LicenseTrak

User Guide

Version 1.0a
July 2026

Software by Daughtry

Complexity Made Simple

TABLE OF CONTENTS

VERSION HISTORY

Version	Date	Description
1.0	June 2026	Initial release
1.0a	July 2026	• New option to send generated License.LIC file to customer after license generation • New option to generate an empty/unencrypted License.LIC file for easier importation into a WinDev analysis • Streamlined the Configuration window

CHAPTER 0 - LICENSETRAK AT A GLANCE

LicenseTrak Workflow

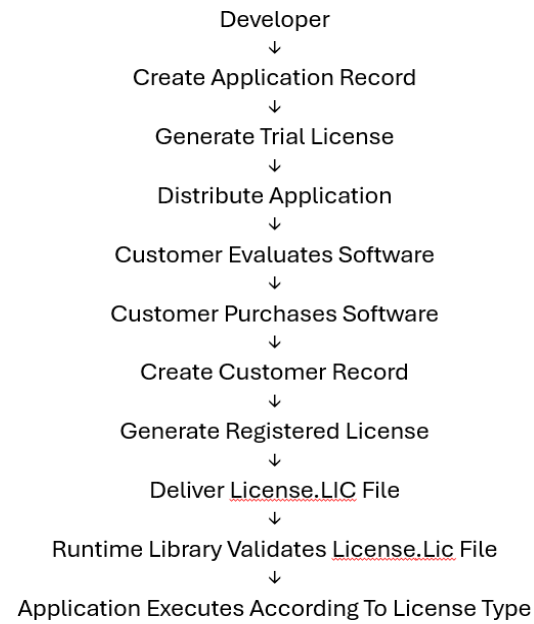
LicenseTrak is designed to provide a simple, self-contained licensing solution for WinDev desktop applications. The typical LicenseTrak workflow begins when a software developer creates an Application record within the LicenseTrak administration program. This record defines the application being protected and serves as the foundation for all future license generation activities.

Once an Application record has been created, the developer typically generates a generic Trial license and includes the resulting License.LIC file with the application's installation package. Prospective customers may then download, install, and evaluate the software using the trial limitations defined by the developer.

If a customer decides to purchase the software, the developer creates a Customer record within LicenseTrak and generates a new License.LIC file specifically for that customer. The registered license file is then delivered to the customer via email or another electronic distribution method.

Each time the protected application is launched, the LicenseTrak Runtime Library validates the License.LIC file and determines whether the software should operate in Trial, Registered, or Locked mode. Additional validation may be performed throughout the application to enforce trial periods, execution limits, subscription expiration dates, or application restrictions.

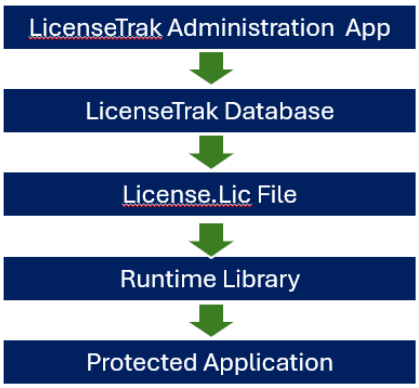
The overall workflow is illustrated below:



This workflow allows software developers to manage application licensing without requiring Internet activation servers, cloud services, recurring subscription fees, or third-party licensing providers.

LicenseTrak Components

LicenseTrak consists of five primary components that work together to provide a complete software licensing solution. Understanding the role of each component will help you successfully deploy and manage LicenseTrak-protected applications:



LicenseTrak Administration Program

The LicenseTrak Administration Program is the central management application used by software developers to administer the licensing process. This application is responsible for maintaining application records, customer records, license information, and support history.

Using the LicenseTrak Administration Program, developers can create trial licenses, generate registered licenses, track customer purchases, document support activity, and manage all aspects of their licensing environment from a single location.

License Database

The License Database stores all information managed by the LicenseTrak Administration Program. This includes application definitions, customer records, issued licenses, support records, and generated license history.

The database serves as the permanent repository for licensing information and provides an audit trail of licenses generated for customers. Regular database backups are strongly recommended to protect this valuable business information.

License.LIC File

The License.LIC file is a special HyperFile database file (*.FIC) generated by the LicenseTrak Administration Program whose file extension is changed to (.lic). This file contains the licensing information required by a protected application.

A License.LIC file may represent a trial license, a registered license, a subscription license, or a locked license. The file is typically distributed with the application installer or delivered directly to a customer after software purchase.

The License.LIC file is the only component required by the end user.

Runtime Library

The LicenseTrak Runtime Library consists of a collection of WinDev procedures that are imported into the developer's application. These procedures perform license validation and provide access to licensing information stored within the License.LIC file.

The Runtime Library determines whether a license is valid and provides functions that allow the application to enforce trial periods, execution limits, subscription expiration dates, and other licensing policies.

Protected Application

The Protected Application is the software created by the developer that incorporates the LicenseTrak Runtime Library.

Each time the application executes, it uses the Runtime Library to validate the License.LIC file and determine how the software should operate. Depending upon the license configuration, the application may execute in Trial mode, Registered mode, or Locked mode.

Together, these five components provide a complete licensing ecosystem that operates entirely under the developer's control without requiring Internet activation servers, cloud-based licensing services, or third-party subscription providers.

CHAPTER 1 - INTRODUCTION

What Is LicenseTrak?

LicenseTrak is a software licensing framework designed specifically for WinDev desktop applications. It provides software developers with the tools necessary to create, distribute, manage, and enforce application licenses without requiring Internet activation servers, cloud-based licensing services, recurring subscription fees, or third-party licensing providers.

LicenseTrak consists of two primary elements. The first is the LicenseTrak Administration Program, which is used to create and manage applications, customers, licenses, and support records. The second is the LicenseTrak Runtime Library, a collection of WinDev procedures that are incorporated into a protected application to validate and enforce licensing policies.

LicenseTrak supports a variety of licensing models, including trial licenses, registered licenses, subscription licenses, and locked licenses. Developers may enforce trial periods based upon elapsed days, application executions, expiration dates, or combinations thereof. Applications may also be restricted to specific licensing conditions defined by the software developer.

Unlike many modern licensing systems, LicenseTrak operates entirely under the developer's control. No external activation servers, Internet connectivity, cloud infrastructure, or recurring service subscriptions are required. License files are generated by the developer and distributed directly to customers, allowing complete ownership of the licensing process.

LicenseTrak was designed for software developers who want a practical, affordable, and self-contained licensing

solution that can be integrated into WinDev applications with minimal effort while retaining complete control over customer licensing and application distribution.

Who Should Use LicenseTrak?

LicenseTrak was designed for software developers who create WinDev desktop applications and require a practical method of managing software licensing. Whether you distribute freeware, shareware, subscription-based software, or commercial business applications, LicenseTrak provides the tools necessary to control and manage application licensing.

Independent software developers can use LicenseTrak to distribute trial versions of their applications, convert trial users into paying customers, and manage registered licenses without relying upon costly third-party licensing services.

Consultants and contract developers may use LicenseTrak to create licensing solutions for customer-specific applications where control of software distribution and usage is important.

Small software companies can use LicenseTrak to manage customer records, application licenses, support history, and software distribution from a single integrated environment.

LicenseTrak is particularly well suited for developers creating vertical-market applications such as church management systems, medical office applications, legal office software, inventory systems, membership management applications, educational software, and other business-oriented desktop applications.

Because LicenseTrak operates entirely under the developer's control, it is especially attractive to organizations that prefer to avoid cloud-based licensing systems, Internet activation requirements, recurring subscription fees, or dependence upon third-party licensing providers.

LicenseTrak is intended for WinDev desktop application developers who want a simple, affordable, and self-contained licensing solution that can be integrated into their software with minimal effort while maintaining complete ownership of the licensing process.

What Problems Does LicenseTrak Solve?

Software developers face a variety of challenges when distributing commercial applications. While creating the software itself is often the primary focus, managing software licensing, trial versions, customer registrations, and license enforcement can quickly become a complex and time-consuming task.

One common challenge is distributing evaluation versions of an application while maintaining control over how long the software may be used. Developers often need the ability to limit application usage by elapsed days, execution counts, expiration dates, or other licensing criteria.

Another challenge involves converting evaluation users into paying customers. Developers require a reliable method of generating and distributing registered licenses while maintaining accurate customer and licensing records.

Many developers also struggle with maintaining a centralized repository of application information, customer information, license history, and support activity. As the number of customers grows, manually tracking this information can become increasingly difficult.

Modern licensing solutions frequently introduce additional complexity by requiring Internet activation servers, cloud-based licensing platforms, recurring subscription fees, online account management, or reliance upon third-party service providers. These approaches may increase costs, introduce new points of failure, and reduce the developer's direct control over the licensing process.

LicenseTrak addresses these challenges by providing a complete licensing ecosystem that combines license generation, customer management, support tracking, license validation, and runtime enforcement within a single solution. Developers retain complete ownership of the licensing process while avoiding the cost and complexity associated with many cloud-based licensing systems.

Whether the goal is distributing trial software, managing registered customers, enforcing subscription expiration dates, tracking support history, or maintaining control over software distribution, LicenseTrak provides a practical and self-contained solution designed specifically for WinDev desktop applications.

What LicenseTrak Can Do

LicenseTrak provides software developers with a complete licensing framework for WinDev desktop applications. The system combines license generation, customer management, support tracking, and runtime license enforcement into a single integrated solution.

LicenseTrak can generate and manage multiple license types, including Trial, Registered, Subscription, and Locked licenses. Developers may choose the licensing model that best fits their application's distribution and business requirements.

LicenseTrak supports several license enforcement methods. Trial licenses may be limited by elapsed days, execution counts, expiration dates, or combinations of these criteria. Runtime functions allow protected applications to validate and enforce licensing policies during application execution.

The LicenseTrak Administration Program maintains centralized records for applications, customers, licenses, support activity, and generated license history. This information provides developers with a convenient method of tracking customer relationships and software distribution activities.

LicenseTrak generates a License.LIC file that contains all information necessary to validate and enforce licensing policies within a protected application. The Runtime Library reads and validates this file during application execution and provides functions that allow developers to incorporate licensing controls into their software.

LicenseTrak includes mechanisms to verify license integrity and detect license file tampering. Applications may validate license information before permitting access to protected features or functionality.

Because LicenseTrak operates entirely under the developer's control, no Internet activation servers, cloud-based licensing services, recurring subscription fees, or third-party licensing providers are required. License files are generated and distributed directly by the software developer.

LicenseTrak can be used with a wide variety of software distribution models, including freeware, shareware, commercial desktop applications, annual maintenance agreements, subscription-based software, and vertical-market business applications.

In summary, LicenseTrak provides the tools necessary to create, distribute, validate, and manage software licenses while allowing developers to retain complete ownership and control of the licensing process.

What LicenseTrak Cannot Do

LicenseTrak was designed to provide a practical and self-contained licensing solution for WinDev desktop applications. While it offers a wide range of licensing capabilities, it is important to understand what LicenseTrak was not designed to accomplish.

LicenseTrak does not provide concurrent-user licensing management. The system cannot determine how many users are simultaneously running an application across a network, nor can it enforce seat-count or floating-license restrictions.

LicenseTrak does not require or provide Internet activation services. While this eliminates the cost and complexity associated with activation servers, it also means that LicenseTrak does not communicate with external systems to validate licenses, deactivate installations, or monitor application usage.

LicenseTrak is not a payment processing system. It does not process credit card transactions, subscriptions, invoices, refunds, or other financial activities. Developers remain responsible for managing software sales and payment collection using the tools and services of their choice.

LicenseTrak is not a customer relationship management (CRM) system. While customer records and support history may be maintained within the application, LicenseTrak is not intended to replace dedicated CRM, accounting, help desk, or sales management software.

LicenseTrak does not provide hardware-based licensing. Licenses are not tied to specific computer hardware, serial numbers, network adapters, processors, or storage devices. License files may be distributed and managed without requiring hardware fingerprinting or activation procedures.

LicenseTrak does not guarantee protection against reverse engineering, software cracking, or other determined attempts to bypass licensing controls. No software licensing system can provide absolute protection against a sufficiently skilled and motivated attacker. LicenseTrak is designed to discourage casual misuse and provide practical license management capabilities rather than serve as an impenetrable security solution.

LicenseTrak is not a cloud-based licensing platform and does not provide online license management portals, web-based customer self-service functions, or remote license administration capabilities.

Finally, LicenseTrak does not attempt to dictate how licensing policies should be enforced within an application. The Runtime Library provides the tools necessary to validate license conditions, but the developer remains responsible for determining how the protected application should respond when licensing restrictions are encountered.

Understanding these limitations will help developers determine whether LicenseTrak is an appropriate solution for their licensing requirements and will establish realistic expectations regarding its capabilities.

System Requirements

The LicenseTrak Administration Program is a Windows desktop application designed for software developers using the WinDev development environment. The following minimum requirements are recommended:

Operating System

- Microsoft Windows 10 or later
- Microsoft Windows Server environments capable of running WinDev desktop applications

Development Environment

- WinDev 2024 or newer (recommended; earlier WinDev versions may work if the WinDev function calls are identical within the older version)
- WinDev desktop application development experience

Database Requirements

- HFSQL Classic

Hardware Requirements

- 4 GB RAM minimum
- 8 GB RAM recommended
- 100 MB available disk space for the LicenseTrak application and database files
- Additional storage as required for customer records, license history, and application data

Runtime Integration Requirements

Applications protected by LicenseTrak must:

- Include the License.LIC file definition within the application analysis
- Import the LicenseTrak Runtime Library procedures
- Deploy a valid License.LIC file with the protected application

Network and Internet Requirements

LicenseTrak does not require:

- Internet connectivity
- Cloud-based services
- Activation servers
- Online licensing portals
- Third-party subscription services

All license generation, validation, and management activities are performed locally under the control of the software developer.

Recommended Environment

For best results, developers should maintain regular backups of the LicenseTrak Database and Developer Seed information. ***Loss of either may make it difficult or impossible to reproduce previously generated licenses***.

Because LicenseTrak uses a self-contained licensing model, protecting and backing up licensing information is considered an important administrative responsibility.

CHAPTER 2 - LICENSETRAK CONCEPTS

LicenseTrak supports several license types that allow software developers to implement licensing models appropriate for their applications and business requirements. Each license type serves a specific **purpose** and may be combined with various license enforcement methods to create flexible licensing strategies.

LicenseTrak currently supports the following license types:

1. **Trial License:** A Trial license permits a prospective customer to evaluate an application before purchase. Trial licenses may be restricted by elapsed days, application executions, expiration dates, or combinations thereof.
2. **Registered License:** A Registered license indicates that a customer has purchased the application and is authorized to use the software according to the licensing terms established by the developer. Registered licenses may be perpetual or subject to additional licensing restrictions.
3. **Locked License:** A Locked license prevents application usage. This license type is typically used when a license has expired, has been replaced, has been revoked, or should otherwise no longer permit application access.
4. **Subscription License:** A Subscription license permits application usage through a specified date. Once the Licensed Through Date has been reached, the application may restrict functionality, enter a trial mode, or deny access entirely depending upon how the developer has implemented license enforcement.

Each license type is represented within the License.LIC file and may be evaluated by the LicenseTrak Runtime Library during application execution. The developer determines how the protected application responds to each license type and licensing condition.

The following sections describe each license type in greater detail.

Trial Licenses

A Trial license allows prospective customers to evaluate an application before making a purchasing decision. Trial licensing has long been one of the most effective methods of introducing software to new users because it allows customers to experience the application's functionality within their own environment before committing to a purchase.

Within LicenseTrak, a Trial license is identified by a License Status value of "Trial". When a protected application detects a Trial license, the Runtime Library may be used to determine whether the trial remains valid and whether any licensing restrictions have been exceeded.

LicenseTrak supports several methods of controlling trial usage.

Trial Days

A Trial license may be configured to remain valid for a specified number of calendar days. The Runtime Library tracks the date the application is first executed and calculates the remaining trial period based upon the number of days authorized within the license.

For **example**, a developer may choose to distribute a 30-day evaluation version of an application. Once the authorized trial period has expired, the application may restrict functionality, deny access, or prompt the user to purchase a registered license.

Trial Runs

A Trial license may also be limited by the number of times the application is executed. Each application launch increments the execution count stored within the License.LIC file.

For **example**, a developer may choose to allow 50 executions of an application regardless of how many calendar days have elapsed. Once the maximum execution count has been reached, the application may respond according to the developer's licensing policy.

Hybrid Trials

LicenseTrak also supports combining multiple trial restrictions within a single license. A developer may choose to limit an application to both a specific number of days and a specific number of executions.

For **example**, a Trial license might permit application use for 30 days or 50 executions, whichever occurs first. This approach prevents users from circumventing trial limitations by either infrequent or excessive application usage.

Converting Trial Users To Registered Users

One of the primary purposes of a Trial license is to encourage software evaluation while providing a straightforward path to registration. When a customer purchases the software, the developer simply generates and distributes a new Registered License.LIC file that replaces the trial license.

Because the application, data files, and configuration settings remain unchanged, customers can typically continue using the software immediately after replacing the Trial License.LIC file with the Registered License.LIC file.

Trial licenses provide developers with a flexible and effective method of introducing software to potential customers while maintaining control over evaluation periods and application usage.

Registered Licenses

A Registered license indicates that a customer has purchased the software and is authorized to use the application according to the licensing terms established by the developer.

Within LicenseTrak, a Registered license is identified by a License Status value of "Registered". When a protected application detects a Registered license, the Runtime Library may be used to validate the license and provide access to registration-related information stored within the License.LIC file.

Registered licenses are typically issued after a customer completes the software purchasing process. The developer creates a customer record within the LicenseTrak Administration Program, generates a Registered License.LIC file, and delivers the file to the customer via email or other electronic means.

Unlike Trial licenses, Registered licenses are generally intended for long-term software usage. Depending upon the developer's licensing model, a Registered license may permit unlimited application use or may be combined with additional restrictions such as subscription expiration dates or application-specific limitations.

The License.LIC file may contain information identifying the registered customer, including the Registered To value. Developers may choose to display this information within application splash screens, About dialogs, report headers, status bars, or other user interface elements.

A Registered license does not require Internet activation, online validation, or communication with external licensing servers. License validation is performed locally by the Runtime Library using information contained within the License.LIC file.

For many software developers, the Registered license represents the most common licensing model. Once a customer purchases the software and receives a Registered License.LIC file, the application may continue operating indefinitely unless additional licensing restrictions have been implemented by the developer.

LicenseTrak allows developers to generate Registered licenses quickly and efficiently while maintaining complete ownership and control of the licensing process.

Locked Licenses

A Locked license indicates that application usage is no longer permitted. Within LicenseTrak, a Locked license is identified by a License Status value of "Locked". When a protected application detects a Locked license, the Runtime Library may be used to determine that the application should restrict access according to the developer's licensing policies.

Locked licenses are typically used when software access should be denied regardless of other license settings. For **example**, a developer may choose to issue a Locked license when a subscription has expired, when a replacement license has been issued, when a license has been revoked, or when a trial period has been intentionally terminated.

The specific response to a Locked license is determined entirely by the developer. Some applications may display an informational message and terminate immediately, while others may allow limited access to certain functions such as viewing existing data, exporting information, or contacting the software publisher.

A Locked license provides a simple and consistent method of preventing normal application operation while allowing the developer to determine the most appropriate user experience for the situation.

Because the Runtime Library only validates licensing conditions, LicenseTrak does not dictate how a protected application must respond when a Locked license is encountered. Developers retain complete control over the actions taken by their applications.

Typical responses to a Locked license include:

- Displaying a license expiration message.
- Prompting the user to contact the software publisher.

- Restricting access to application features.
- Preventing new data entry.
- Allowing read-only access to existing information.
- Terminating the application.

Locked licenses provide developers with a flexible mechanism for controlling application access while maintaining a consistent licensing model across all protected applications.

Subscription Licenses

A Subscription license permits application usage through a specified date defined by the software developer. Unlike traditional Registered licenses that may allow indefinite application use, Subscription licenses are intended to support licensing models that require periodic renewal.

Within LicenseTrak, Subscription licensing is typically implemented through the Licensed Through Date stored within the License.LIC file. The Runtime Library may be used to determine whether the current date remains within the authorized subscription period.

Subscription licensing can be used to support a variety of business models, including:

- Annual software subscriptions.
- Monthly software subscriptions.
- Software maintenance agreements.
- Membership-based applications.
- Educational licensing.
- Time-limited commercial software usage.

For **example**, a developer may issue a license that permits application use through December 31, 2027. Prior to that date, the application operates normally. Once the Licensed Through Date has been exceeded, the application may respond according to the developer's licensing policies.

The response to an expired subscription is entirely under the developer's control. Common approaches include:

- Displaying renewal reminders.
- Restricting selected application features.
- Preventing new data entry.
- Allowing read-only access to existing information.
- Transitioning the license to a Locked state.
- Terminating application access.

Subscription licensing does not require Internet connectivity or communication with external licensing servers. All validation is performed locally using information stored within the License.LIC file.

Many developers use Subscription licenses in conjunction with Registered licenses. In this model, a customer is recognized as a registered user while the Licensed Through Date determines eligibility for continued software use, maintenance, upgrades, or support services.

Because subscription validation is performed locally, developers retain complete control over how renewal dates are

managed and how expired subscriptions are handled within their applications.

Subscription licenses provide a flexible mechanism for implementing renewable licensing models while preserving the simplicity and self-contained nature of the LicenseTrak licensing framework.

LICENSE ENFORCEMENT METHODS

Trial Days

Trial Days provide a method of limiting application usage based upon the number of calendar days that have elapsed since the software was first executed.

This licensing method is commonly used when developers wish to provide prospective customers with a fixed evaluation period. Examples include 7-day, 14-day, 30-day, or 60-day software trials.

When a Trial license containing a Trial Days value is first executed, the LicenseTrak Runtime Library records the initial execution date within the FirstRunDate field of the License.LIC file. Each subsequent application execution compares the current date against the FirstRunDate and the authorized Trial Days value to determine whether the trial period remains valid.

For **example**, a Trial license configured with a Trial Days value of 30 permits application usage for thirty calendar days from the date of first execution. Once the authorized period has expired, the Runtime Library can indicate that the trial is no longer valid.

The specific actions taken after trial expiration are determined entirely by the software developer. Common responses include:

- Displaying a trial expiration message.
- Prompting the user to purchase a Registered license.
- Restricting selected application functions.
- Preventing data modifications.
- Allowing read-only access to existing information.
- Terminating application execution.

Trial Days licensing is particularly effective for applications that are expected to be evaluated over a period of time. Because the trial period is based upon elapsed calendar days rather than application usage frequency, customers receive a consistent evaluation experience regardless of how often the software is executed.

Developers may choose to use Trial Days as a standalone licensing method or combine it with other LicenseTrak enforcement mechanisms such as Trial Runs or Licensed Through Dates to create more sophisticated licensing strategies.

Trial Days remain one of the most widely used and easily understood software evaluation methods due to their simplicity, predictability, and ease of implementation.

Trial Runs

Trial Runs provide a method of limiting application usage based upon the number of times an application is executed rather than the number of calendar days that have elapsed.

This licensing method is particularly useful for applications that may be used infrequently, intermittently, or only when a specific need arises. In such situations, a traditional Trial Days model may not provide a meaningful evaluation period because the customer may only launch the application a few times during the trial period.

When a Trial license containing a Trial Runs value is executed, the LicenseTrak Runtime Library increments the ExecutionCount field stored within the License.LIC file. Each subsequent application execution compares the current ExecutionCount against the authorized Trial Runs value to determine whether the trial remains valid.

For **example**, a Trial license configured with a Trial Runs value of 50 permits fifty application executions. Regardless of whether those fifty executions occur within a single week or over several months, the customer receives the same number of opportunities to evaluate the software.

Once the authorized execution count has been reached, the Runtime Library can indicate that the trial is no longer valid. The protected application may then respond according to the developer's licensing policies.

Common responses include:

- Displaying a trial expiration message.
- Prompting the user to purchase a Registered license.
- Restricting selected application functions.
- Allowing read-only access to existing information.
- Preventing new data entry.
- Terminating application execution.

Trial Runs licensing is often well suited for utility applications, maintenance tools, conversion utilities, reporting applications, administrative software, and other programs that may not be executed on a daily basis.

Developers may choose to use Trial Runs as a standalone licensing method or combine it with Trial Days licensing to create a hybrid evaluation model. In a hybrid configuration, a trial may expire when either the maximum number of executions or the maximum number of days has been reached.

Because Trial Runs licensing is based upon actual application usage rather than elapsed time, it provides developers with a flexible alternative to traditional calendar-based evaluation periods.

Trial Runs offer an effective method of balancing customer evaluation opportunities with the developer's need to control software distribution and usage.

Licensed Through Date

Licensed Through Date provides a method of controlling application usage based upon a specific expiration date contained within the License.LIC file.

Unlike Trial Days, which calculate validity based upon the number of days that have elapsed since the application was

first executed, Licensed Through Date compares the current date against a fixed calendar date specified by the software developer.

For **example**, a developer may issue a license containing a Licensed Through Date of December 31, 2027. The application remains authorized for use until that date is reached. Once the Licensed Through Date has been exceeded, the Runtime Library can indicate that the license is no longer valid.

This licensing method is particularly well suited for:

- Annual software subscriptions.
- Monthly software subscriptions.
- Software maintenance agreements.
- Educational licensing programs.
- Membership-based applications.
- Time-limited commercial deployments.
- Support and upgrade entitlement programs.

Licensed Through Date licensing offers several advantages over traditional trial-based licensing models. Because the expiration date is explicitly defined within the license, developers can easily issue licenses that remain valid through a specific day, month, or year without requiring complex calculations based upon application installation or first execution dates.

Many developers use Licensed Through Date in conjunction with Registered licenses. In this model, the customer is recognized as a registered user while the Licensed Through Date determines whether the customer remains entitled to continue using the software, receive updates, obtain technical support, or access subscription-based features.

When a Licensed Through Date is exceeded, the developer determines how the application should respond. Common approaches include:

- Displaying renewal reminders.
- Restricting selected application functions.
- Preventing new data entry.
- Allowing read-only access to existing information.
- Restricting software updates.
- Transitioning the license to a Locked state.
- Terminating application execution.

Because all validation is performed locally using information stored within the License.LIC file, Licensed Through Date enforcement does not require Internet connectivity, activation servers, cloud-based licensing services, or communication with external systems.

Licensed Through Date is one of the most flexible licensing mechanisms available within LicenseTrak because it can be applied to a wide variety of business models while remaining simple for both developers and customers to understand.

For many commercial applications, the Licensed Through Date serves as the primary mechanism for controlling renewable software licenses, maintenance agreements, and subscription-based access.

Application Restriction

Application Restriction provides a mechanism for ensuring that a License.LIC file is only valid for the application for which it was originally generated.

In many software environments, a developer may distribute multiple applications to customers. Without application-specific validation, there would be a risk that a license intended for one application could be copied and used with another application.

LicenseTrak addresses this concern by storing application-specific information within the License.LIC file. During application initialization, the Runtime Library compares information contained within the license against information supplied by the protected application.

If the license was generated for a different application, the Runtime Library can indicate that the license should not be considered valid for the current application.

For **example**, a software developer may distribute the following applications:

- ChurchTrak
- MemberTrak
- FinanceTrak
- InventoryTrak

A license generated for ChurchTrak should not automatically permit usage of FinanceTrak or InventoryTrak. Application Restriction allows each application to verify that the License.LIC file was created specifically for that application.

