() {
    /**
     * Encrypts and decrypts a string. Uses the global string variable textToEncrypt as input.
     *
     * Sets the global string variables encryptionEncrypted and encryptionDecrypted.
     */

    encryptionEncrypted = EncryptionManager.encryptText(textToEncrypt, keys.appFrameworkSecretKey);
    encryptionDecrypted = EncryptionManager.decryptText(encryptionEncrypted, keys.appFrameworkSecretKey);

  }

}
```

EncryptionManager.decryptFile

```
public class encryptionDecryptFile extends ApplicationMethod {

  public void execute() {
    /**
     * Decrypts a file and lets the user save the decrypted file to a local folder.
     * The file should be encrypted using the web interface of the Deflexional License Manager at:
     * https://deflexional.com/license-manager/encrypt-file.php
     */

    String pathInput = "upload:///file1";
    String fullInputPath = getFilePath(pathInput);
    String[] splitResult = split(fullInputPath, "\\");
    String fileName = replace(splitResult[splitResult.length-1], "file1-", "");
    String[] splitFileExtension = split(fileName, ".");
    String fileExtension = "."+splitFileExtension[splitFileExtension.length-1];
    createDirectory("user:///decrypt/"); //Create a new directory where to place the original file and the decrypted file
    String pathOriginalFile = "user:///decrypt/"+fileName;
    copyFile(fullInputPath, pathOriginalFile);
    String fullPathOriginalFile = getFilePath(pathOriginalFile);
    String fullPathDecryptedFile = replace(fullPathOriginalFile, fileExtension, "."+encryptionOutputFileFormat);
    EncryptionManager.decryptFile(fullPathOriginalFile, fullPathDecryptedFile, keys.appFrameworkSecretKey);
    boolean isSaved = fileSaveAs(fullPathDecryptedFile);
    if (isSaved) {
      alert("The file was decrypted succesfully!");
    } else {
      alert("The file was decrypted succesfully, but not saved to a local directory.");
    }
    //Delete the files from user:///decrypt/
    deleteFile(fullPathOriginalFile);
    deleteFile(fullPathDecryptedFile);
  }
}
```