Application Restriction is particularly valuable for developers who maintain multiple commercial products because it helps ensure that licenses remain associated with their intended applications.

The specific response to an application restriction violation is determined by the developer. Common responses include:

- Displaying a license validation message.
- Prompting the user to obtain the correct license.
- Transitioning the application to a Locked state.
- Restricting application access.
- Terminating application execution.

Application Restriction operates entirely within the LicenseTrak Runtime Library and does not require Internet connectivity, activation servers, or communication with external systems.

By validating that a license belongs to the correct application, Application Restriction provides an additional layer of licensing control while maintaining the simplicity and self-contained nature of the LicenseTrak framework.

License Hash Validation

License Hash Validation provides a mechanism for detecting unauthorized modifications to the contents of a License.LIC file.

Software licenses often contain important information such as license status, trial limits, expiration dates, customer information, and other licensing

. If these values could be freely modified without detection, the effectiveness of the licensing system would be significantly reduced.

LicenseTrak addresses this concern through the use of a validation value stored within the License.LIC file. This validation value is generated when the license is created and is later examined by the Runtime Library during application execution.

When the Runtime Library validates a license, it verifies that the information contained within the license remains consistent with the original values that were present when the license was generated. If the validation process determines that the license information has been altered, the Runtime Library can indicate that the license should no longer be considered valid.

For **example**, a user might attempt to modify:

- Trial Days
- Trial Runs
- Licensed Through Date
- License Status
- Application information

If such modifications are detected, the application may respond according to the developer's licensing policies.

Common responses include:

- Displaying a license validation error.
- Restricting application functionality.
- Transitioning the license to a Locked state.
- Preventing application execution.
- Requesting a replacement license from the software publisher.

License Hash Validation is intended to discourage casual modification of license information and provide developers with confidence that license data has not been altered without authorization.

Like all LicenseTrak licensing functions, validation is performed locally and does not require Internet connectivity, activation servers, cloud services, or communication with external systems.

While no desktop licensing system can provide absolute protection against a sufficiently skilled and determined attacker, License Hash Validation provides an effective mechanism for detecting unauthorized modifications and maintaining the integrity of license information.

License Hash Validation serves as an important component of the overall LicenseTrak licensing framework by helping ensure that license information remains consistent, trustworthy, and under the control of the software developer.

The Developer Seed

The Developer Seed is one of the most important elements within the LicenseTrak licensing framework. It serves as a developer-defined value that helps establish the uniqueness and integrity of licenses generated for a protected application.

When a new application is created within the LicenseTrak Administration Program, the developer assigns a Developer Seed value to that application. This value becomes part of the licensing information used during license generation and validation.

The primary **purpose** of the Developer Seed is to help ensure that licenses generated for one developer or application cannot be freely reused within another application or licensing environment. By incorporating a developer-specific value into the licensing process, LicenseTrak provides an additional layer of uniqueness and separation between applications.

The Developer Seed plays an important role in several areas of the LicenseTrak framework, including:

- License generation.
- License validation.
- Application restriction.
- License integrity verification.
- License hash validation.

Although the Runtime Library makes use of the Developer Seed during license validation, end users are never required to enter, modify, or interact with the Developer Seed directly. The value exists solely as part of the licensing infrastructure maintained by the software developer.

Because the Developer Seed contributes to license validation, it should be considered confidential information. Developers should avoid publishing Developer Seed values within documentation, source code examples, websites, online forums, or other publicly accessible locations.

It is strongly recommended that developers maintain secure backups of all Developer Seed values associated with their applications. Loss of a Developer Seed may make it difficult or impossible to reproduce previously generated licenses or maintain compatibility with existing customer licenses.

The Developer Seed should be viewed as a permanent application attribute. Once licenses have been generated and distributed to customers, changing the Developer Seed may invalidate existing licenses and require new licenses to be generated.

For this reason, developers are encouraged to select Developer Seed values carefully and preserve them for the lifetime of the application.

The Developer Seed serves as a foundational component of the LicenseTrak licensing framework and helps ensure that licensing information remains unique, consistent, and under the control of the software developer.

CHAPTER 3 - QUICK START TUTORIAL

This chapter provides a hands-on introduction to the LicenseTrak licensing framework. By following the steps presented in this tutorial, you will create your first application record, generate a trial license, and deploy a functioning License.LIC file.

The objective of this chapter is not to explore every feature of LicenseTrak. Rather, it is intended to provide a practical introduction that demonstrates the basic workflow used to create and distribute software licenses.

By the end of this chapter, you will have:

- Created an Application record.
- Defined a Developer Seed.
- Created a Trial customer record.
- Generated a Trial License.LIC file.
- Deployed the License.LIC file.
- Successfully tested a protected application.

Additional features and licensing strategies are discussed in later chapters.

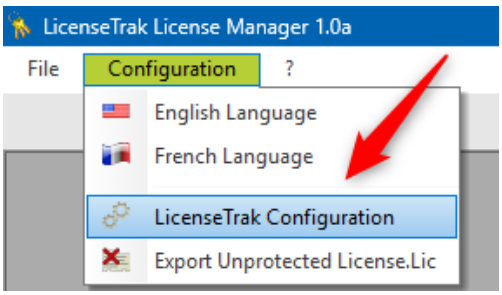
Before You Begin

Before generating licenses, LicenseTrak must be configured with a Developer Seed value. The Developer Seed is stored within the LicenseTrak.ini file and is used during license generation and validation.

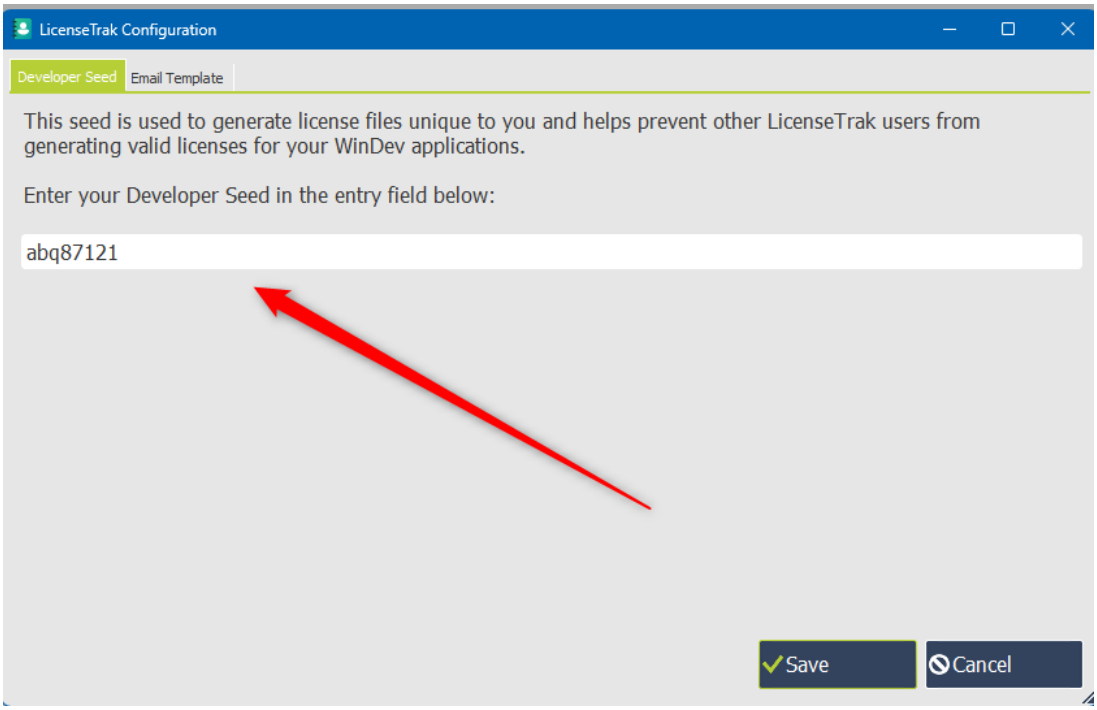
The Developer Seed should be considered confidential information and should be backed up regularly. Once licenses have been generated and distributed, changing the Developer Seed may invalidate previously generated licenses.

For this reason, developers should select a Developer Seed carefully and preserve it for the lifetime of the application portfolio.

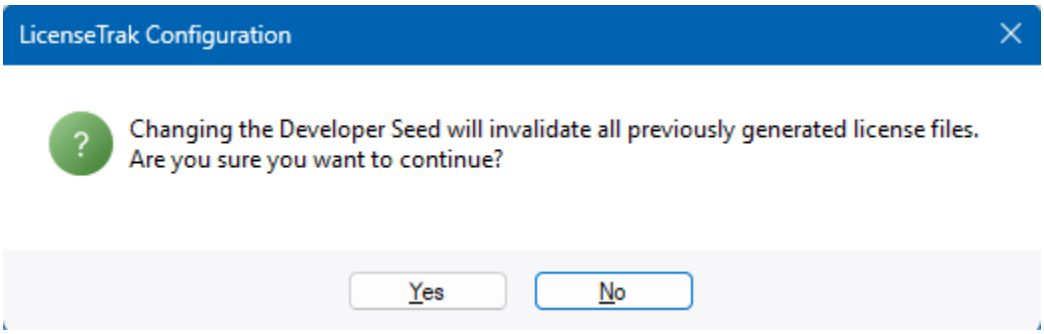
To define the Developer Seed, within the LicenseTrak main menu, select CONFIGURATION, then PROGRAMMER SEED:



A popup window is displayed to enter your Developer Seed:



When the SAVE button is clicked, a warning message is displayed should a Developer Seed already exist:



the value is written to the LicenseTrak.Ini file (example below):

[Main]
Seed=abq87121

Ensure you do not lose your Developer Seed – write it down elsewhere / send an email to yourself with its value.

Create an Application Record

Before licenses can be generated, LicenseTrak must know which application is being protected. This information is stored within an Application record. Every Trial, Registered, Subscription, or Locked license generated by LicenseTrak is associated with an Application record.

For this Quick Start tutorial, we will create a fictional application named **FictionTrak**.

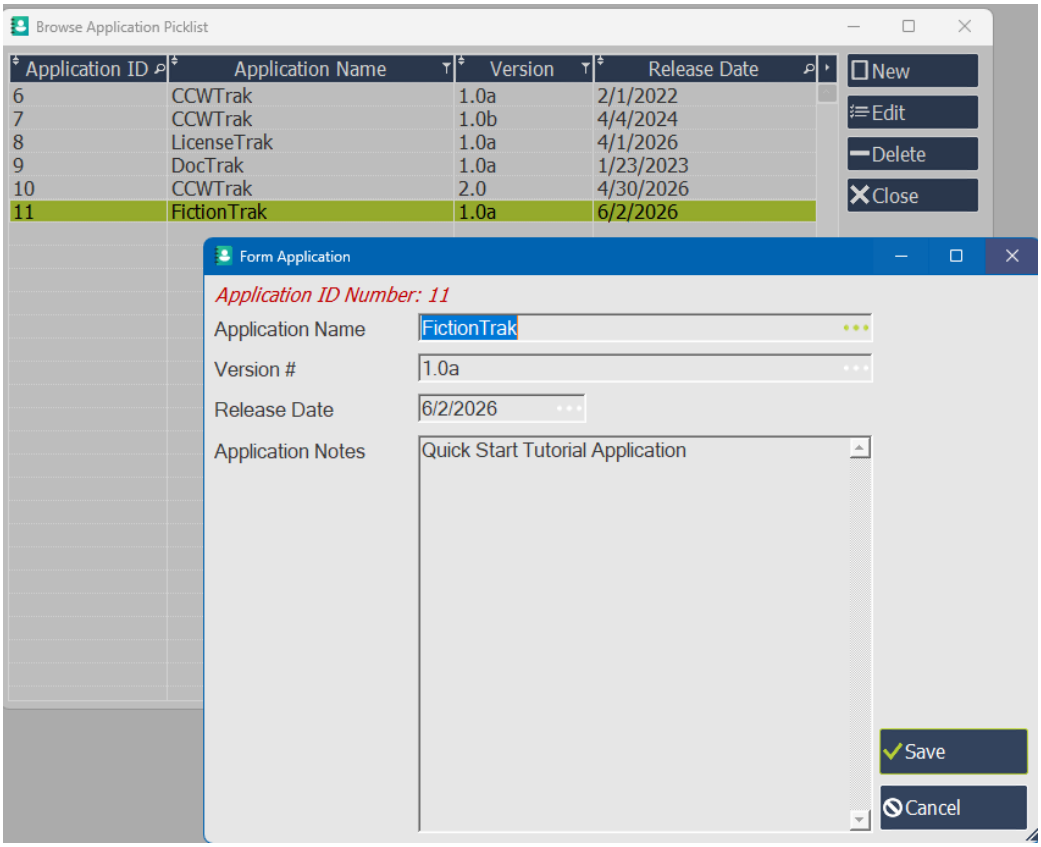
1. Creating the Application Record

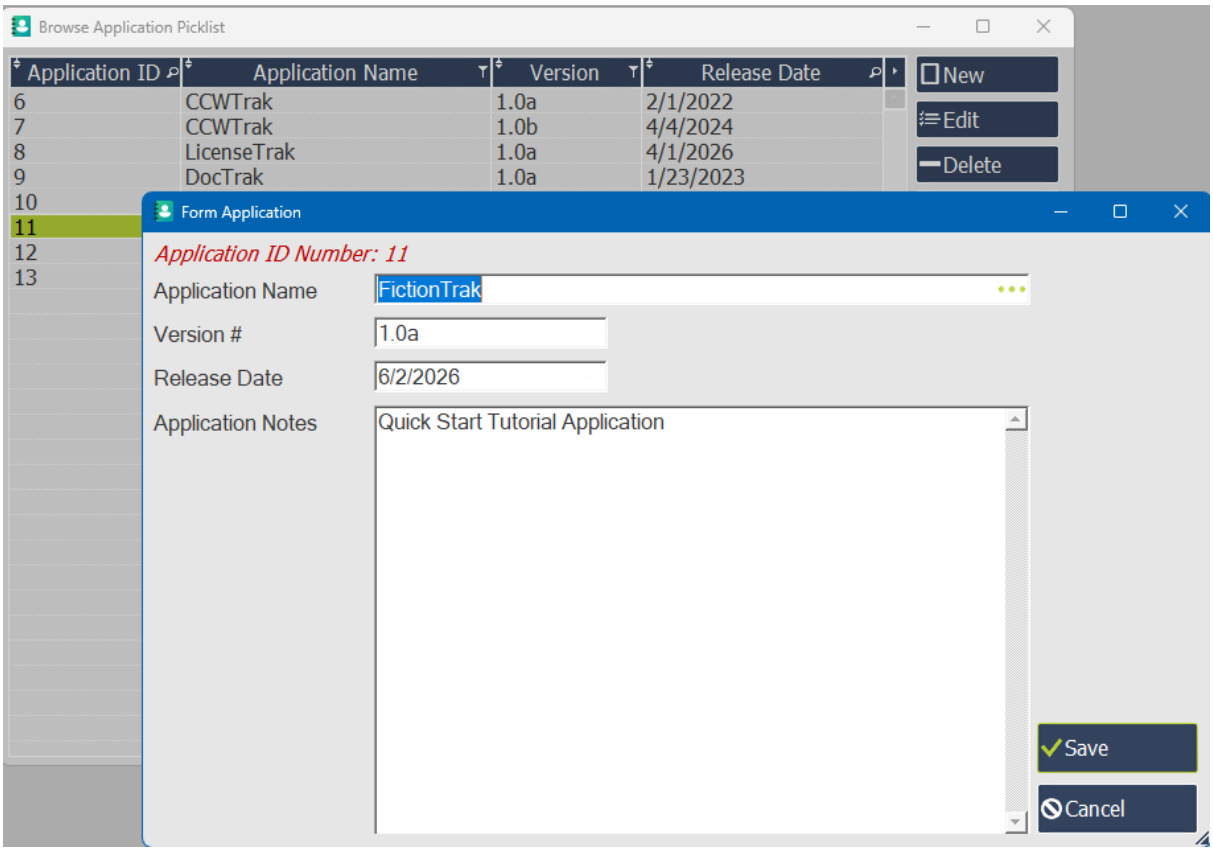
To create a new Application record:

a) Start the LicenseTrak Administration Program.
b) Select **Application Library** from the main menu.
c) Click the **Add** button.
d) Enter the following information:

Field	Value
Application Name	FictionTrak
Version #	1.0a
Release Date	Current Date
Application Seed	100
Notes	Quick Start Tutorial Application

Your screen will resemble this:





Save the Application record.

The Application record is now available for use throughout the LicenseTrak system.

Why the Application Record Is Important

The Application record serves as the foundation for all future licensing activities. LicenseTrak uses this information when generating license files and validating licenses within protected applications.

A typical Application record contains:

- Application Name
- Version
- Release Date
- Developer Seed
- Developer Notes

Additional application information may be maintained as required by the software developer.

About the Developer Seed

The Developer Seed is one of the most important values associated with an Application record. It contributes to license generation and validation and helps ensure that licenses remain associated with the intended application.

For this tutorial we are using: **`abq87121 (your developer seed would be used in your apps)`**

In a production environment, developers should select a unique value and maintain secure backups of all Developer Seeds used within their applications.

Important: Once licenses have been distributed to customers, changing the Developer Seed may invalidate previously generated licenses. **For this reason, Developer Seeds should be treated as permanent application attributes**.

After saving the record, the Application Library should display the newly created FictionTrak application.

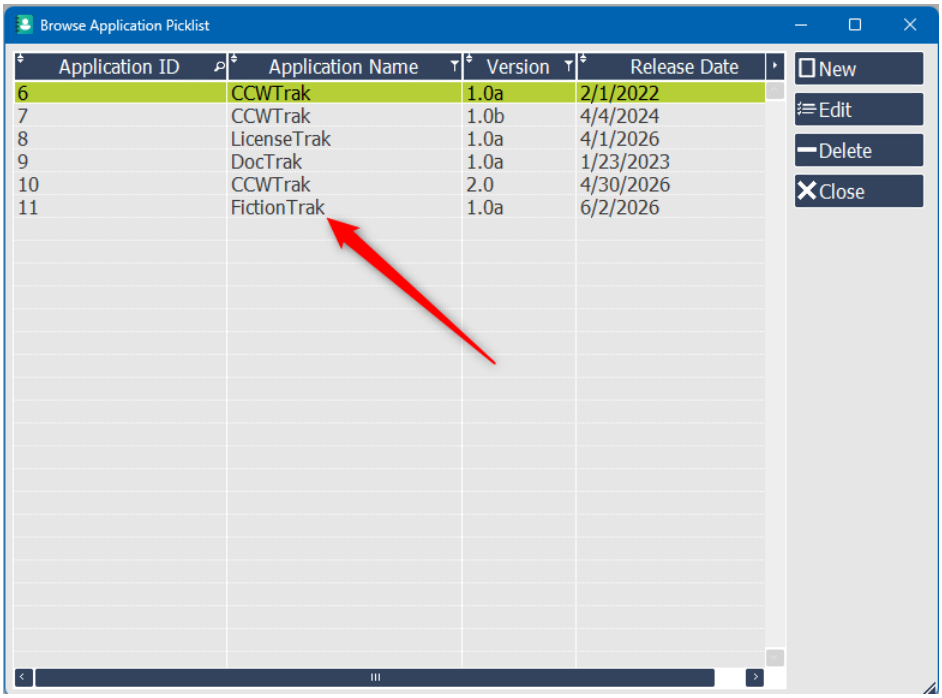
Before proceeding, verify that:

✓ The Application Name appears correctly.

✓ The Version number is correct.

✓ The Developer Seed has been entered correctly.

✓ The Application record has been successfully saved.



Once the Application record has been verified, you are ready to create your first customer record and generate a Trial License.LIC file.

Quick Start Scenario

For this Quick Start tutorial, we will follow a typical shareware software distribution model.

A software developer has completed an application named FictionTrak 1.0a and wishes to distribute a demonstration version of the application from the company web site.

To accomplish this, the developer must first create a Trial License.LIC file that will be included within the application's Setup.exe installation package.

The Trial License.LIC file allows prospective customers to evaluate the application before making a purchasing decision. For this tutorial, we will create a Trial license that permits application usage for 45 days.

The following steps will be performed:

1. Create an Application record for FictionTrak 1.0a.
2. Create a generic Trial User customer record.
3. Generate a 45-day Trial License.LIC file.
4. Deploy the Trial License.LIC file with the application installation package.
5. Test the protected application.

Once the application has been distributed, prospective customers may download and evaluate the software using the Trial License.LIC file.

If a customer later purchases the application, the developer creates a Customer record for that individual and generates a Registered License.LIC file that replaces the original Trial License.LIC file.

This tutorial focuses on creating and deploying the Trial License.LIC file that will be distributed with the application.

Create a Trial User

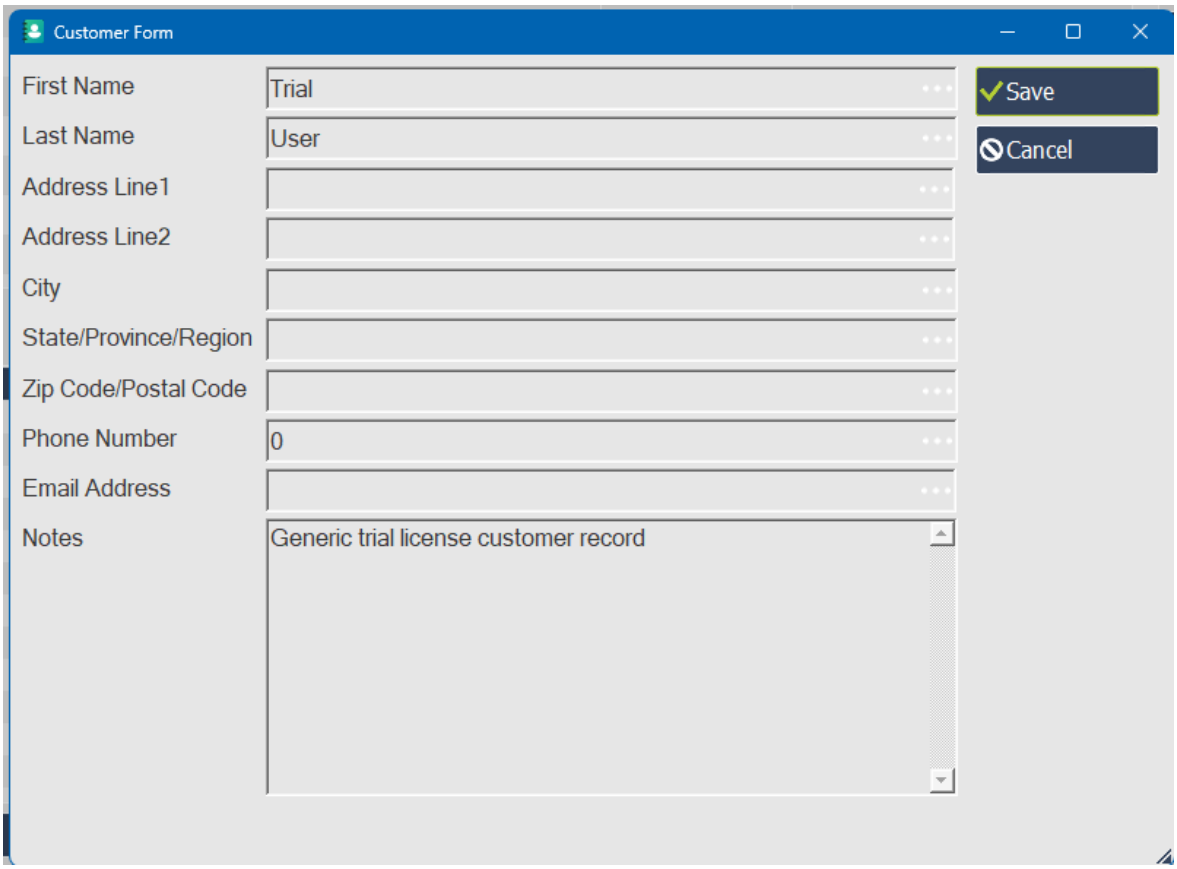
Before a Trial License.LIC file can be generated, a Customer record must exist within LicenseTrak.

For this Quick Start tutorial, we will create a generic customer record named Trial User. This customer record will be used to generate the Trial License.LIC file that will be distributed with the FictionTrak Setup.exe installation package (e.g. a demo version placed upon your web site for evaluation as a shareware application).

1. Select File, then Browse Customers.
2. Click the New button.
3. Enter the following information:

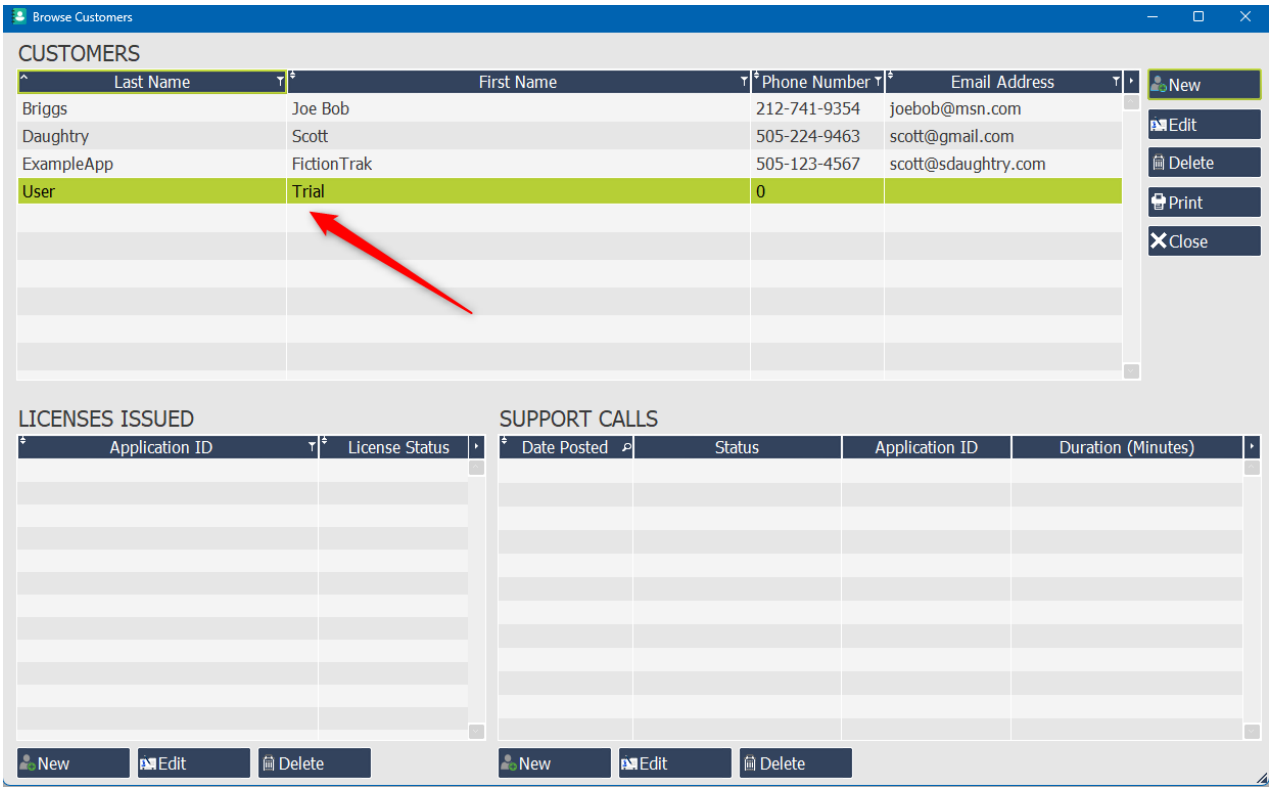
Field	Value
First Name	Trial
Last Name	User
Address Line 1	
Address Line 2	
City	
State/Province/Region	
Zip Code/Postal Code	
Phone Number	
Email Address	
Notes	Generic trial license customer record

Your screen should look like this:



4. Click the Save button.

The Browse Customers screen has been updated:



The Trial User customer record is now available for license generation activities.

In a typical shareware distribution model, a single Trial User customer record may be used to generate the evaluation License.LIC file that is included within the application's Setup.exe installation package.

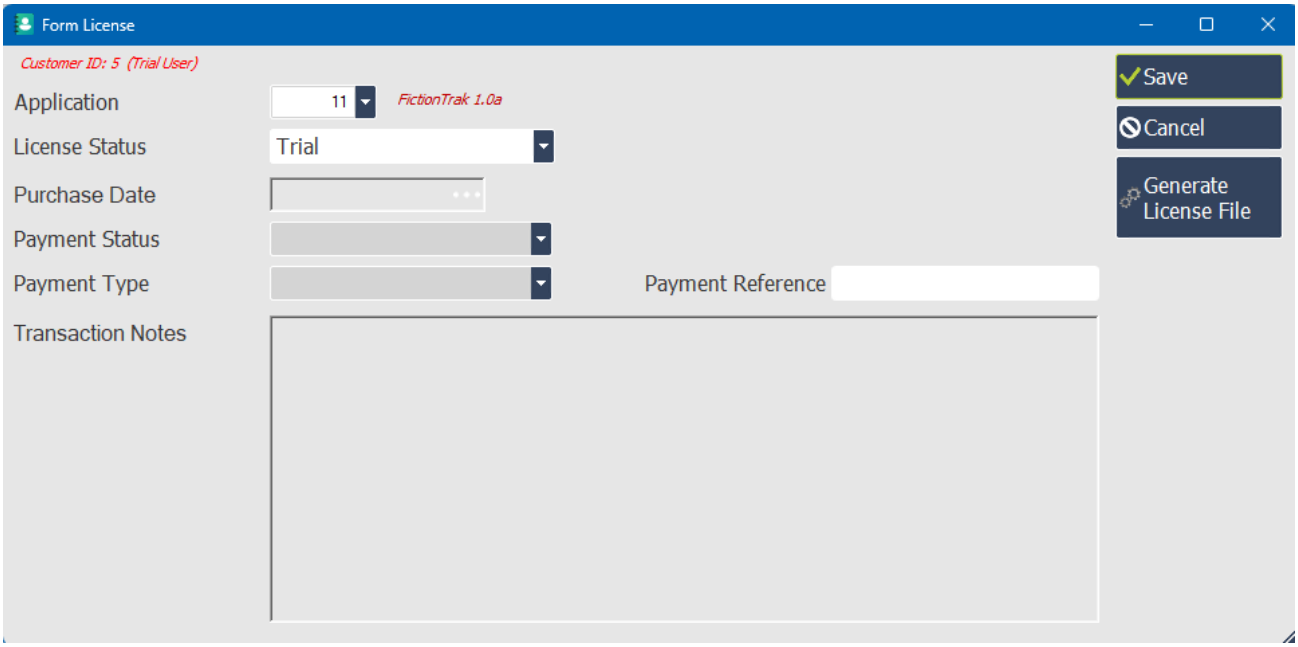
Once the Trial User customer record has been created, you are ready to generate a Trial License.LIC file for distribution with the FictionTrak application.

Generate a Trial License

We are now ready to generate the Trial License.LIC file that will be included with the FictionTrak Setup.exe installation package.

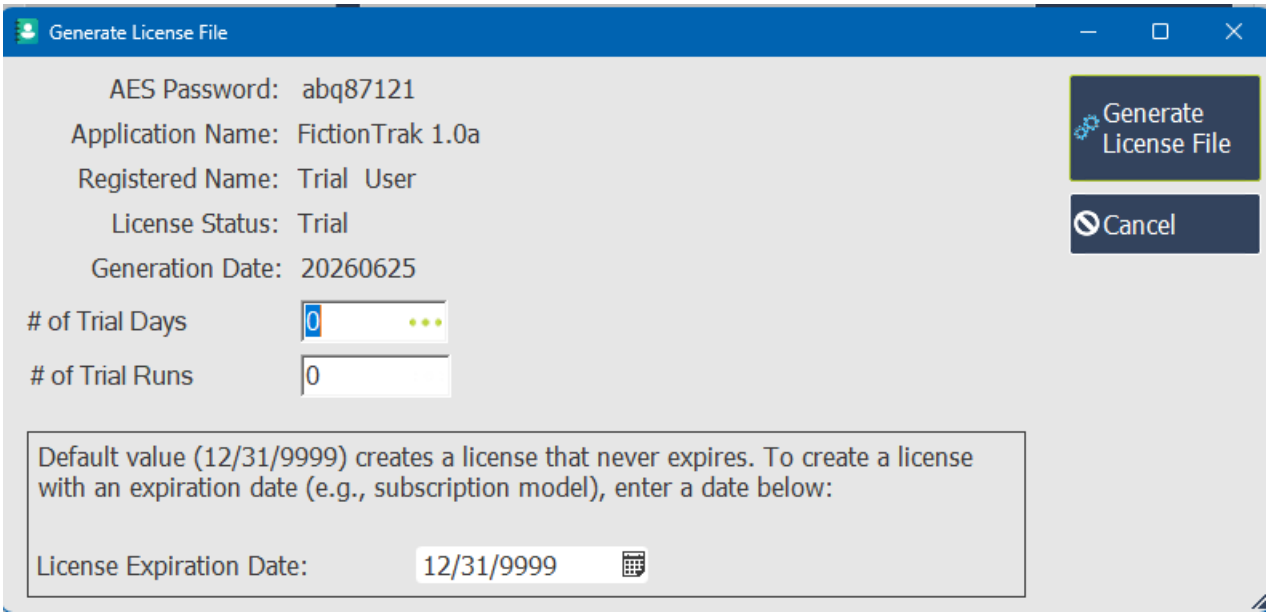
For this Quick Start tutorial, we will create a Trial license that permits application usage for 45 days. Select the newly created Trial User from the CUSTOMERS section of the Browse Customers window; within the LICENSES ISSUED section of the window (bottom left), click the NEW button to display the Form License.

1. Verify that FictionTrak 1.0a is selected within the Application dropdown list.
2. Set License Status to Trial.
3. Leave the payment-related fields blank. Because this is a demonstration license intended for software evaluation, no payment information is required. Your screen will resemble:



4. Click the Generate License File button.

The Generate License File window is displayed.



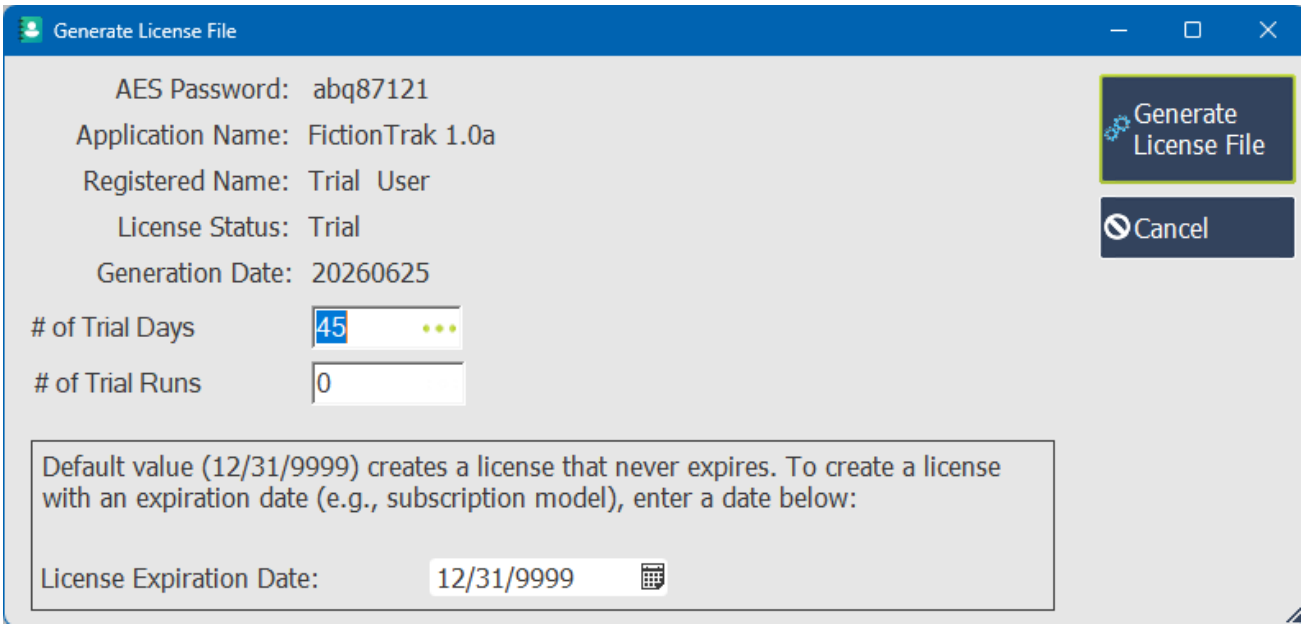
The Generate License File window summarizes the customer, application, and license information that will be used to create the License.LIC file.

For this tutorial, enter the following values:

Field	Value
# of Trial Days	45
# of Trial Runs	0
License Expiration Date	12/31/9999

A Trial Runs value of zero indicates that application usage is controlled exclusively by the Trial Days value.

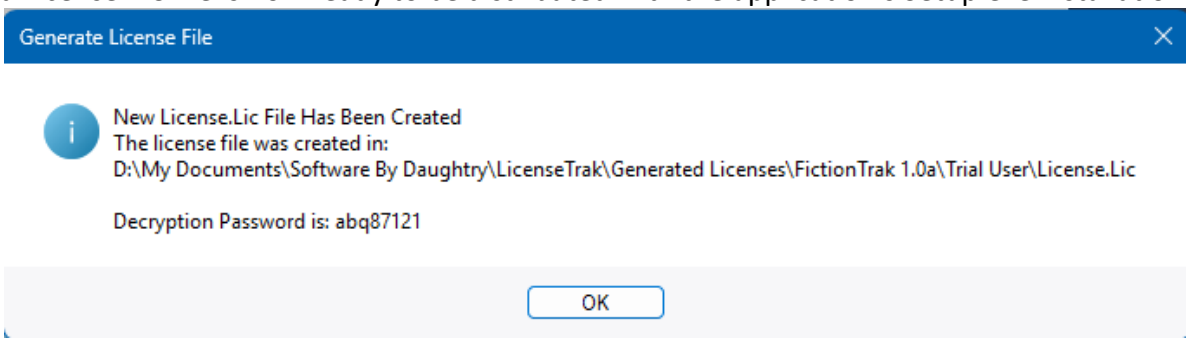
A License Expiration Date of 12/31/9999 indicates that the license itself does not expire. In this **example**, the application's evaluation period is controlled solely by the 45-day trial limit. Your screen will resemble:



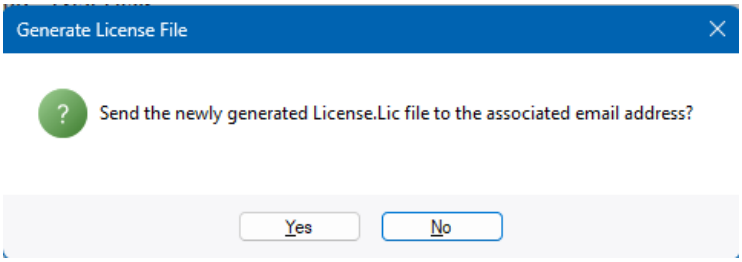
After verifying the displayed information, click the Generate License File button.

LicenseTrak creates a License.LIC file containing the licensing information required to evaluate the FictionTrak application.

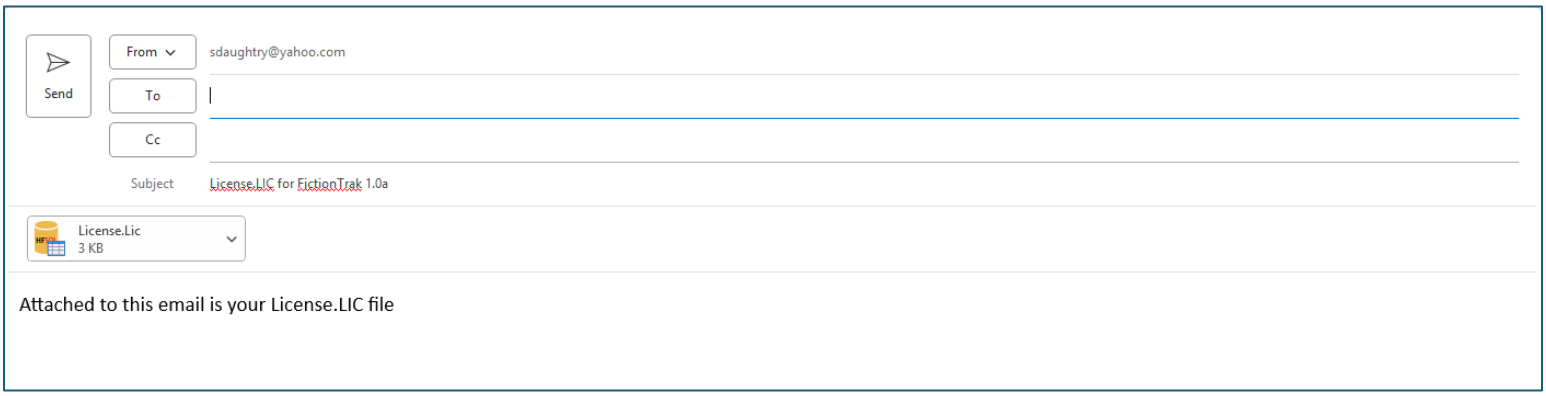
The generated License.LIC file is now ready to be distributed with the application's Setup.exe installation package:



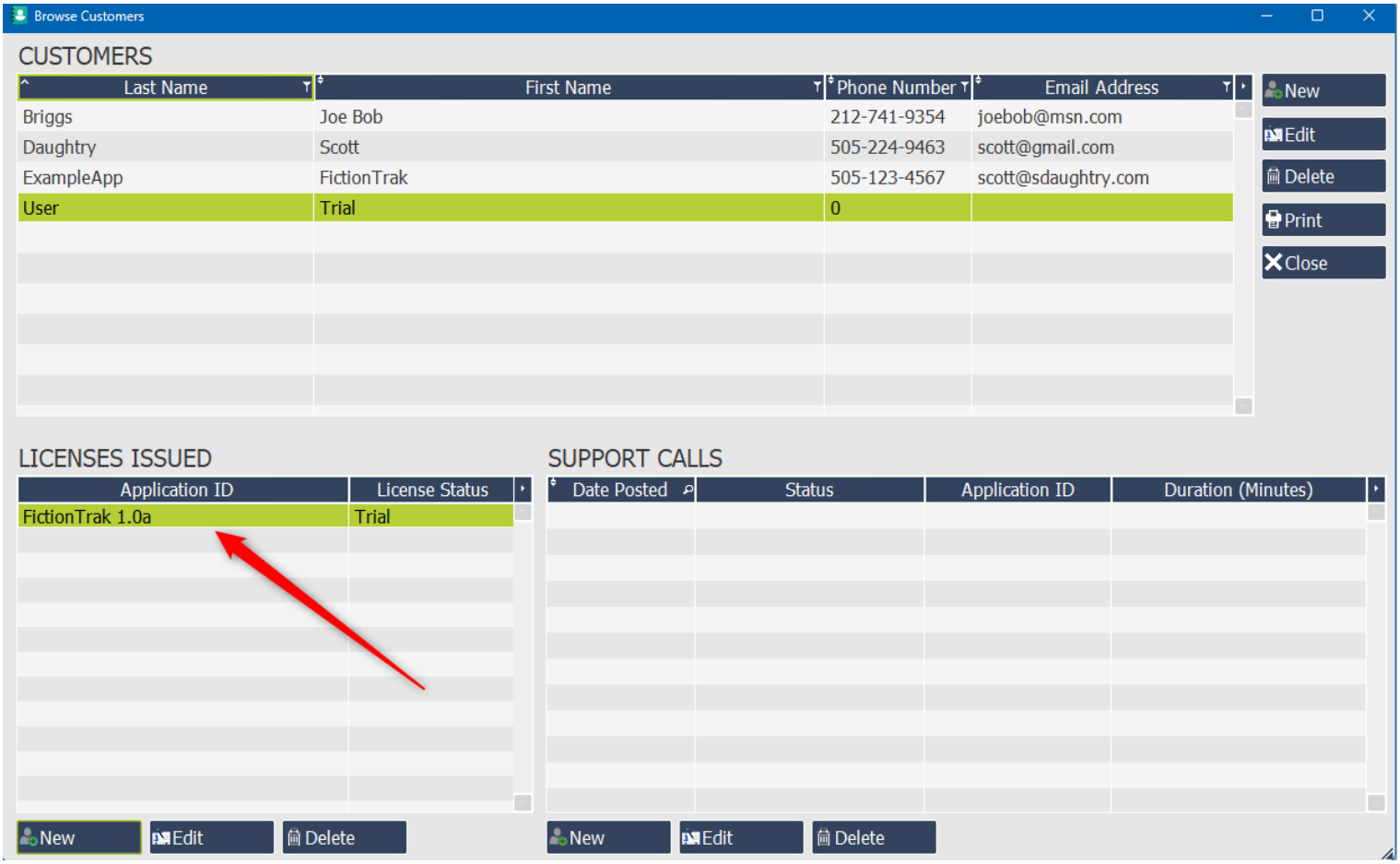
A Yes/No popup dialogue is then displayed to optionally create a new email to send the License.LIC file to the email account that is on file for this customer:



Answering YES will generate a new email similar to this screen capture:



Click SAVE to update the license database and close the Form License window. The Browse Customers window now displays a License Issuance for Trial User for FictionTrak:



Create a Trial Customer

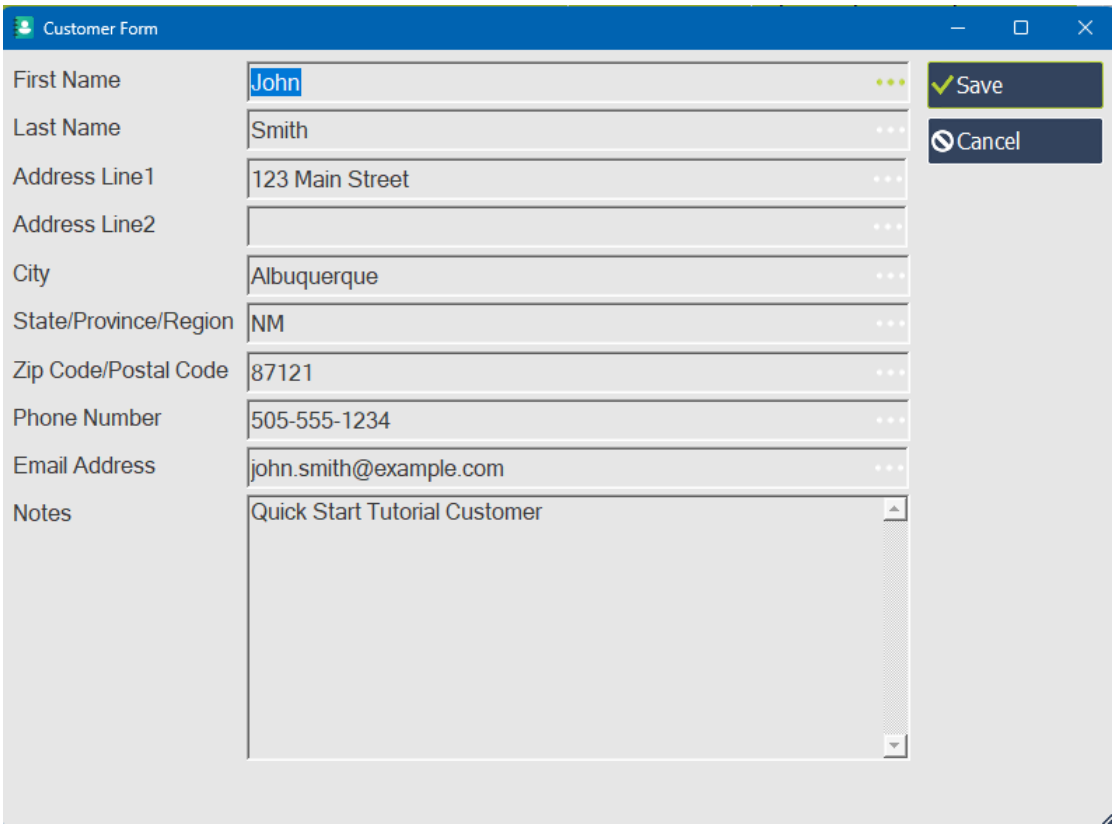
Before a license can be generated, a customer record must exist within LicenseTrak. Customer records contain the information associated with the individual who will receive the license.

For this Quick Start tutorial, we will create a fictional customer named John Smith.

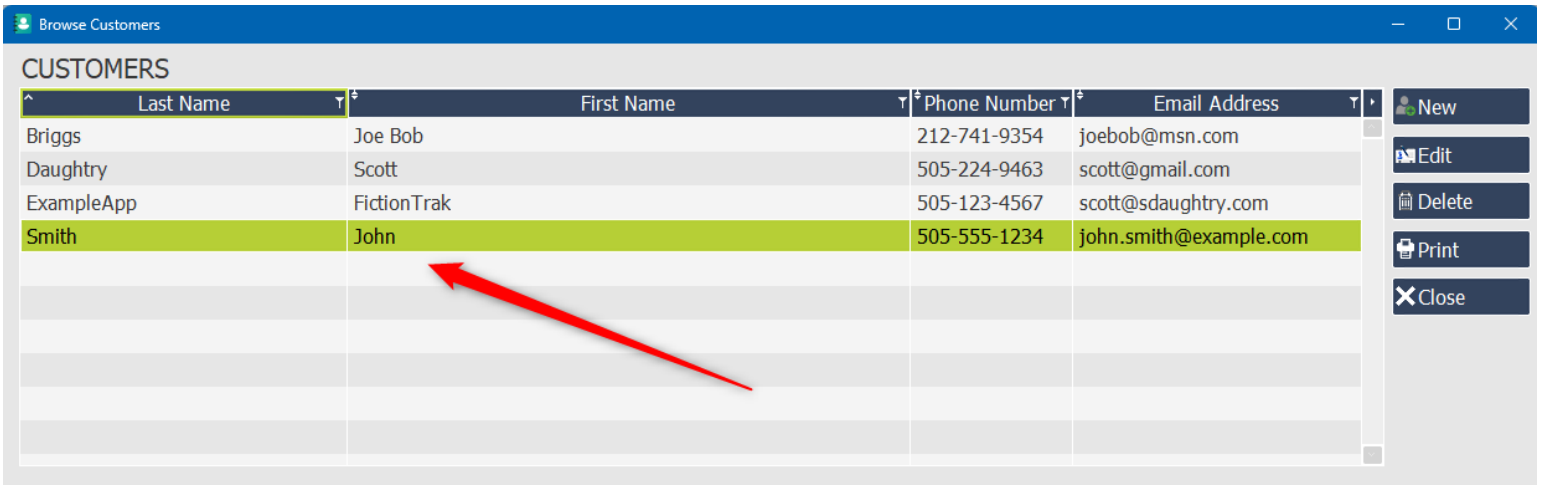
1. Select Customer Library from the main menu.
2. Click the New button.
3. Enter the following information:

Field	Value
First Name	John
Last Name	Smith
Address Line 1	123 Main Street
Address Line 2	
City	Albuquerque
State/Province/Region	NM
Zip Code/Postal Code	87121
Phone Number	505-555-1234
Email Address	john.smith@example.com

Notes	Quick Start Tutorial Customer



4. Click the SAVE button. The customer record is now available for license generation activities.



Customer records may be modified at any time to update contact information, address information, notes, or other customer-related details.

Once the customer record has been created, you are ready to generate your first trial License.LIC file.

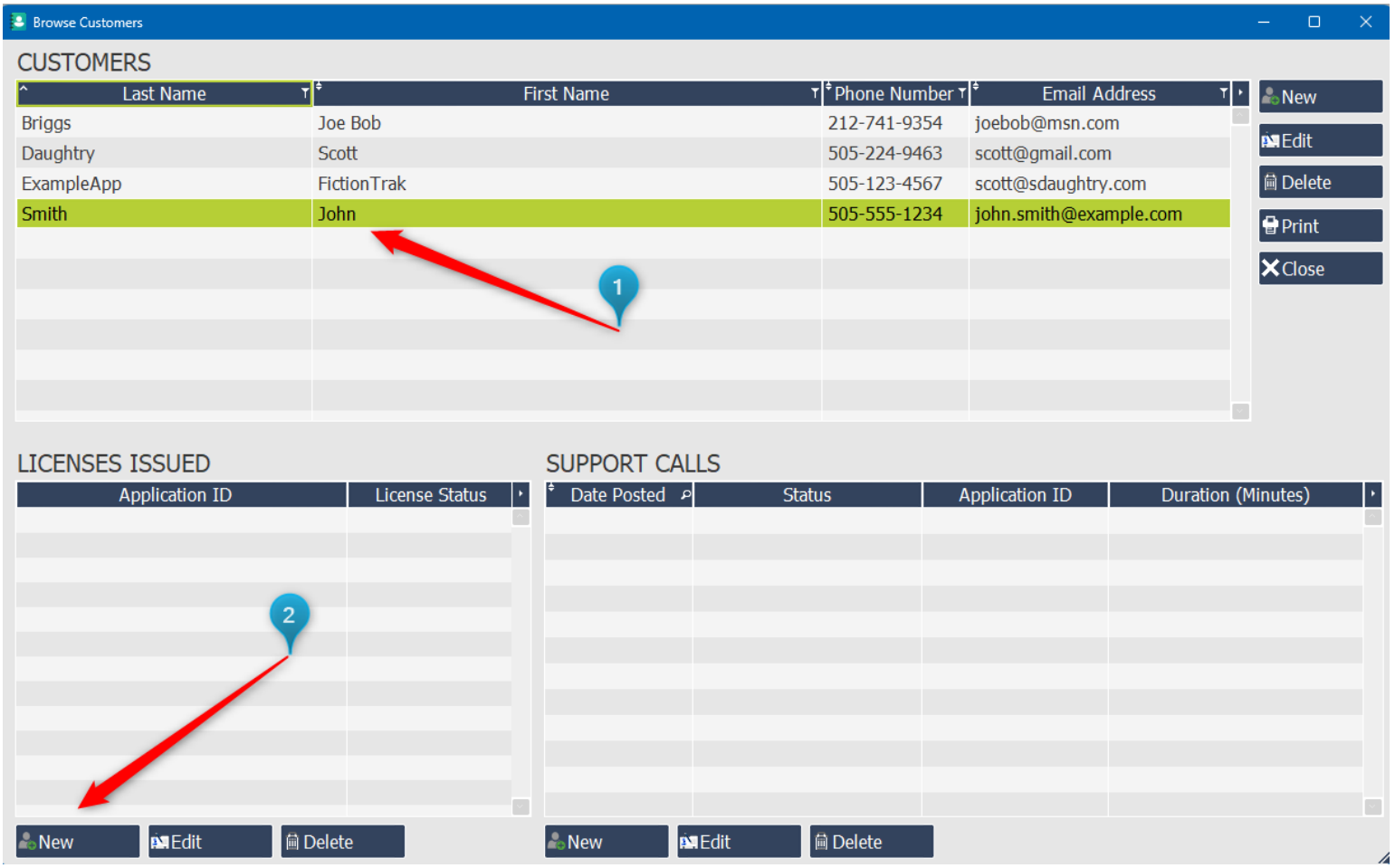
Generate a Trial License

Before a License.LIC file can be generated, both an Application record and a Customer record must exist within LicenseTrak.

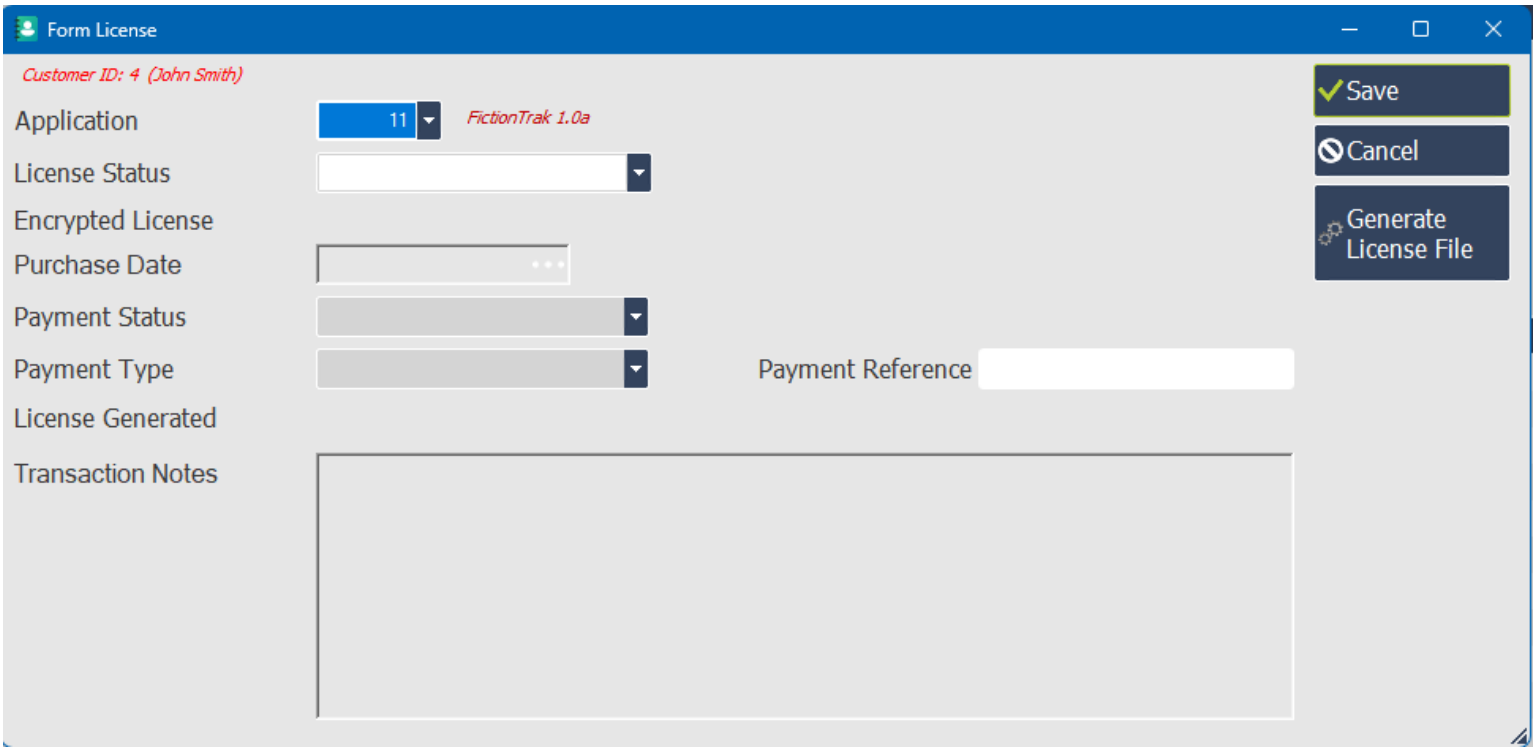
For this Quick Start tutorial, we will generate a Trial license for customer **John Smith** using the **FictionTrak 1.0a** application.

Opening the License Form

1. Select **File → Browse Customers** from the main menu.
2. Locate and select the **John Smith** customer record.
3. Click the **New** button located within the **Licenses Issued** section of the Customer Browse window.



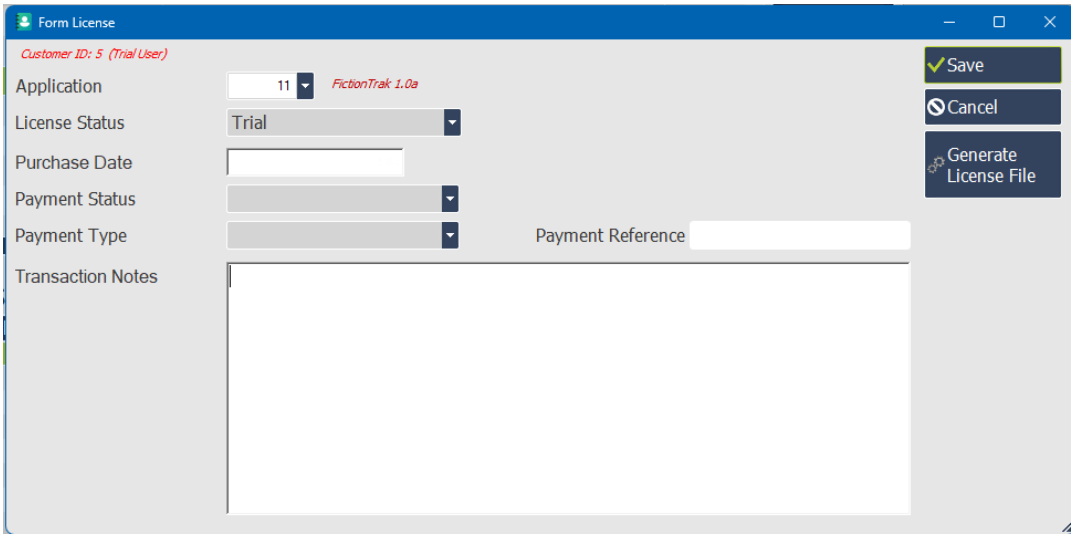
The **Form License** window is displayed:



Creating the Trial License

1. Select FictionTrak 1.0a from the Application dropdown list.
2. Set License Status to Trial.
3. Leave the remaining fields blank for this tutorial.
4. Click the Generate License File button.

The LicenseTrak License Generator window is displayed:



LicenseTrak now has all of the information necessary to generate a Trial License.LIC file for John Smith.

The next section explains how to configure the trial options and generate the physical License.LIC file that will be deployed with the protected application.

License.LIC Trial Application Deployment

The Trial License.LIC file generated during this tutorial is intended to be distributed with the FictionTrak demonstration version, which would be posted to the developer's web site for download.

Before creating the application's Setup.exe installation package, copy the Trial License.LIC file into the same folder as the FictionTrak executable program.

When the application is installed by a prospective customer, both FictionTrak.exe and License.LIC will be placed into the application's installation folder by the setup application.

Upon application startup, the LicenseTrak Runtime Library locates the License.LIC file and validates the licensing information contained within the file.

Because this is a Trial License.LIC file, the application will operate according to the evaluation restrictions defined by the developer.

The Trial License.LIC file is now ready to be distributed with the application's Setup.exe installation package.

Test the Application

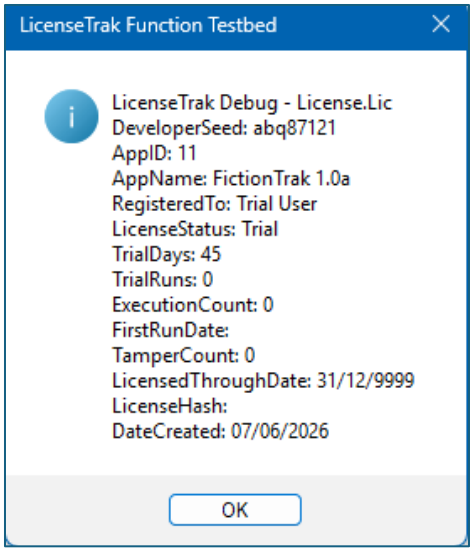
The final step in this Quick Start tutorial is to verify that the generated Trial License.LIC file can be successfully read and validated by the protected application.

Copy the Trial License.LIC file into the FictionTrak application folder and start the application.

For this tutorial, FictionTrak includes a LicenseTrak Runtime Testbed window that displays information retrieved from the License.LIC file.

After starting the application, verify that the LicenseTrak Runtime Library successfully reads the license information and displays values similar to the following:

- Registered To: Trial User
- License Status: Trial
- Trial Days: 45
- Trial Runs: 0
- Execution Count: 0
- Tamper Count: 0



The successful display of license information confirms that:

- The License.LIC file was located successfully.
- The License.LIC file was read successfully.
- The LicenseTrak Runtime Library initialized correctly.
- The license information passed validation.
- The application is operating under the Trial license.

Congratulations. You have successfully:

- Configured LicenseTrak.
- Created an Application record.
- Created a Trial User record.
- Generated a Trial License.LIC file.
- Deployed the License.LIC file.
- Verified successful Runtime Library operation.

You are now ready to explore the remaining chapters of this manual and implement LicenseTrak protection within your own WinDev applications.

PART I - USING LICENSETRAK

Chapter 4 - LicenseTrak Administration Program

Overview

The LicenseTrak Administration Program is used to create and manage the information required to generate software licenses for WinDev applications.

The Administration Program maintains three primary libraries:

1. Application Library
2. Customer Library
3. Licenses Issued

Together, these libraries provide the information required to generate Trial, Registered, and Locked License.LIC files.

In addition to license management, LicenseTrak provides a Support Calls tracking system that allows developers to document customer support interactions associated with licensed applications.

Configuration options are also available for language selection and Developer Seed management.

The following sections describe each area of the LicenseTrak Administration Program in detail.

Main Menu

The LicenseTrak Main Menu provides access to the application's primary functions and configuration options.

The menu structure has been intentionally designed to be simple and easy to navigate.

The Main Menu consists of two primary menu categories: FILE and CONFIGURATION

The File menu provides access to the Application Library, Customer Library, the Source Code repository, and program exit functions.

The Configuration menu provides access to language selection options, the Developer Seed, the default email message text, and generation of an empty/unencrypted License.LIC file for WinDev Analysis importation.

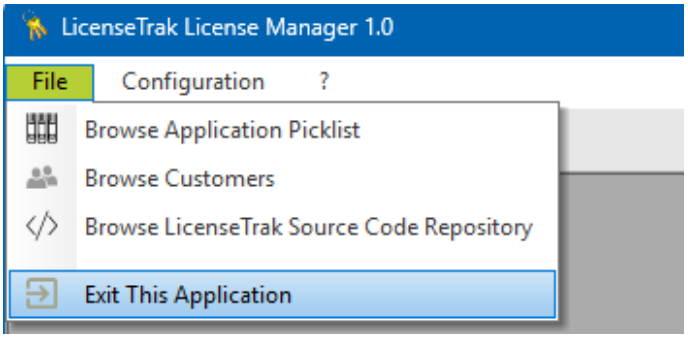
Each menu option is described in detail within the following sections.

File Menu

The File menu provides access to the primary data management functions within LicenseTrak.

The available menu options are:

- Browse Application Picklist
- Browse Customers
- Browse LicenseTrak Source Code Repository
- Exit This Application



Browse Application Picklist opens the Application Library, where software applications are defined and maintained.

Browse Customers opens the Customer Library, which serves as the central administrative screen for customer records, issued licenses, and support calls.

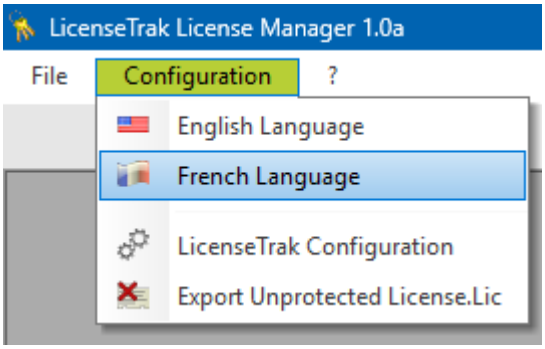
Browse LicenseTrak Source Code Repository displays a traditional browse/form window where you may store variations of your LicenseTrak .WL file for use across your software applications.

Exit This Application closes the LicenseTrak Administration Program and **returns** control to Windows.

Configuration Menu

The Configuration menu provides access to application settings and language options. The available menu options are:

- English Language
- French Language
- LicenseTrak Configuration
- Export Unprotected License.LIC



Selecting English Language immediately switches the user interface to English.

Selecting French Language immediately switches the user interface to French.

Selecting Developer Seed displays the Developer Seed configuration window, which is used to define the unique Developer Seed value used during license generation and validation.

Application Library

The Application Library contains the software applications that may be protected by LicenseTrak.

Before a License.LIC file can be generated, an Application record must exist within the Application Library.

Each Application record contains identifying information used during license generation, including the application name, version number, release date, and developer notes.

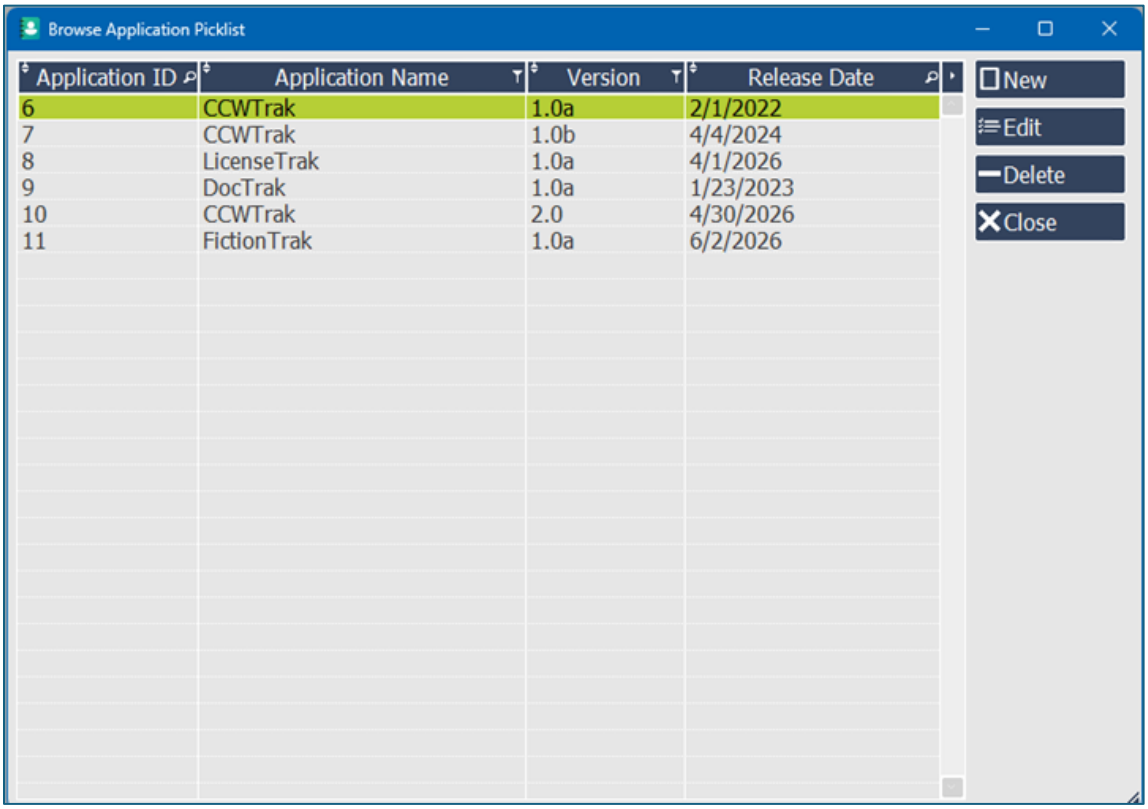
Applications are maintained through the Browse Application Picklist window.

Browse Application Picklist

The Browse Application Picklist window displays all applications currently defined within LicenseTrak.

The following information is displayed for each application:

- Application ID
- Application Name
- Version
- Release Date



The buttons available on this screen are:

- New
- Edit
- Delete

- Close

Selecting an application and clicking Edit displays the Application Form populated with the selected application's information.

Clicking New displays a blank Application Form for the creation of a new application record.

The Delete button permanently removes the selected application record from the Application Library.

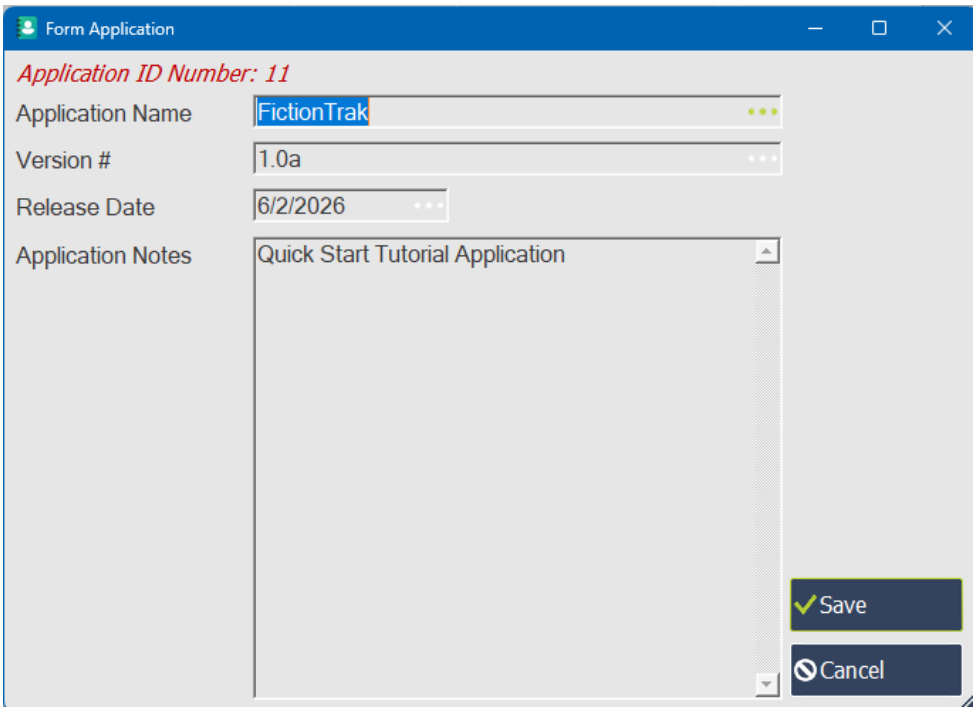
The Close button **returns** to the LicenseTrak Main Menu.

Creating an Application Record

Both the New and Edit buttons display the Application Form window.

When opened using the New button, all entry fields are initially blank.

When opened using the Edit button, the form fields are populated using the information stored within the selected Application record.



The Application Form contains the following fields:

- Application Name
- Version Number
- Release Date
- Application Notes

The Release Date field includes a popup calendar control that may be used to select a date from a calendar rather than manually entering a date value.

Application Notes may be used to record version history, release information, licensing requirements, or other developer-specific information related to the application.

Click Save to create or update the Application record.

Click Cancel to close the form without saving changes.

Editing and Deleting Applications from the Picklist

Application records may be modified or removed directly from the Browse Application Picklist window.

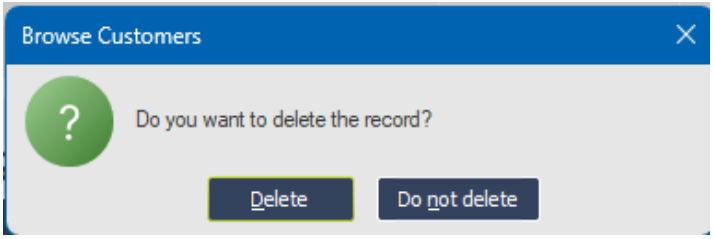
To edit an application:

1. Select the desired application from the Browse Application Picklist.
2. Click the Edit button.
3. Modify the desired information within the Application Form.
4. Click Save to commit the changes.

The updated information is immediately reflected within the Application Library.

To delete an application:

1. Select the desired application from the Browse Application Picklist.
2. Click the Delete button
3. Confirm the deletion when prompted:



The selected application record is permanently removed from the Application Library.

Developers should exercise caution when deleting application records. If licenses have previously been generated for an application, deleting the associated Application record may make future license management activities more difficult.

For this reason, many developers choose to retain Application records indefinitely and use the Application Notes field to document version history, retirement status, or other administrative information.

Customer Library

The Customer Library is used to create and maintain customer information within LicenseTrak.

Customer records serve as the foundation for license generation and support call tracking activities. Before a license can be generated, a Customer record must exist within the Customer Library.

Customer information is maintained through the Browse Customers window.

In addition to customer information, the Browse Customers window provides access to customer-related License records and Support Call records.

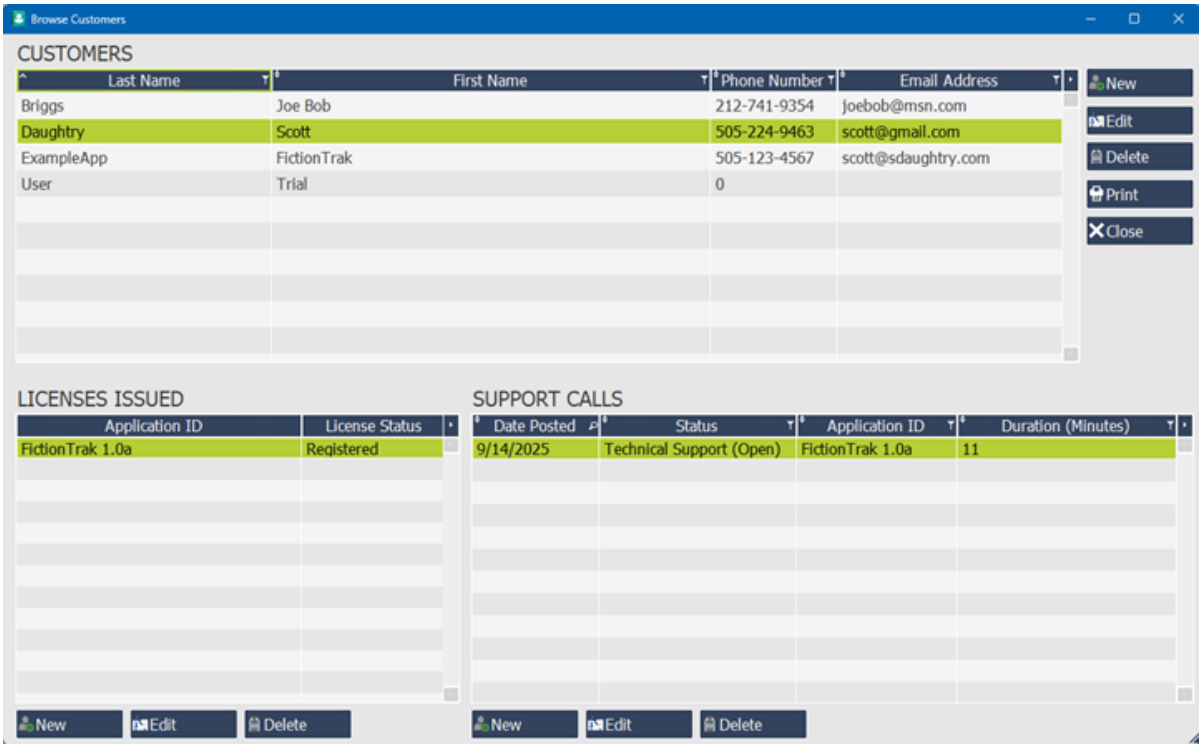
As a result, the Customer Library serves as the central administrative hub of the LicenseTrak Administration Program.

Customer Browse Window

The Browse Customers window displays all customer records currently stored within LicenseTrak.

The window is divided into three functional areas:

- Customers
- Licenses Issued
- Support Calls



The Customers section displays the customer database records maintained within LicenseTrak.

The Licenses Issued section displays all licenses associated with the currently selected customer.

The Support Calls section displays all support calls associated with the currently selected customer.

Selecting a different customer automatically refreshes the Licenses Issued and Support Calls sections to display only records associated with the highlighted customer.

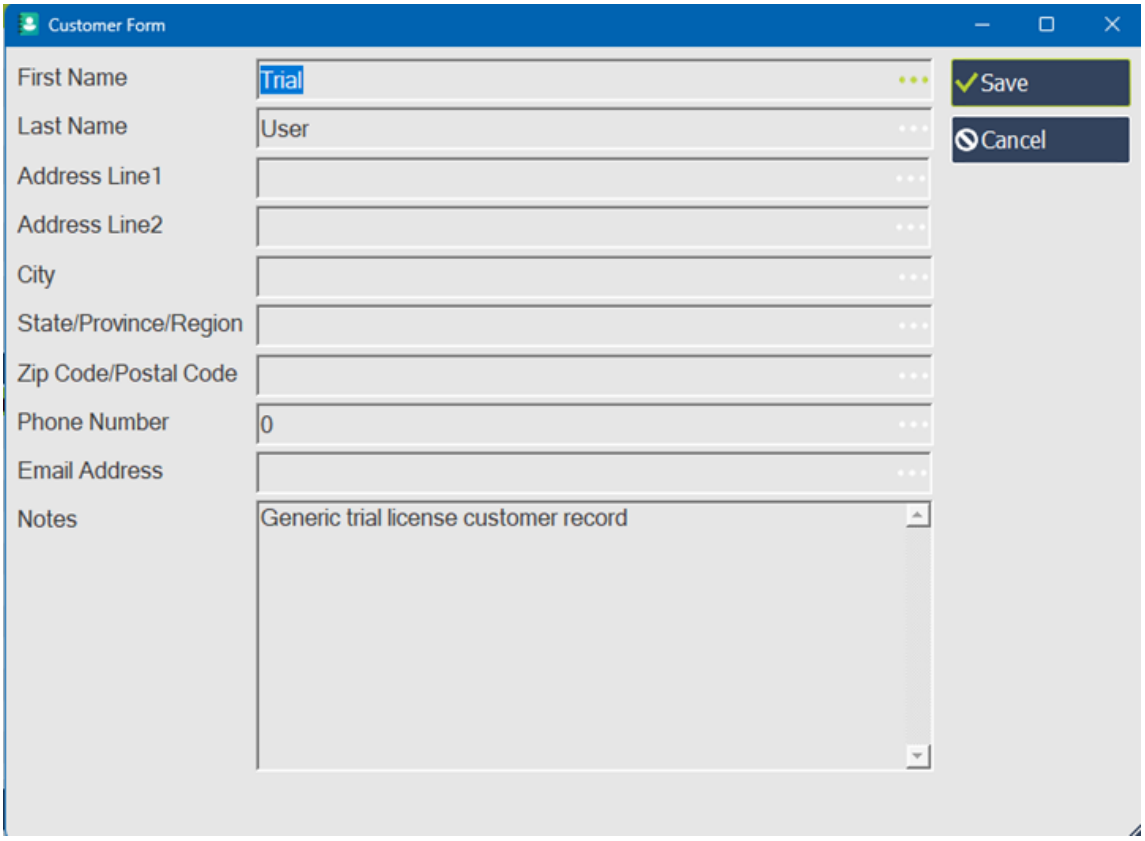
This relationship provides a convenient method of viewing a customer's licensing history and support history from a single administrative screen.

Creating a Customer Record

Customer records are created and maintained through the Customer Form window.

To create a new customer:

1. Open the Browse Customers window.
2. Click the New button located within the Customers section.
3. Enter the desired customer information.
4. Click Save.



The Customer Form contains the following fields:

- First Name
- Last Name
- Address Line 1
- Address Line 2
- City
- State/Province/Region
- Zip Code/Postal Code
- Phone Number
- Email Address
- Notes

Customer records may represent Trial Users, individual customers, organizations, or other entities requiring software

licenses.

The information entered within the Customer Form may be modified at any time by selecting the customer and clicking the Edit button.

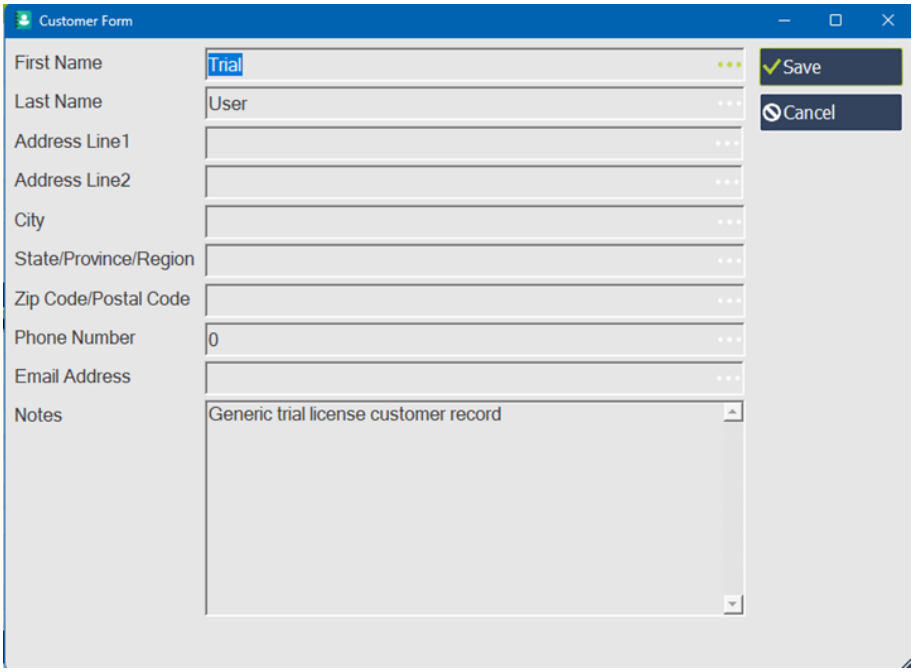
Editing and Deleting Customers

Existing customer records may be modified or removed directly from the Browse Customers window.

To edit a customer:

1. Select the desired customer record.
2. Click the Edit button located within the Customers section.
3. Modify the desired information.
4. Click Save to commit the changes.

The updated information is immediately reflected within the Customer Library.



To delete a customer:

1. Select the desired customer record.
2. Click the Delete button located within the Customers section.
3. Confirm the deletion when prompted.

If the selected customer does not have any associated License or Support Call records, the customer record is permanently removed from the database.

Customer Relationships

LicenseTrak uses relational integrity rules to maintain consistency between Customers, Licenses Issued, and Support Calls.

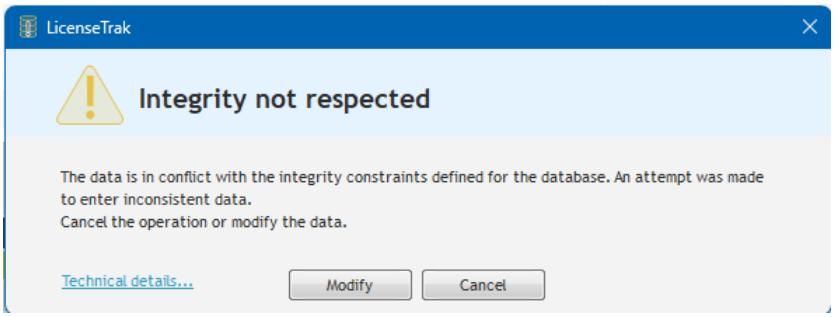
Because License and Support Call records are linked to a specific customer, LicenseTrak prevents the deletion of a customer record when related records exist.

For **example**, if a customer has:

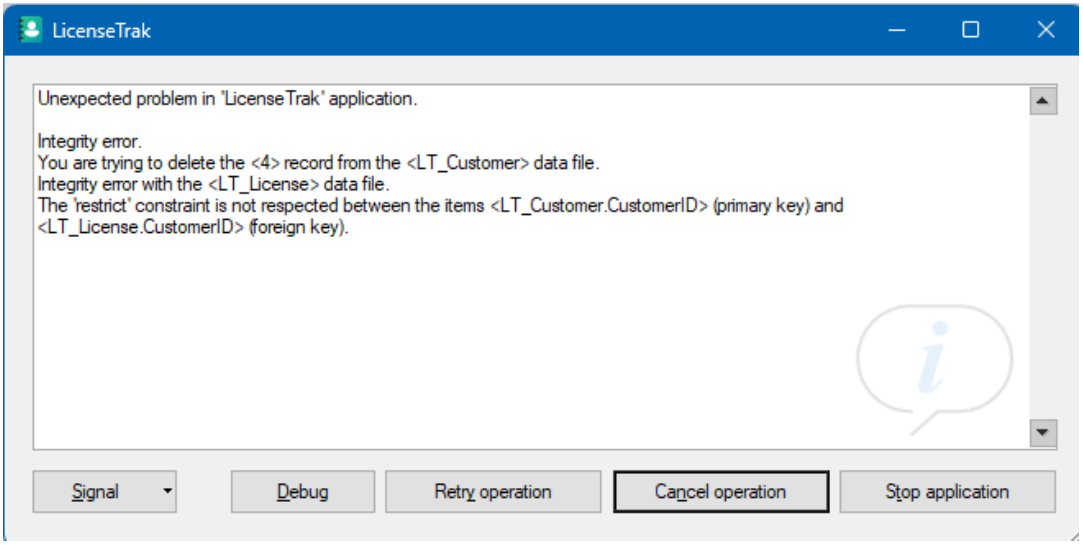
• One or more License records **or** One or more Support Call records

the customer record cannot be deleted until the related records have been removed.

Attempting to delete such a customer causes LicenseTrak to display an informational message indicating that related records exist.



Click the CANCEL button; a second warning screen informs why the record deletion cannot occur:



Click **CANCEL OPERATION** to return back to the LicenseTrak application.

These safeguards help prevent accidental loss of licensing history and customer support information.

Because software customers may require replacement licenses years after their original purchase, *it is generally recommended that Customer records be retained indefinitely whenever practical.* Maintaining complete customer

histories simplifies license replacement, customer support, and historical record keeping activities.

Licenses Issued

The Licenses Issued section of the Browse Customers window is used to create and maintain software licenses associated with a customer.

Each License record links a customer to a specific application and contains information related to licensing status, purchase information, and license generation activities.

License records serve as the source information used when generating License.LIC files.

A customer may possess multiple licenses for different applications, versions, or licensing arrangements.

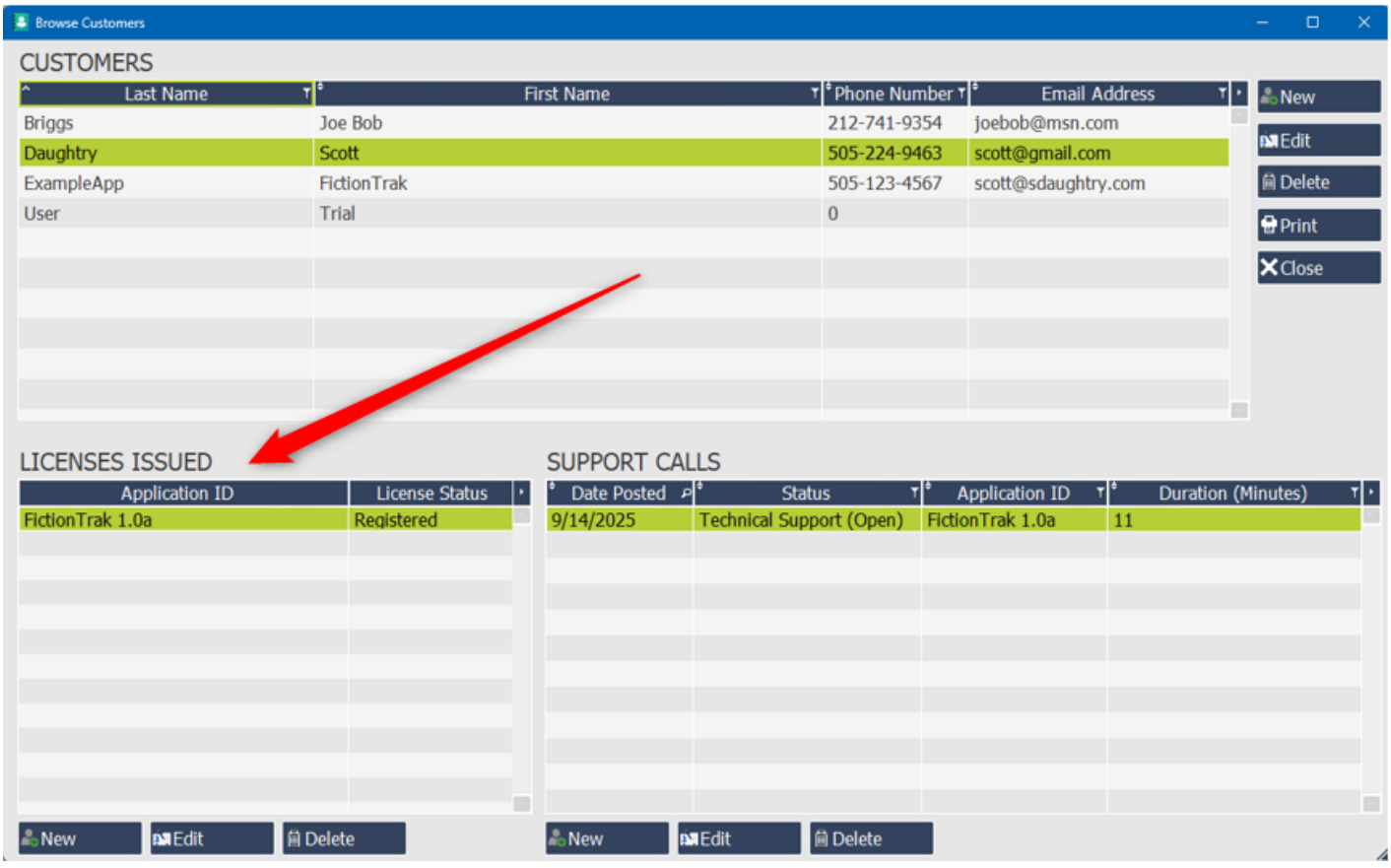
All licenses associated with the currently selected customer are displayed within the Licenses Issued section.

License Browse Area

The Licenses Issued section displays all license records associated with the currently selected customer.

The buttons available within this section are:

- New
- Edit
- Delete



Clicking New displays a blank License Form for the creation of a new license record.

Clicking Edit displays the License Form populated with information from the selected license record.

Clicking Delete permanently removes the selected license record from the database.

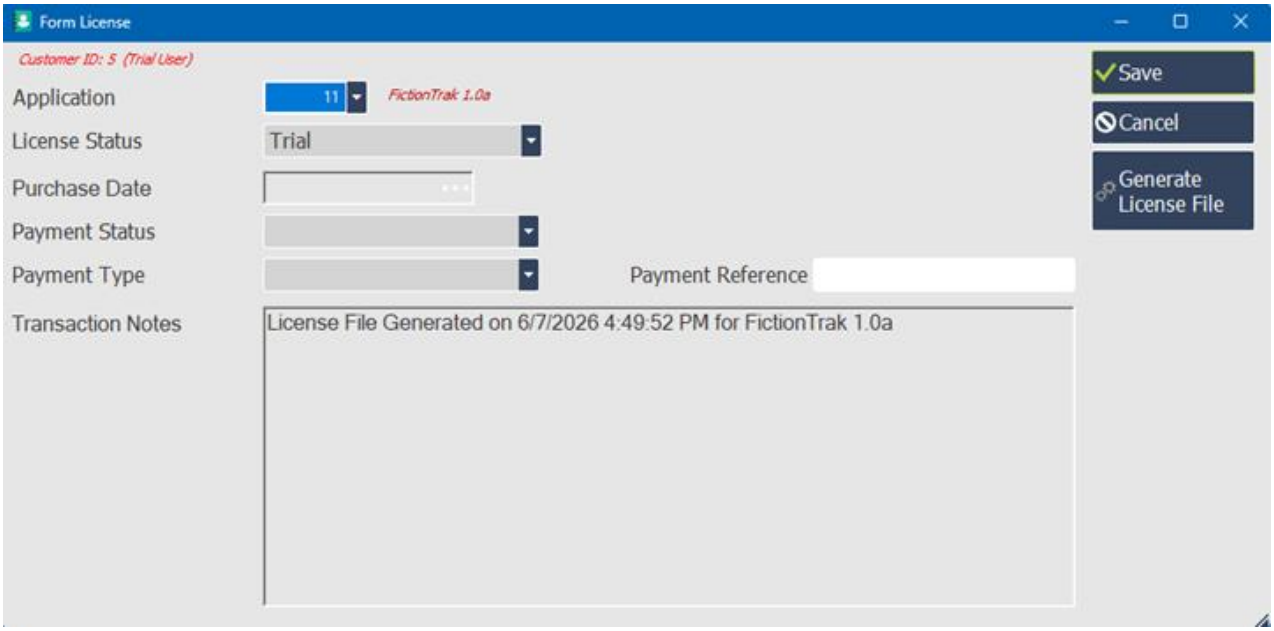
The displayed license information provides a convenient method of reviewing a customer's licensing history directly from the Browse Customers window.

Creating a License Record

License records are created and maintained through the License Form window.

To create a new license:

1. Select a customer within the Browse Customers window.
2. Click the New button located within the Licenses Issued section.
3. Enter the desired license information.
4. Click Save to create the License record.



The License Form contains the following information:

- Application
- License Status
- Purchase Date
- Payment Status
- Payment Type
- Check Number
- Amount Paid
- Notes

The Application dropdown list displays entries maintained within the Application Library.

The License Status dropdown list determines the licensing state that will be associated with the generated License.LIC file.

Purchase and payment information may be recorded for administrative and accounting purposes.

The Notes field may be used to document special licensing arrangements, customer requests, upgrade information, or other license-related details.

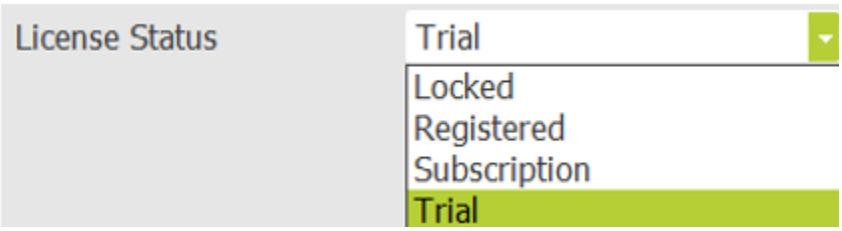
Once the License record has been created, the Generate License File button may be used to create a License.LIC file associated with the selected customer and application.

License Status

The License Status field determines the licensing state associated with the generated License.LIC file.

The available License Status values are:

- Locked
- Registered
- Subscription
- Trial



Trial licenses are typically used for software evaluation and demonstration purposes.

Registered licenses indicate that a customer has purchased the software and is authorized to use the application according to the developer's licensing policies.

Locked licenses prevent normal application operation and are typically used when application access should be restricted.

The selected License Status value is incorporated into the generated License.LIC file and may be evaluated by the LicenseTrak Runtime Library during application execution.

Payment Information

The License Form provides several fields for recording payment-related information associated with a software license. The available payment fields are:

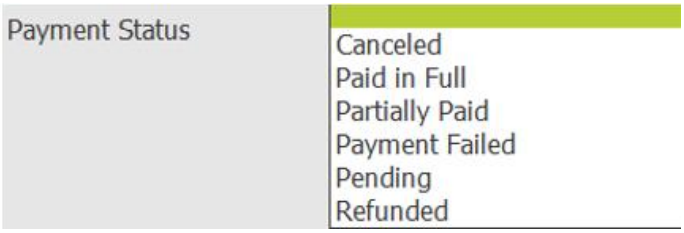
- Payment Status
- Payment Type
- Payment Reference

- Amount Paid

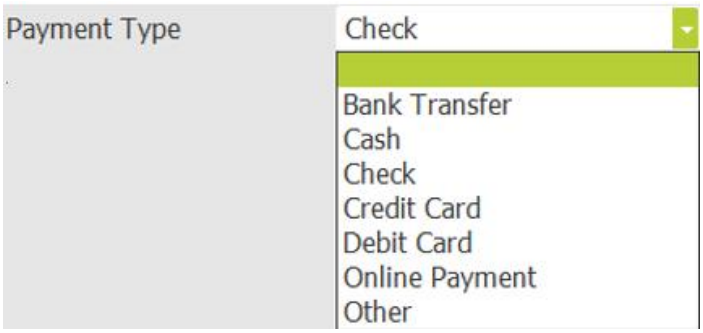
These fields are provided for administrative record keeping purposes and are not evaluated by the LicenseTrak Runtime Library during application execution.

Payment information may be useful when tracking software sales, customer purchases, renewals, upgrades, maintenance agreements, or other licensing transactions.

The Payment Status field may be used to indicate whether payment has been received, is pending, has been refunded, or is associated with another developer-defined status.



The Payment Type field identifies the payment method used by the customer, such as cash, check, credit card, PayPal, or other forms of payment.



The Payment Reference field may be used to record a customer check number or other transaction reference identifier.

The Amount Paid field records the amount received for the license transaction.

Because these fields are stored within the LicenseTrak database, developers can maintain historical payment information alongside customer and licensing records without requiring a separate tracking system.

Editing and Deleting Licenses

Existing license records may be modified or removed directly from the Licenses Issued section of the Browse Customers window.

To edit a license:

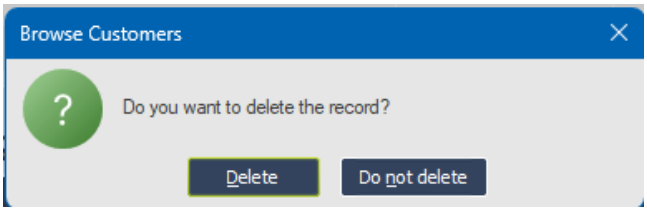
1. Select the desired license record.
2. Click the Edit button.
3. Modify the desired information.
4. Click Save to commit the changes.

The updated information is immediately reflected within the customer's licensing history.

To delete a license:

1. Select the desired license record.
2. Click the Delete button.
3. Confirm the deletion when prompted.

The selected license record is permanently removed from the database.



Developers should exercise caution when deleting license records. Historical license information may prove valuable when troubleshooting customer issues, recreating lost License.LIC files, researching prior purchases, or reviewing a customer's licensing history.

For this reason, many developers choose to retain license records indefinitely whenever practical.

Generating a License.LIC file

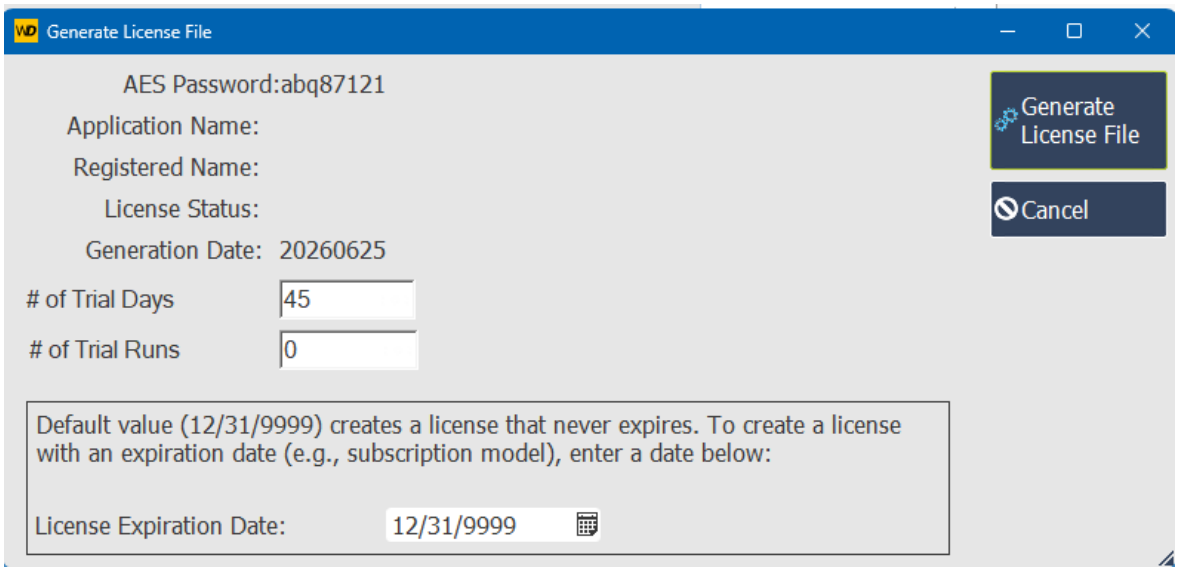
The Generate License File window is used to create the physical License.LIC file that will be distributed with a protected application.

This window combines information from the selected Application record, Customer record, License record, and Developer Seed configuration to generate a License.LIC file suitable for deployment.

To display the Generate License File window:

1. Open the Browse Customers window.
2. Select the desired customer.
3. Select the desired license record.
4. Click the Edit button within the Licenses Issued section.
5. Click the Generate License File button.

The Generate License File window is displayed:



The window displays information associated with the selected customer, application, and license record.

Displayed information includes:

- AES Encryption Password (this is the Developer Seed; it is listed here in case you need to manually edit it within WinDev)
- Application Name
- Registered Name
- License Status
- Generation Date

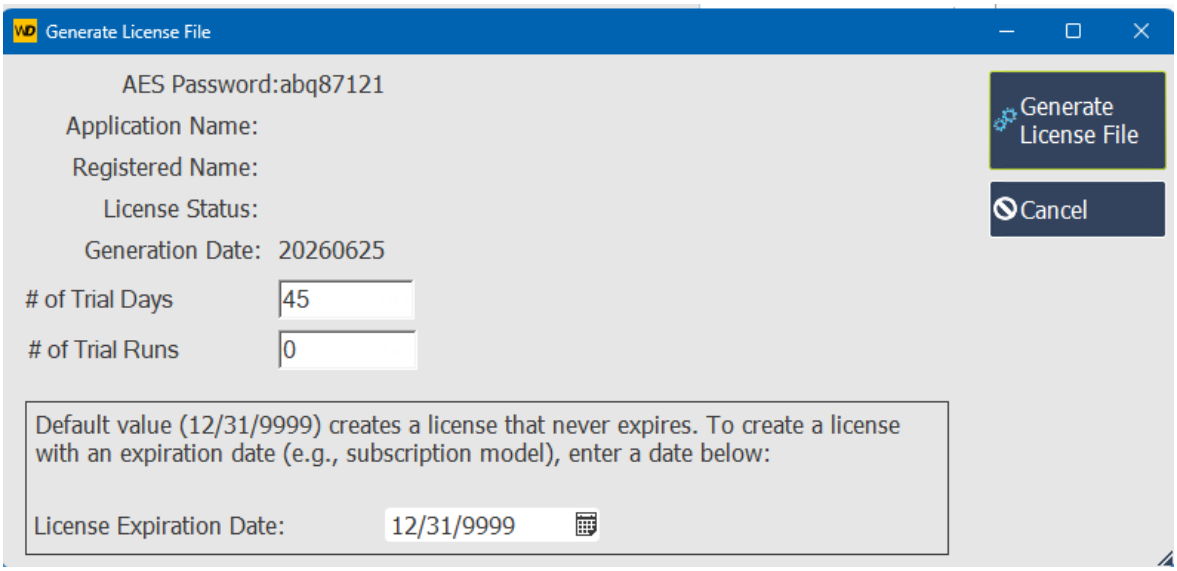
In addition to the displayed information, the developer may specify trial and expiration settings that will be incorporated into the generated License.LIC file.

Trial Days, Trial Runs, and Expiration Dates

The Generate License File window allows developers to define trial restrictions and expiration dates that will be incorporated into the generated License.LIC file.

The available fields are:

- Number of Trial Days
- Number of Trial Runs
- License Expiration Date



Trial Days

The Number of Trial Days field specifies how many calendar days the application may be used after the first successful execution.

For **example**, a value of 30 permits application usage for thirty days.

Trial Runs

The Number of Trial Runs field specifies how many times the application may be executed.

For **example**, a value of 50 permits fifty application launches regardless of how much time has elapsed.

A value of zero indicates that no Trial Runs restriction has been defined.

License Expiration Date

The License Expiration Date field specifies the date after which the license should no longer be considered valid.

This field is commonly used when implementing subscription-based licensing, annual maintenance agreements, educational licensing, or other time-limited licensing models.

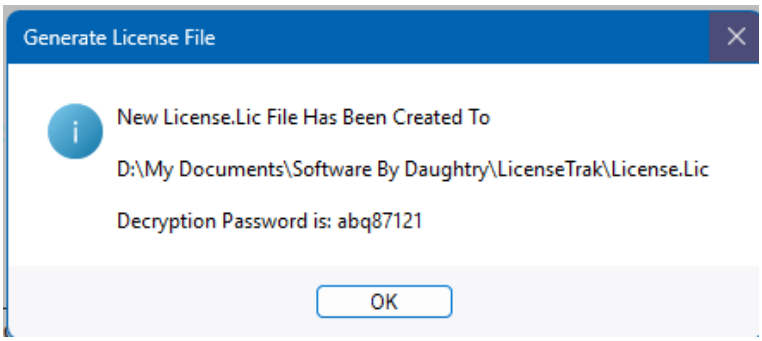
Developers may use Trial Days, Trial Runs, Licensed Expiration Dates, or any combination thereof to create licensing models appropriate for their applications.

Creating the License.LIC File

After all desired licensing options have been specified, click the Generate License File button.

LicenseTrak creates a License.LIC file containing the licensing information associated with the selected customer, application, and license record.

Upon successful generation, LicenseTrak displays a confirmation message indicating that the License.LIC file has been created successfully.



The generated License.LIC file may then be:

- Included within a software installation package.
- Distributed to a customer via email.
- Provided through a customer download portal.
- Delivered using any other method preferred by the software publisher.

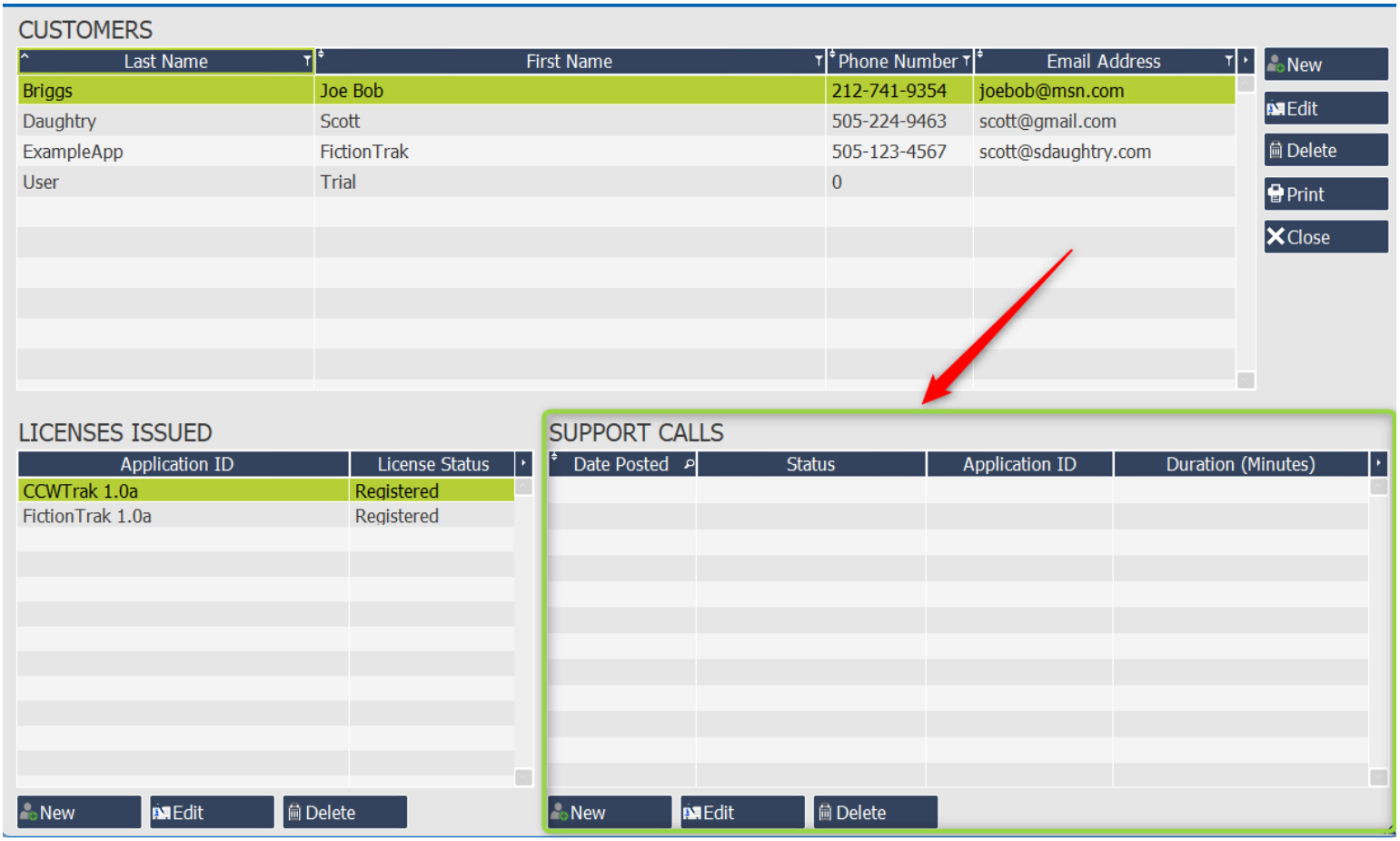
Because the License.LIC file contains all information required by the LicenseTrak Runtime Library, no Internet activation, online validation, cloud licensing service, or external licensing server is required.

The generated License.LIC file is immediately ready for deployment.

Support Calls

The Support Calls section of LicenseTrak is used to record and maintain customer support interactions.

Support Call records are associated with a specific customer and provide a historical record of support activities related to licensed applications.



Maintaining support histories can simplify troubleshooting, improve customer service, and provide valuable context during future customer interactions.

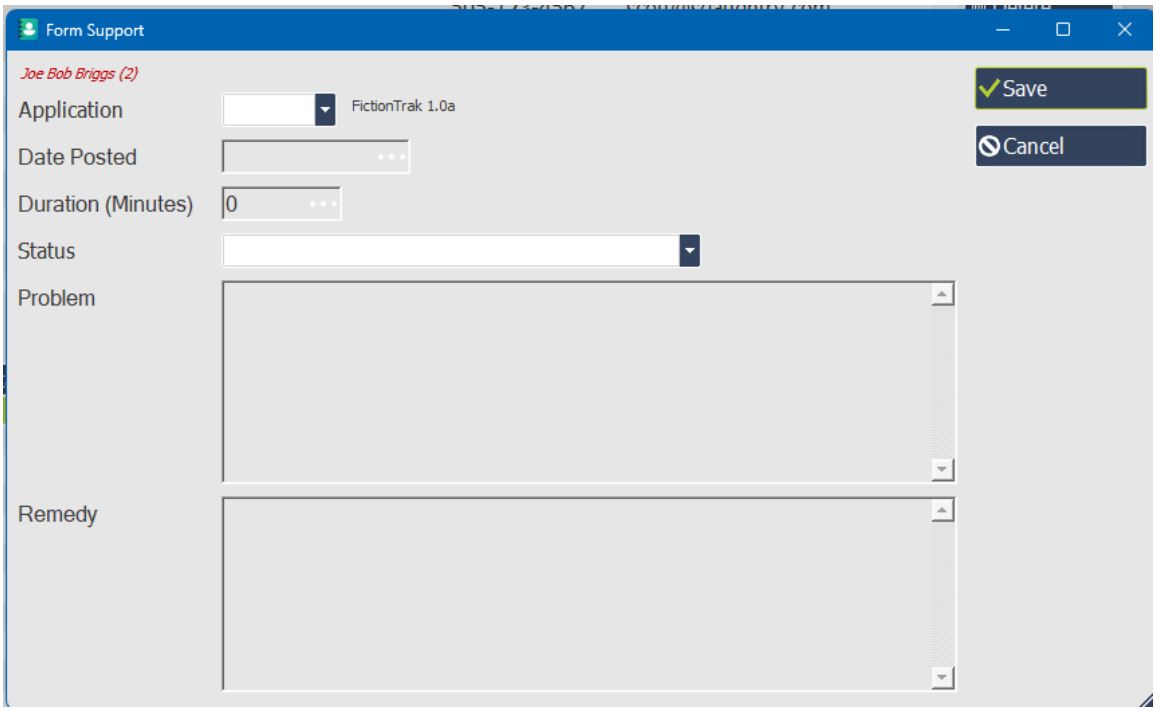
All support calls associated with the currently selected customer are displayed within the Support Calls section of the Browse Customers window.

Creating Support Calls

Support Call records are created through the Support Call Form window.

To create a new Support Call record:

1. Open the Browse Customers window.
2. Select the desired customer.
3. Click the New button located within the Support Calls section.
4. Complete the Support Call Form.
5. Click Save.



The Support Call Form contains information related to the customer interaction, including:

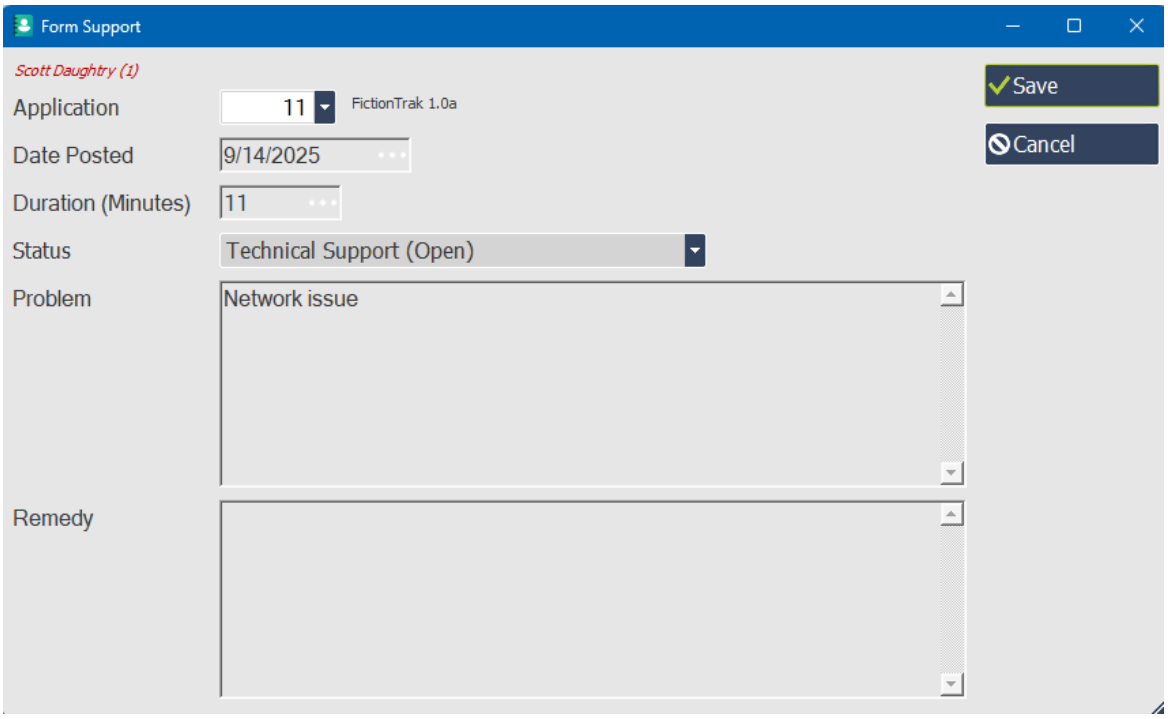
- Date Posted
- Application
- Support Status
- Duration
- Notes

Support Call records provide a permanent history of customer support activities and may be reviewed at any time from the Customer Library.

Editing Support Calls

Existing Support Call records may be modified at any time. To edit a Support Call record:

1. Open the Browse Customers window.
2. Select the desired customer.
3. Select the desired Support Call record.
4. Click the Edit button located within the Support Calls section.
5. Modify the desired information.
6. Click Save.



Typical reasons for editing a Support Call record include:

- Updating support status.
- Correcting entered information.
- Recording additional troubleshooting details.
- Documenting support resolution information.

The updated information is immediately available within the customer's support history.

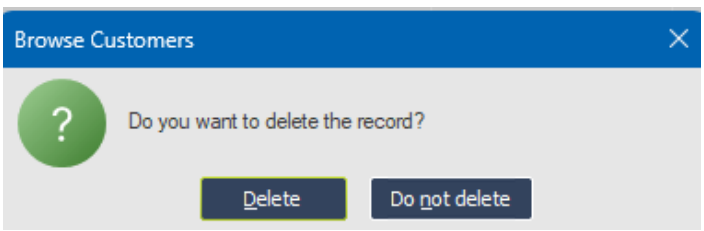
Deleting Support Calls

Support Call records may be removed from the Support Calls section when they are no longer required.

To delete a Support Call record:

1. Open the Browse Customers window.
2. Select the desired customer.
3. Select the desired Support Call record.
4. Click the Delete button located within the Support Calls section.
5. Confirm the deletion when prompted.

The selected Support Call record is permanently removed from the database.



Developers should carefully consider whether support history information may be useful in the future before deleting Support Call records.

Historical support records often provide valuable troubleshooting information and customer context that may prove useful during future customer interactions.

Best Practices

The following recommendations have proven useful when maintaining Support Call records within LicenseTrak.

Record Support Interactions Promptly

Enter Support Call information as soon as practical following a customer interaction. Accurate information is easier to capture while details remain fresh.

Document The Resolution

Whenever possible, record the final resolution within the Support Call Notes field. Future support efforts may benefit from understanding how similar issues were previously resolved.

Include Relevant Details

Document error messages, troubleshooting steps, configuration information, and other details that may assist future support activities.

Track Time Consistently

Record support duration information consistently. Historical duration data may provide useful insight into support requirements and customer service activities.

Maintain Historical Records

Support histories often become more valuable over time. Similar issues frequently reappear months or years later, making historical records a valuable troubleshooting resource.

Link Support To Licensing

Support records become even more valuable when viewed alongside a customer's licensing history. LicenseTrak's integrated customer view provides immediate access to both types of information.

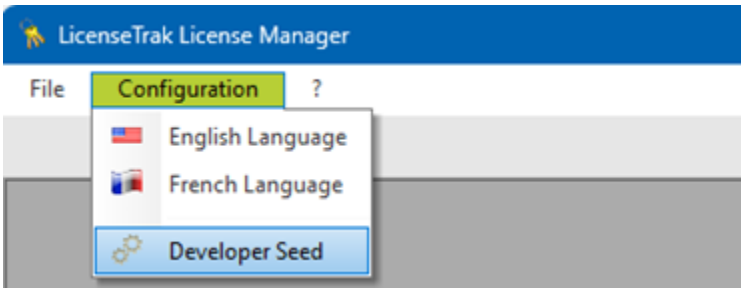
Think Like Future You

A detailed Support Call record may save hours of troubleshooting effort years later. Information that seems obvious today may not be obvious after dozens of software releases and hundreds of customer interactions.

Language Selection

LicenseTrak supports both English and French user interface languages.

Language selection is available from the Configuration menu.



To change the displayed language:

1. Select Configuration.
2. Select English Language or French Language.

The selected language is applied immediately throughout the LicenseTrak Administration Program.

No application restart is required.

Language selection affects menu items, prompts, messages, and other user interface elements displayed by the Administration Program.

Developers may switch between languages at any time.

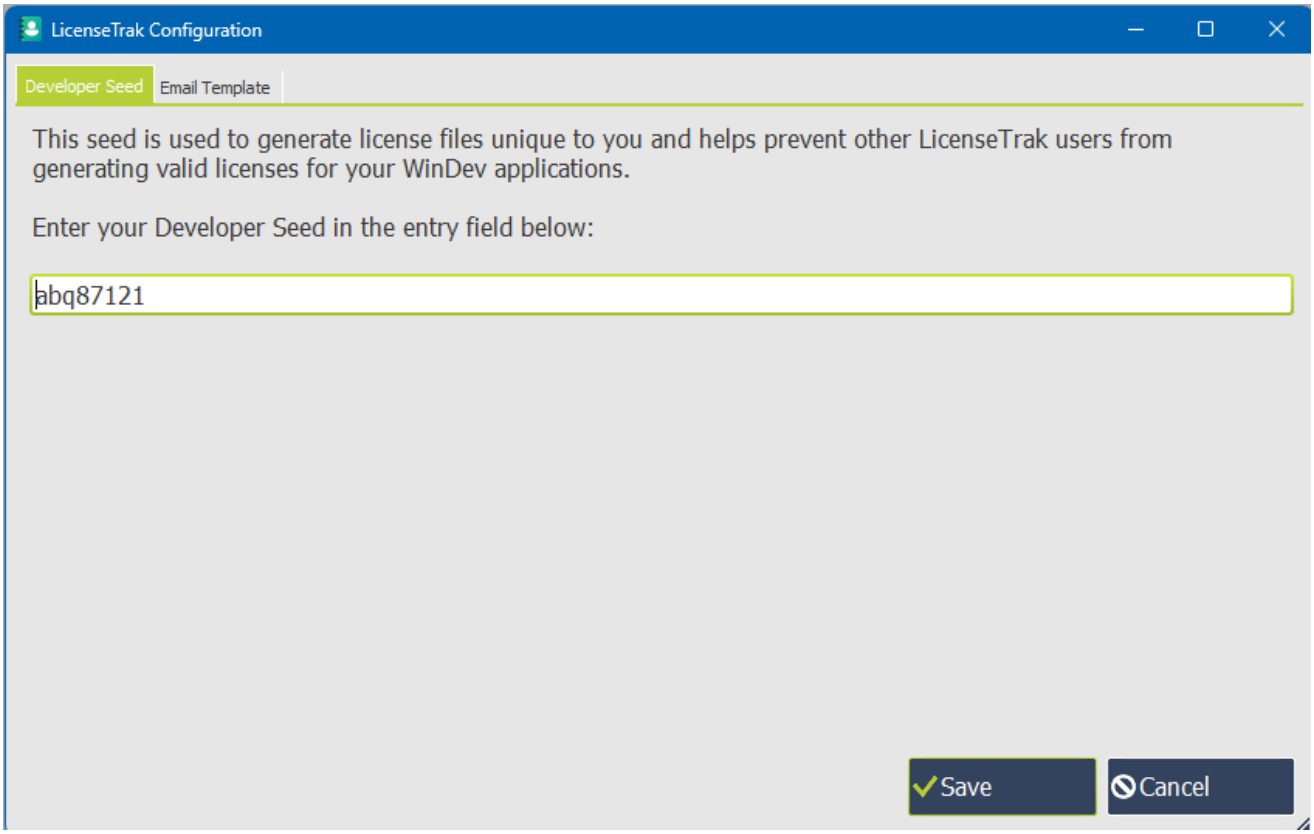
Configuration: Developer Seed and Email Template

This menu option displays a window to configure the Developer Seed and the Email Template:

Developer Seed

The **Developer Seed** tab defines the unique seed value used during license generation and validation. This value is written to the LicenseTrak.ini file and is required whenever LicenseTrak generates or validates a **License.LIC** file. The Developer Seed should be considered confidential information. Once licenses have been distributed to customers, changing this value may invalidate previously generated licenses. For this reason, the Developer Seed should remain constant for the lifetime of the application.

Your screen should resemble the following:

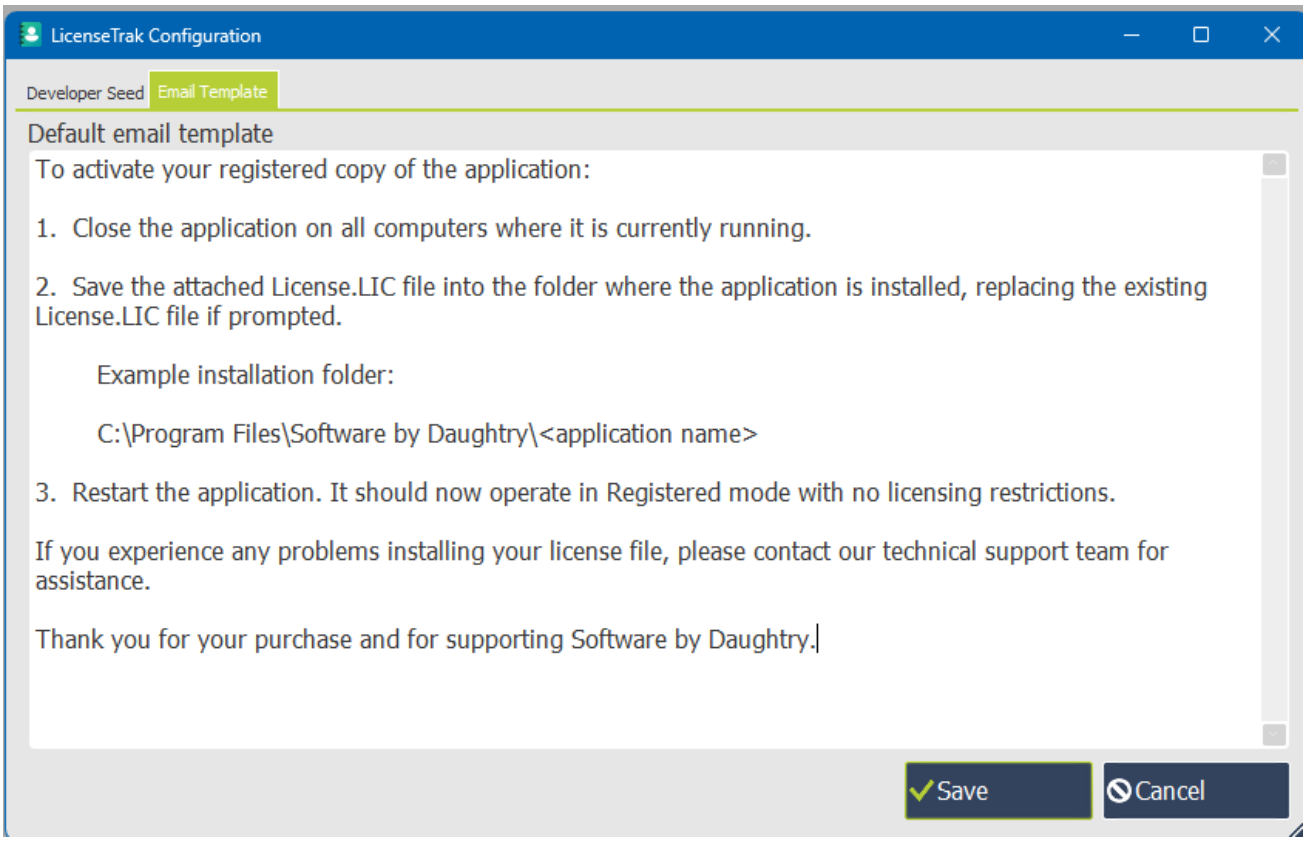


Email Template

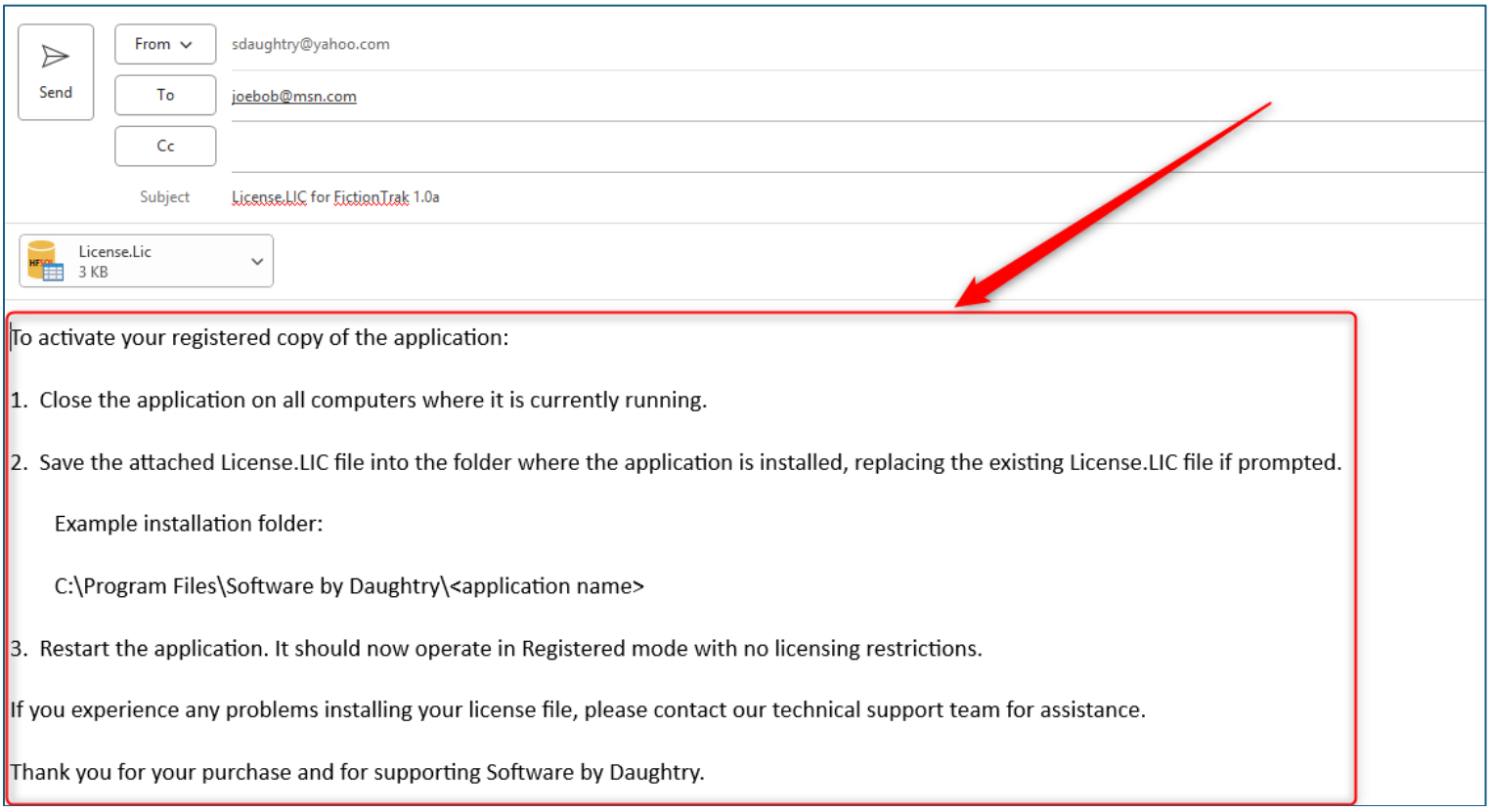
The **Email Template** tab stores the default email message used when distributing generated **License.LIC** files to customers. Defining a standard template helps ensure that each customer receives consistent installation instructions and licensing information.

The template may be edited at any time. Changes affect future email messages but do not modify licenses that have already been generated or distributed.

Your screen should resemble the following:



The text entered here is used within the LicenseTrak IDE to generate an email which contains the License.LIC file to the email address associated with the customer:



Note
The Developer Seed is stored within the LicenseTrak.ini file. **Example:**

```
[Main]
Seed=abq87121
```

The Developer Seed should be considered confidential information and should be backed up regularly.

Because previously generated License.LIC files are associated with the Developer Seed value used during their creation, developers should exercise caution when modifying an existing Developer Seed.

Loss of the Developer Seed may complicate future license generation and replacement activities.

Developer Seed Best Practices

The following recommendations have proven useful when managing Developer Seed values:

Define The Seed Once

Select a Developer Seed value carefully and avoid changing it unless absolutely necessary.

Maintain Secure Backups

Include the LicenseTrak.ini file within normal backup procedures.

Store The Seed Separately

Maintain a secondary record of the Developer Seed in a secure location.

Treat The Seed As Critical Information

The Developer Seed is a foundational component of the LicenseTrak licensing architecture and should be protected accordingly.

Think Long-Term

Future license generation, customer support, and license replacement activities may depend upon access to the original Developer Seed value.

Source Code Repository

The Source Code Repository stores one or more versions of the LicenseTrak Runtime Library (.WL) within the encrypted LicenseTrak database. Rather than distributing the Runtime Library as a separate source file, LicenseTrak securely stores each version internally, allowing Software by Daughtry to maintain multiple Runtime Library releases from a single location.

Registered LicenseTrak users may browse the repository, review available Runtime Library versions, and export the desired .WL source code file directly to a folder of their choosing. The exported file can then be imported into an existing WinDev project as described in Chapter 10.

This repository-based approach provides several advantages. It allows multiple Runtime Library revisions to coexist within a single database, preserves older versions for compatibility with existing customer applications, and ensures that only registered LicenseTrak users have access to the distributable source code.

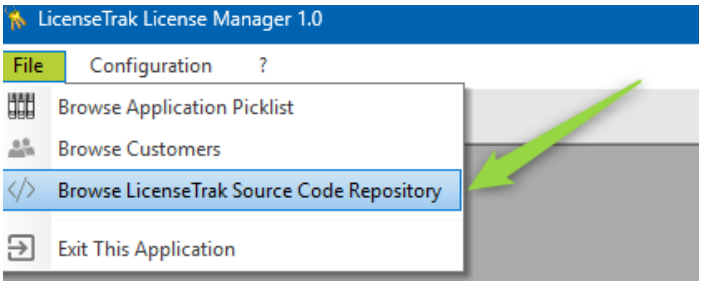
As future LicenseTrak releases become available, additional Runtime Library versions may be added to the repository without affecting previously released versions, allowing developers to continue maintaining older applications while taking advantage of new LicenseTrak capabilities in future projects.

Browsing the Source Code Repository

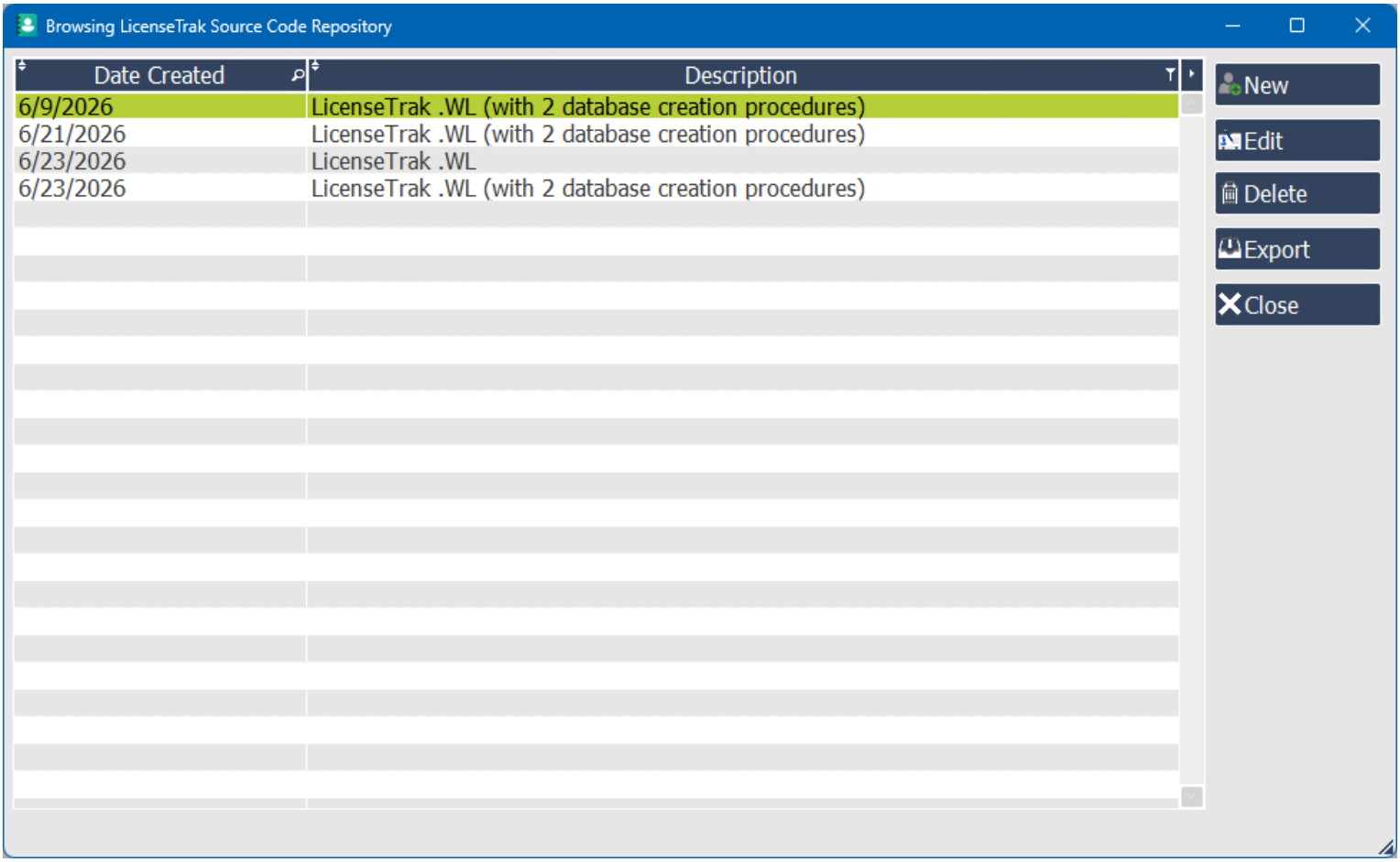
The Source Code Repository provides a centralized location for storing one or more versions of the LicenseTrak Runtime Library (.WL). Each record represents a complete Runtime Library that may be exported and incorporated into one or more WinDev applications.

The repository is designed to simplify Runtime Library management while providing a historical archive of previous releases. As new LicenseTrak versions are developed, additional Runtime Libraries may be added to the repository without affecting existing entries. This allows developers to continue supporting previously deployed applications while simultaneously taking advantage of newer LicenseTrak features in future projects.

To access the Source Code Repository, select **File → Browse LicenseTrak Source Code** from the LicenseTrak main menu.



The Source Code Repository browse window is then displayed.



Each row within the browse window represents a complete Runtime Library version. The browse displays the following information:

• **Date Created** – Indicates when the Runtime Library record was created within the repository.

• **Description** – A brief description identifying the Runtime Library version, its intended purpose, or significant characteristics. Examples might include a production release, a development build, or a Runtime Library containing additional demonstration procedures.

The buttons located along the right side of the browse window provide the following functions:

New
Creates a new Runtime Library repository record. This function is typically used by Software by Daughtry when maintaining new Runtime Library versions or internal development releases.

Edit
Opens the selected Runtime Library record, allowing its descriptive information and embedded source code to be viewed or modified.

Delete
Removes the selected Runtime Library from the repository. This function should be used cautiously, as deleting a Runtime Library permanently removes that version from the repository.

Export
Exports the selected Runtime Library to a standard WinDev (.WL) source file. Registered LicenseTrak users may then import the exported Runtime Library into their own WinDev projects as described in Chapter 10.

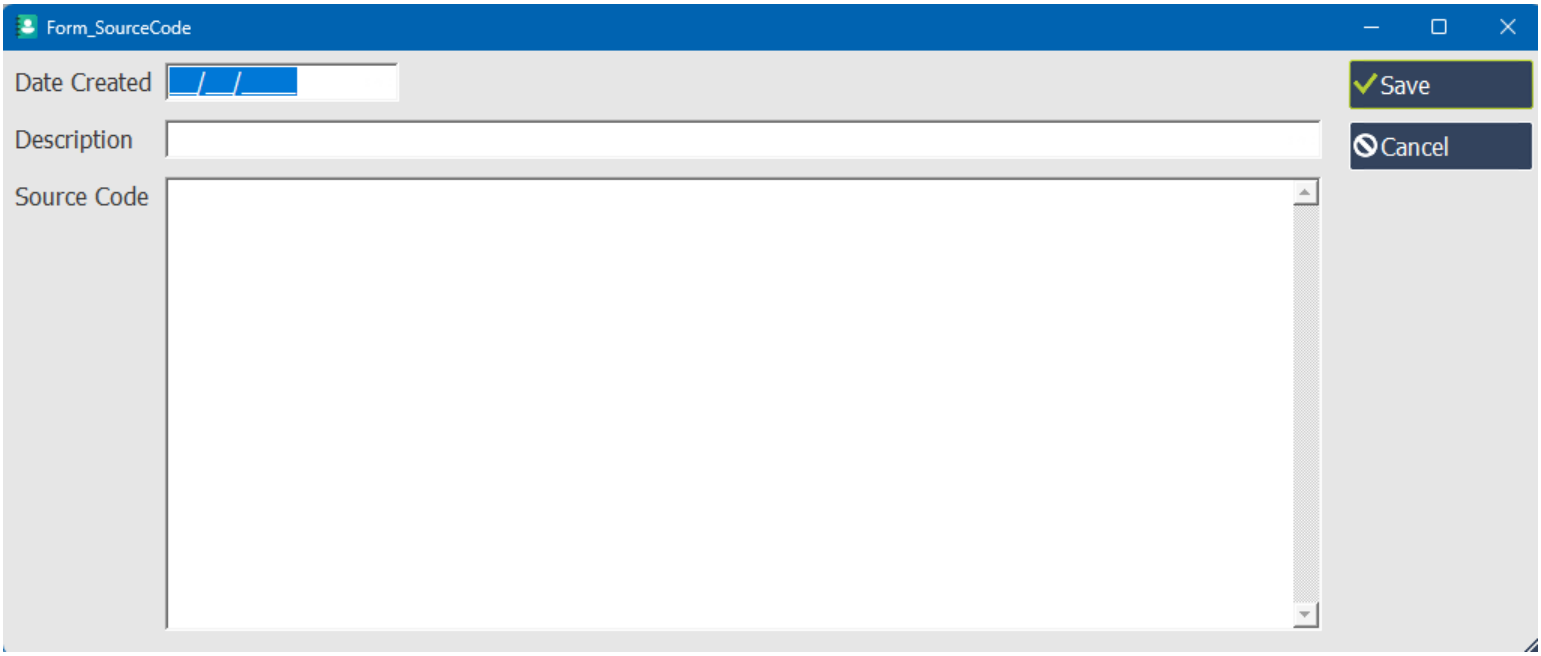
Close
Closes the Source Code Repository browse window and returns to the LicenseTrak main menu.

Although multiple Runtime Library versions may exist within the repository, only the version selected by the user is exported. This allows developers to maintain compatibility with previously released applications while retaining access to newer Runtime Library revisions as they become available.

Creating a New Runtime Library

The **New** button is used to create a new Runtime Library record within the Source Code Repository. This feature is primarily intended for Software by Daughtry to maintain future versions of the LicenseTrak Runtime Library, although developers may also use it to archive customized Runtime Library variations for their own projects.

Selecting **New** displays the Runtime Library maintenance window.



Each Runtime Library record contains the following information:

Date Created
Specifies the date the Runtime Library was added to the Source Code Repository. This value provides a chronological history of Runtime Library releases. A popup calendar can be accessed by clicking the far right interior of the entry field.

Description
A brief description identifying the Runtime Library. The description should clearly distinguish one Runtime Library version from another. Examples include version numbers, supported WinDev releases, or a summary of significant enhancements.

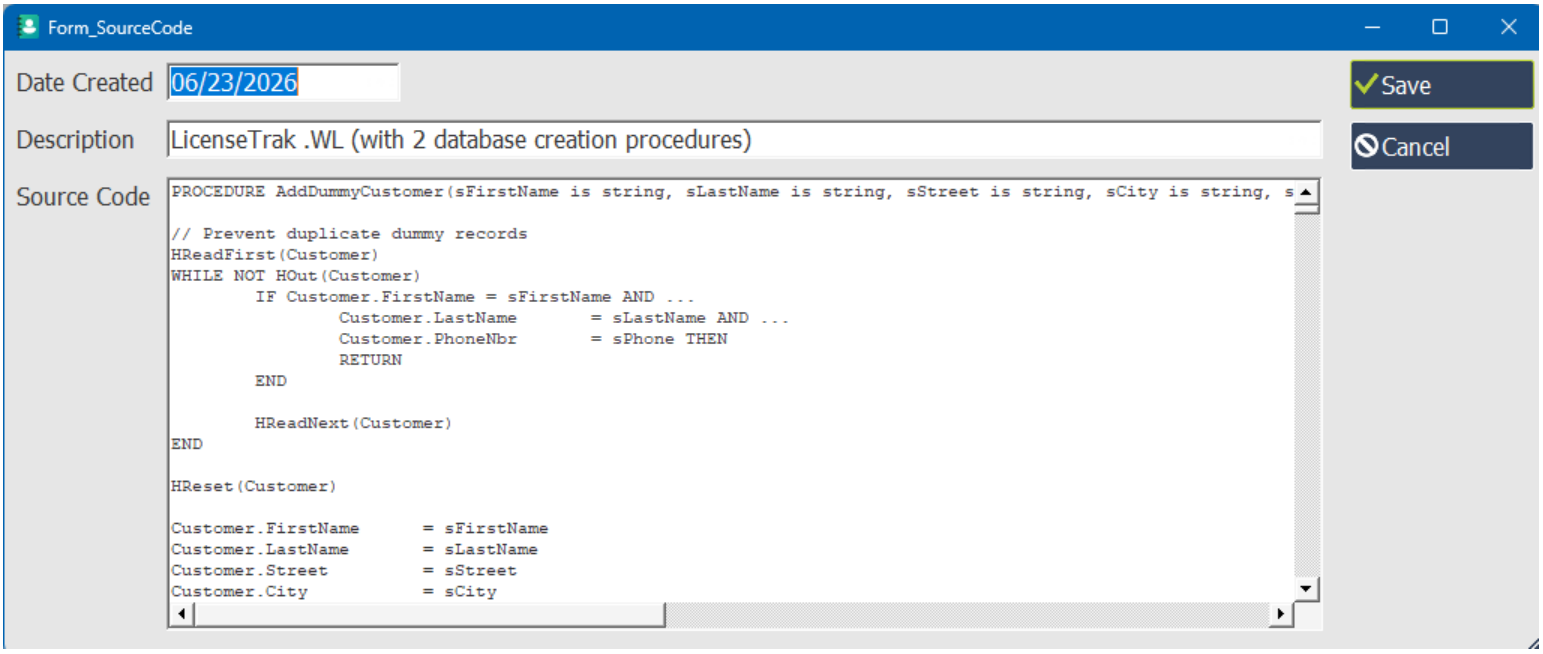
Source Code
Contains the complete contents of the LicenseTrak Runtime Library (.WL) source file. The source code is stored within an encrypted LicenseTrak database, providing a centralized repository for one or more Runtime Library versions. This field is Unicode compatible.

After entering the desired information, click **Save** to store the new Runtime Library within the Source Code Repository or **Cancel** to abandon the operation without saving any changes.

As additional LicenseTrak releases become available, new Runtime Library versions may be added to the repository without affecting previously stored versions. This approach provides a convenient historical archive while ensuring that older Runtime Libraries remain available for maintaining existing customer applications.

Editing Existing Runtime Libraries

Selecting the **Edit** button opens the selected Runtime Library record, allowing its descriptive information and embedded source code to be viewed or modified. This function is primarily intended for maintaining existing Runtime Library versions, correcting documentation, or incorporating enhancements into future LicenseTrak releases.



The Runtime Library maintenance window displays the following fields:

Date Created
Displays the date the Runtime Library record was originally created. This value provides a historical reference for the Runtime Library and is normally not modified after the record has been created.

Description
Contains a concise description of the Runtime Library. This field should uniquely identify the Runtime Library version and may include information such as the supported WinDev version, release number, or a summary of significant enhancements.

Source Code
Contains the complete Unicode-compatible LicenseTrak Runtime Library (.WL) source code stored within the encrypted Source Code Repository. The source code may be viewed, modified, or replaced as new LicenseTrak

features are developed.

After making the desired changes, click **Save** to permanently update the Runtime Library within the Source Code Repository. Selecting **Cancel** closes the maintenance window without saving any modifications.

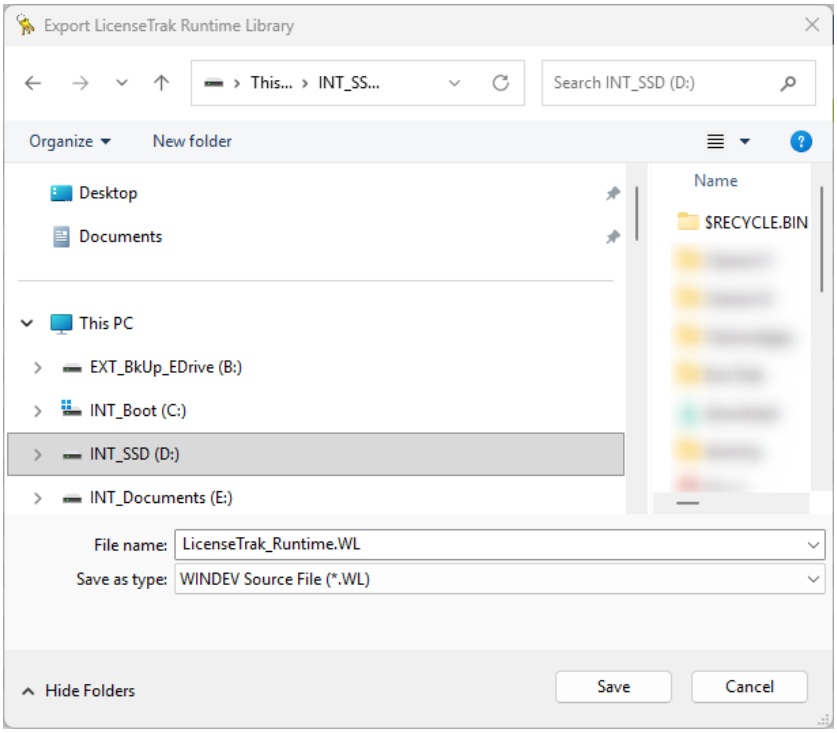
Because the Runtime Library represents the core source code distributed to registered LicenseTrak customers, it is recommended that significant modifications be accompanied by a new Runtime Library record rather than overwriting an existing production version. Maintaining separate Runtime Library versions provides a historical archive of released code and simplifies long-term maintenance of customer applications that depend upon earlier LicenseTrak releases.

As a general practice, production Runtime Libraries should remain unchanged once they have been distributed to customers. Creating a new Runtime Library record for each significant release provides a reliable version history while ensuring that previously released Runtime Libraries remain available for future maintenance and support.

Exporting a Runtime Library

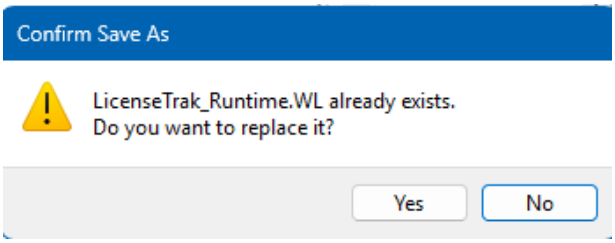
The **Export** button is used to write the selected Runtime Library from the Source Code Repository to a standard WinDev (.WL) source file. Once exported, the Runtime Library may be imported into any compatible WinDev application using the procedures described in Chapter 10.

To export a Runtime Library, first select the desired Runtime Library record from the Source Code Repository browse window, and then click the **Export** button. A standard Windows **Save As** dialog is then displayed, allowing the destination folder and filename to be specified:

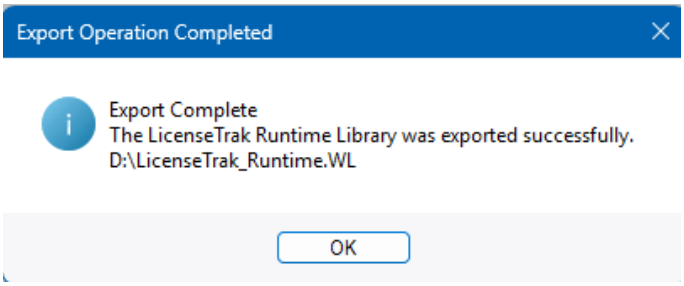


Although any valid filename may be entered, it is recommended that the default filename be retained whenever practical. This simplifies future Runtime Library identification and minimizes confusion when multiple Runtime Library versions are maintained.

After clicking **Save**, LicenseTrak writes the complete Runtime Library source code contained within the selected repository record to the specified .WL file. If the file already exists within the destination folder, a warning message is displayed for overwrite confirmation:



Upon successful completion, a confirmation message is displayed indicating that the Runtime Library has been exported successfully.



The exported Runtime Library is a standard WinDev source code file and may be imported into any compatible WinDev project using WinDev's **Import WLanguage Source Code** feature.

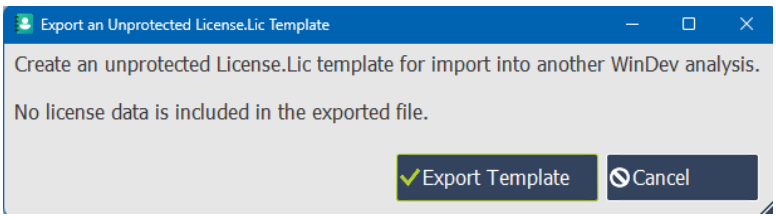
For security purposes, the Runtime Library is stored within LicenseTrak's encrypted Source Code Repository and is only exported when explicitly requested by a registered LicenseTrak user. This approach protects the Runtime Library from casual access while providing developers with a convenient mechanism for obtaining the version required by their application.

If multiple Runtime Library versions exist within the repository, ensure that the appropriate version is selected before exporting. This allows different WinDev applications to continue using the Runtime Library version with which they were originally developed while newer applications may benefit from subsequent LicenseTrak enhancements.

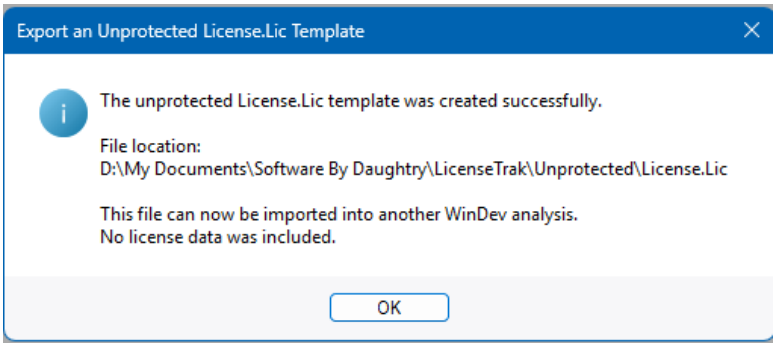
The exported .WL file can now be imported into your WinDev application's Global Procedures.

Export Unprotected License.LIC

This menu option generates an empty, unencrypted License.LIC file that can be imported into your WinDev analysis. A popup dialogue is displayed:



After clicking **Export Template**, an empty HyperFile database is created in the **Unprotected** subfolder beneath the LicenseTrak installation folder. A confirmation dialog displays the location of the generated file:



Note: The exported **License.LIC** file contains only the file structure required by the Runtime Library. No customer information, licensing data, or Developer Seed values are included.

Chapter 5 - Managing Applications

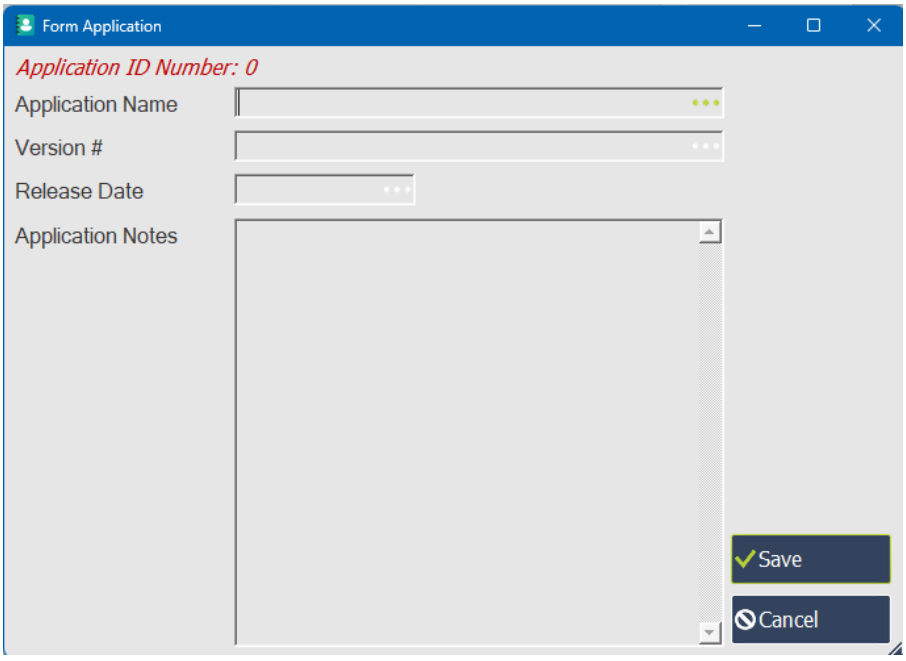
Creating Applications

Before LicenseTrak can generate licenses, an Application record must exist within the Application Library.

Each Application record represents a specific software application that will be protected by LicenseTrak.

To create a new Application record:

1. Select File, then Browse Application Picklist.
2. Click the New button.
3. Complete the Application Form.
4. Click Save.



The Application Form contains the following fields:

- Application Name
- Version Number
- Release Date
- Application Notes

The Application Name and Version Number uniquely identify the software product that will be associated with generated License.LIC files.

The Release Date field records the application's release date and may be useful for version tracking and historical reference purposes.

The Application Notes field may be used to document version history, release information, feature changes, licensing policies, or other developer-specific information.

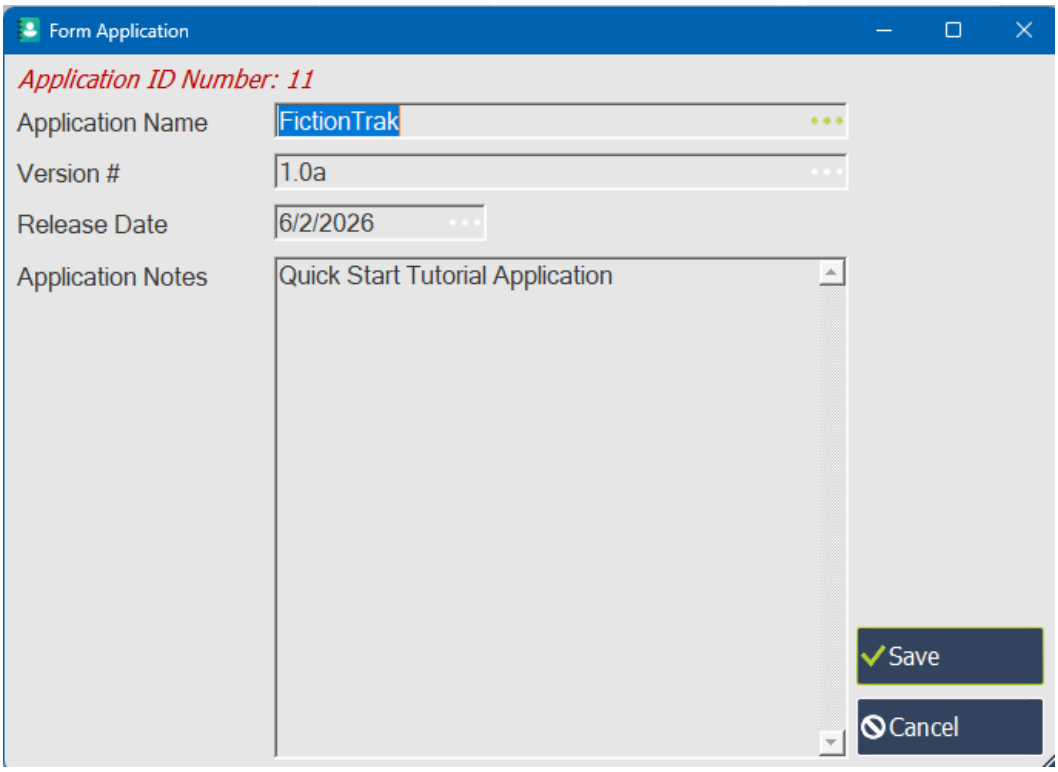
Once saved, the Application record becomes available for license generation activities.

Editing Applications

Application records may be modified at any time through the Application Library.

To edit an Application record:

1. Open the Browse Application Picklist window.
2. Select the desired Application record.
3. Click the Edit button.
4. Modify the desired information.
5. Click Save.



Typical reasons for editing an Application record include:

- Correcting typographical errors.
- Updating release dates.
- Revising Application Notes.
- Documenting version-specific information.

Changes made to an Application record are immediately available throughout LicenseTrak.

Developers should exercise caution when modifying Application Name and Version Number values after licenses have already been generated. These values are used during license generation and validation activities.

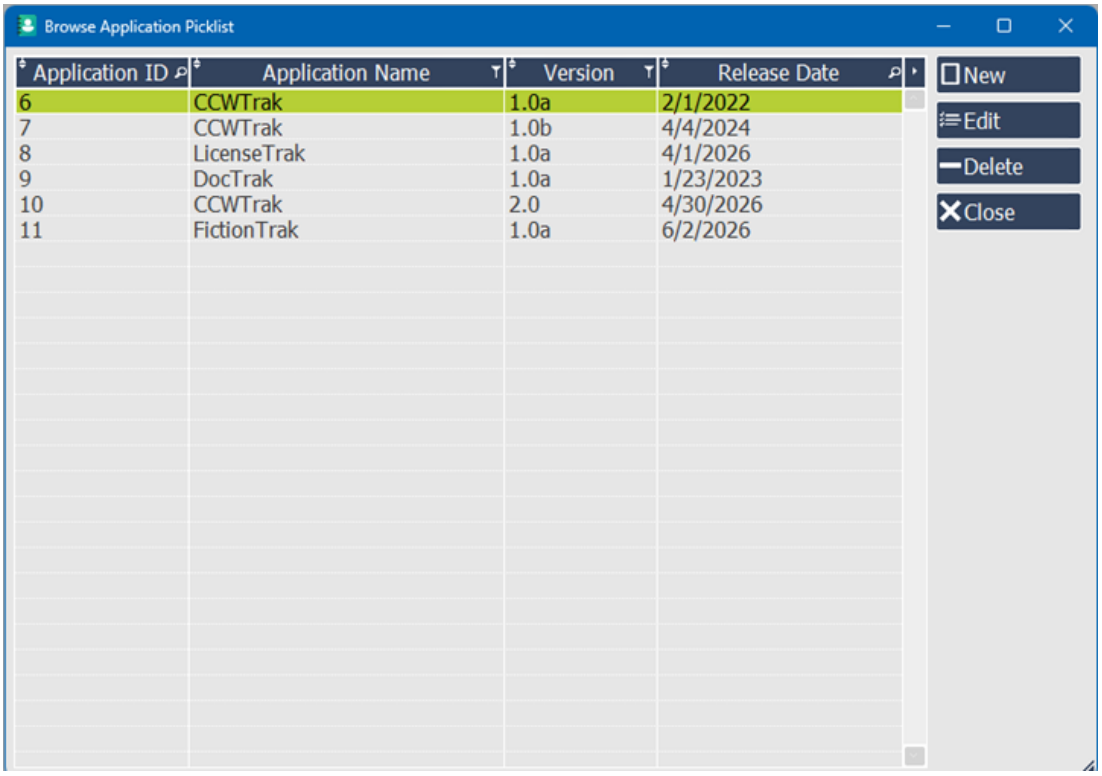
Deleting Applications

Application records may be removed from the Application Library when they are no longer required.

To delete an Application record:

1. Open the Browse Application Picklist window.
2. Select the desired Application record.
3. Click the Delete button.
4. Confirm the deletion when prompted.

The selected Application record is permanently removed from the Application Library.



Before deleting an Application record, developers should carefully consider whether licenses have previously been generated for the application.

Although a software version may no longer be actively sold or supported, customers may continue using the application for many years. Historical Application records can simplify license replacement, customer support, version tracking, and other administrative activities.

For this reason, many developers choose to retain Application records indefinitely whenever practical.

Application Versioning

Most software applications evolve over time through bug fixes, feature enhancements, maintenance releases, and major version upgrades.

LicenseTrak provides flexibility in how developers manage application versions.

A common approach is to create a separate Application record for each released version.

For **example:**

- FictionTrak 1.0a
- FictionTrak 1.0b
- FictionTrak 2.0

Each version is maintained as a separate Application record within the Application Library.

This approach provides several advantages:

- Historical version tracking.
- Simplified license replacement.
- Clear identification of customer purchases.
- Preservation of legacy licensing information.
- Improved customer support capabilities.

Maintaining separate Application records allows developers to determine exactly which version was licensed to a customer and recreate replacement License.LIC files when necessary.

Although LicenseTrak permits Application records to be modified, many developers prefer creating new Application records for major releases rather than overwriting existing version information.

This practice preserves licensing history and provides a permanent record of the application's evolution over time.

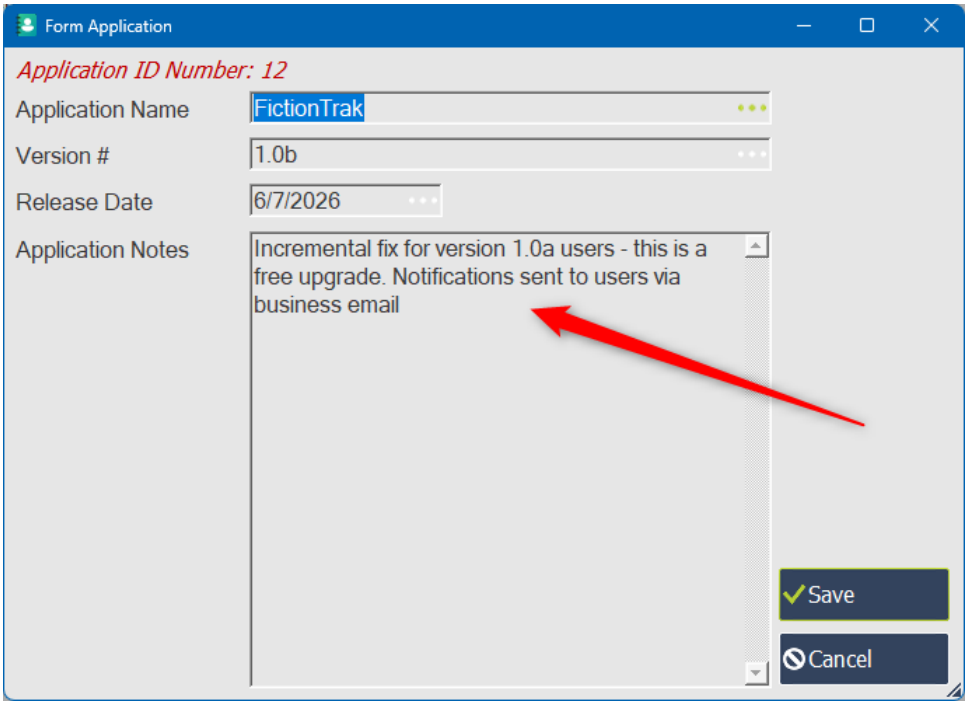
Application Notes

The Application Notes field provides a convenient location for storing developer-specific information related to an application.

Although the field is optional, many developers find it useful for recording information that may be difficult to remember months or years later.

Typical uses include:

- Version history information.
- Major feature additions.
- Licensing policy changes.
- Upgrade eligibility information.
- Release-specific notes.
- Internal development comments.



For **example**, a developer might record:

"Version 2.0 introduced multi-user support."

or

"Customers licensed under Version 1.x are eligible for a free upgrade to Version 2.0."

The Application Notes field may also be used to document application retirement dates, support policies, or special licensing arrangements.

Because software products often remain in use for many years, maintaining useful notes can simplify customer support and administrative activities long after the original release date.

Best Practices

The following recommendations have proven useful when managing applications within LicenseTrak:

Maintain Separate Application Records

Create separate Application records for major releases and version changes rather than overwriting historical information.

Preserve Historical Records

Avoid deleting Application records whenever practical. Historical records simplify customer support, license replacement, and version tracking activities.

Use Meaningful Version Numbers

Adopt a consistent version numbering strategy to simplify customer communication and license management.

Document Important Changes

Use the Application Notes field to record information that may be useful months or years later.

Plan For Long-Term Support

Customers often continue using software versions long after newer releases become available. Maintaining complete application histories can significantly reduce future support efforts.

Think Like Future You

When making administrative decisions, consider how those decisions may affect customer support, license replacement, and historical record keeping years from now.

Future You will usually appreciate having more information rather than less.

Chapter 6 - Managing Customers

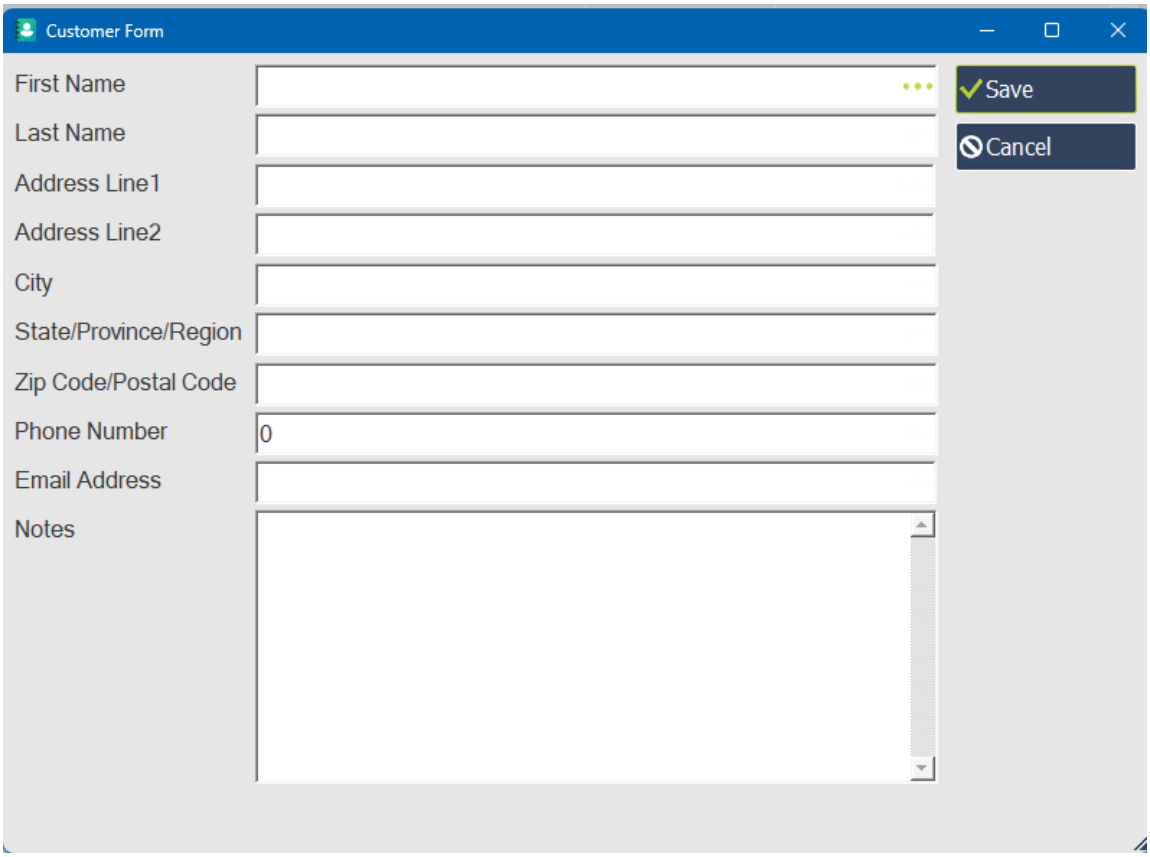
Creating Customers

Customer records are used to identify the individuals and organizations that receive software licenses generated by LicenseTrak.

Before a License.LIC file can be generated, a Customer record must exist within the Customer Library.

To create a new Customer record:

1. Select File, then Browse Customers.
2. Click the New button located within the Customers section.
3. Complete the Customer Form.
4. Click Save.



The Customer Form contains the following fields:

- First Name
- Last Name
- Address Line 1
- Address Line 2
- City
- State/Province/Region
- Zip Code/Postal Code

- Phone Number
- Email Address
- Notes

The information entered within the Customer Form is used for administrative purposes and may also be incorporated into generated License.LIC files depending upon the licensing model implemented by the software developer.

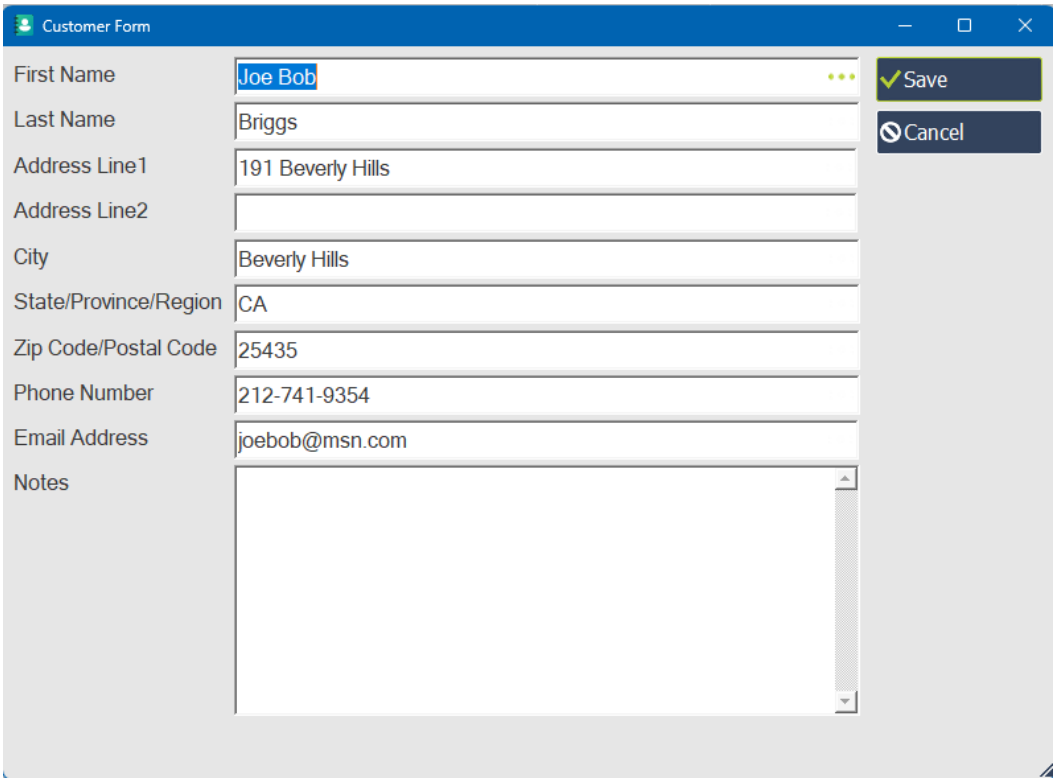
Once saved, the Customer record becomes available for license generation and support tracking activities.

Editing Customers

Customer information may be modified at any time through the Customer Library.

To edit a Customer record:

1. Open the Browse Customers window.
2. Select the desired Customer record.
3. Click the Edit button.
4. Modify the desired information.
5. Click Save.



Typical reasons for editing a Customer record include:

- Address changes.
- Email address updates.
- Telephone number changes.
- Name corrections.
- Additional customer notes.

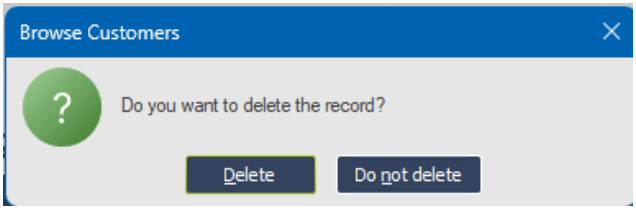
The updated information is immediately available throughout LicenseTrak and may be used when generating future License.LIC files for the customer.

Deleting Customers

Customer records may be removed from the Customer Library when they are no longer required.

To delete a Customer record:

1. Open the Browse Customers window.
2. Select the desired Customer record.
3. Click the Delete button located within the Customers section.
4. Confirm the deletion when prompted.



If the selected Customer record does not contain any associated License or Support Call records, the customer is permanently removed from the database.

However, if related License or Support Call records exist, LicenseTrak prevents the deletion from occurring and displays an informational message.

This behavior protects the integrity of customer, licensing, and support history information stored within LicenseTrak.

Because customers may require replacement licenses, upgrades, or support assistance years after their original purchase, developers should carefully consider whether a Customer record should be permanently removed.

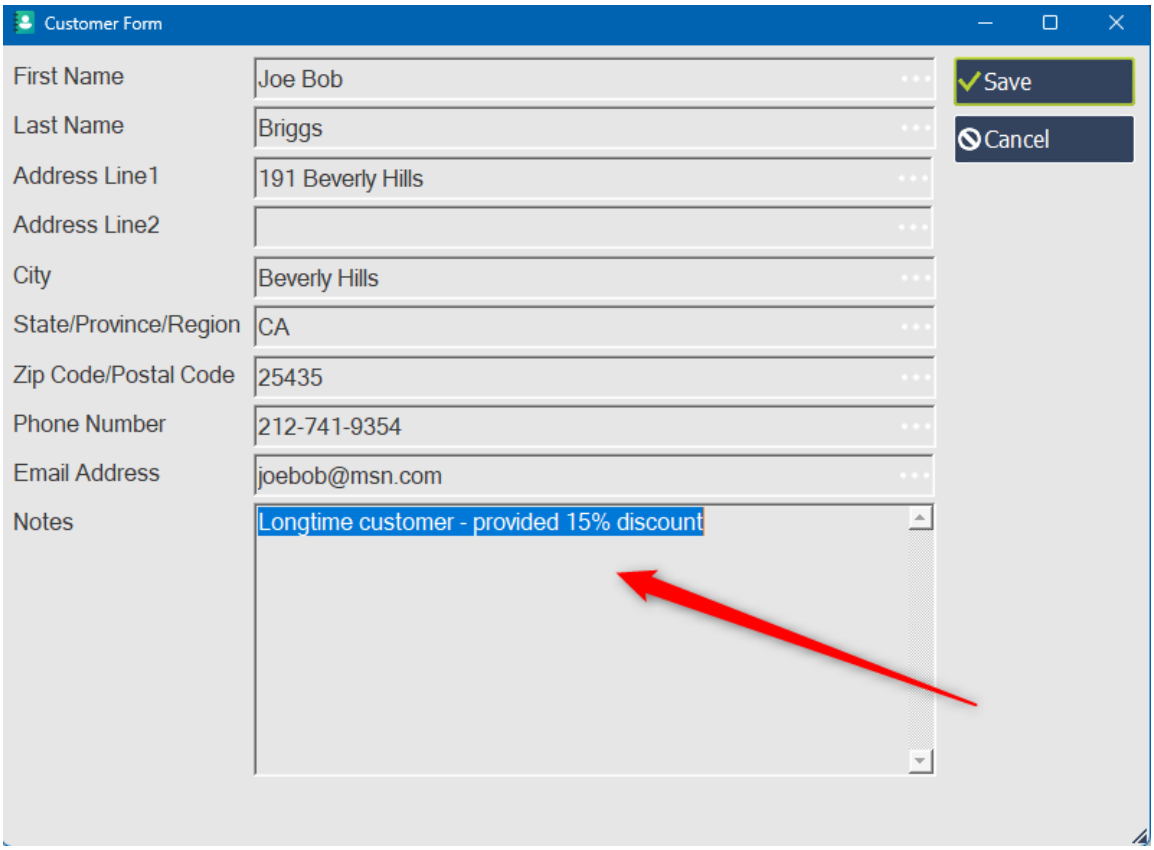
Customer Notes

The Notes field provides a convenient location for storing customer-specific information that may not fit within the standard Customer Form fields.

Although optional, many developers find the Notes field valuable when maintaining long-term customer relationships.

Typical uses include:

- Special licensing arrangements.
- Upgrade eligibility information.
- Customer preferences.
- Historical purchase information.
- Support-related observations.
- Administrative reminders.



For **example**, a developer might record:

"Customer eligible for free upgrade to Version 2.0."

or

"Customer purchased a five-seat license agreement."

or

"Customer requested future communication by email only."

The Notes field can serve as a valuable reference when interacting with customers months or years after the original software purchase.

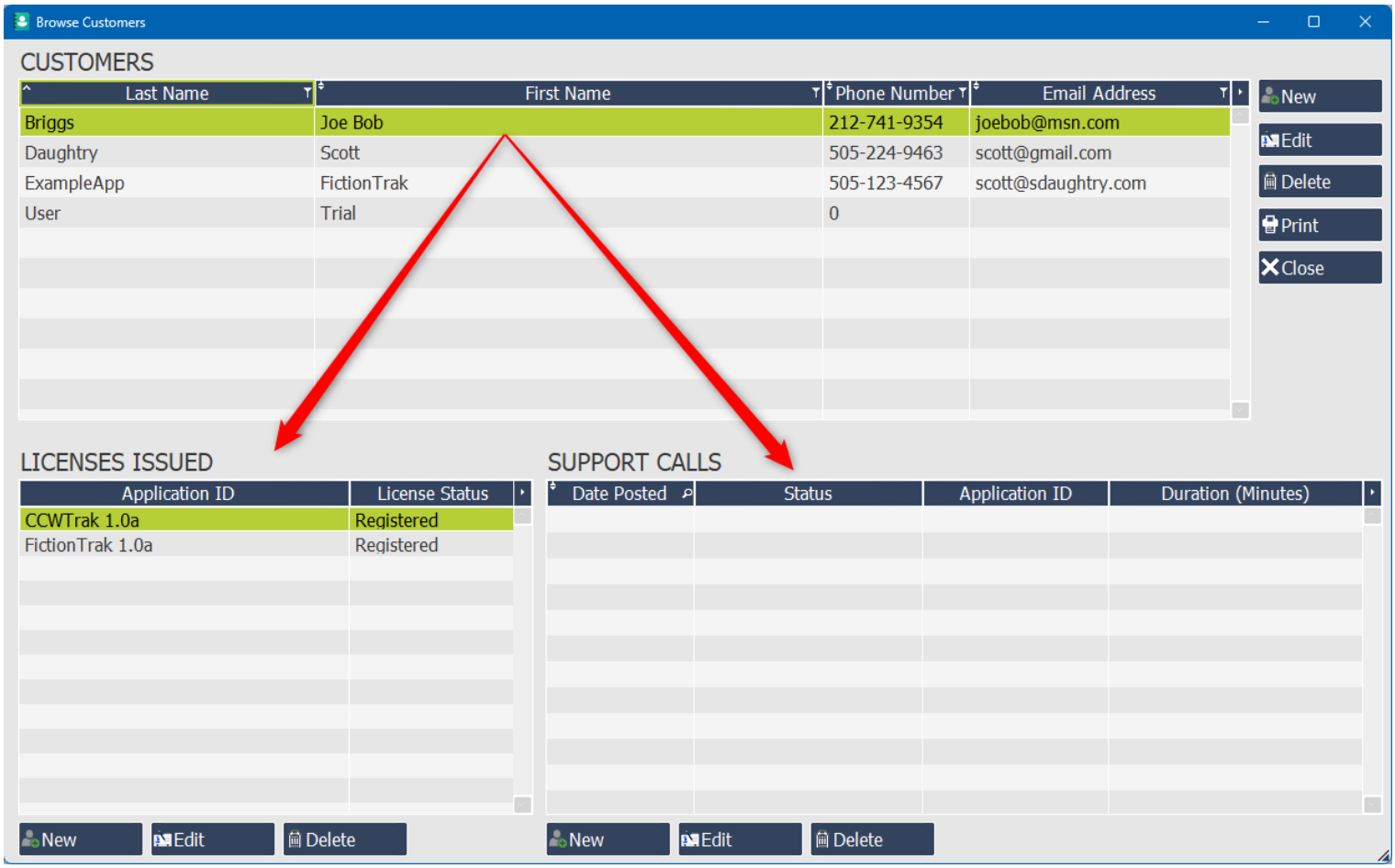
Maintaining useful notes often reduces support time and improves customer service activities.

Customer Relationships

The Customer Library serves as the central relationship point within LicenseTrak.

Each Customer record may be associated with:

- Multiple License records
- Multiple Support Call records



When a Customer record is selected, LicenseTrak automatically displays all related License and Support Call records associated with that customer.

This design provides a consolidated view of a customer's licensing history and support history from a single administrative screen.

For **example**, a developer may quickly determine:

- Which applications a customer has licensed.
- Which license types have been issued.
- Whether support calls have been reported.
- Historical licensing and support activity.

This relationship-based design simplifies customer support, license replacement, troubleshooting, and administrative record keeping activities.

Maintaining complete customer histories often proves valuable long after the original software purchase has occurred.

Best Practices

The following recommendations have proven useful when managing customer information within LicenseTrak.

Preserve Customer Records

Avoid deleting Customer records whenever practical. Customers may require replacement licenses, upgrades, support assistance, or historical information years after their original software purchase.

Maintain Accurate Contact Information

Keep customer email addresses, telephone numbers, and mailing information current whenever possible. Accurate contact information simplifies software support and license distribution activities.

Document Important Information

Use the Notes field to record information that may be useful during future customer interactions. Special licensing arrangements, upgrade eligibility, and customer preferences are common examples.

Record Support Activity

Whenever practical, document customer support interactions using the Support Calls feature. Maintaining a support history can simplify troubleshooting and provide valuable historical context.

Think Long-Term

Many customers continue using software versions long after newer releases become available. Maintaining complete customer records helps simplify future support and licensing activities.

Maintain Historical Information

Historical customer information often becomes more valuable over time. Information that appears unimportant today may prove extremely useful when recreating licenses, researching purchases, or resolving support issues years later.

Treat Customer Records As Business Assets

Customer records represent relationships that may span many years. Maintaining complete and accurate customer information helps protect those relationships and improves the overall customer experience.

Think Like Future You

Before deleting or significantly modifying customer information, consider whether the change could complicate future support, licensing, or administrative activities.

Future You will usually appreciate having access to more information rather than less.

Chapter 7 - Generating Licenses

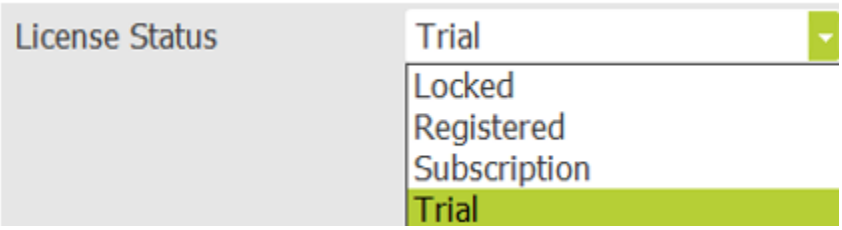
License Status Values

LicenseTrak uses the License Status field to define the operational state of a generated License.LIC file.

The selected License Status value is stored within the generated license and may be evaluated by the LicenseTrak Runtime Library during application execution.

The available License Status values are:

- Locked
- Registered
- Subscription
- Trial



Each License Status value serves a specific **purpose** within the licensing lifecycle.

Trial

Trial licenses permit software evaluation prior to purchase. Developers may implement restrictions based upon trial days, trial runs, or other licensing policies.

Registered

Registered licenses indicate that a customer has purchased the software and is authorized to use the application according to the developer's licensing policies.

Locked

Locked licenses prevent normal application operation. Locked licenses may be used when access to an application should be restricted or disabled.

The selected License Status value is incorporated into the generated License.LIC file and becomes available to the LicenseTrak Runtime Library through the LT_IsTrial(), LT_IsRegistered(), and LT_IsLocked() validation functions.

Subscription

Subscription-based licenses permit the company to enforce a cut-off date for its LicenseTrak-protected software when installed on a customer’s computer.

Trial Licenses

Trial licenses are commonly used to distribute demonstration, evaluation, or shareware versions of software applications.

A Trial license permits prospective customers to evaluate an application before making a purchasing decision.

LicenseTrak supports several trial models:

- Days-Based Trials
- Run-Based Trials
- Hybrid Trials

Developers may select the model most appropriate for their application and target audience.

For **example:**

- A 30-day evaluation period.
- A 50-execution evaluation period.
- A 30-day evaluation period limited to 50 executions.

Each trial model is discussed in detail within the following sections.

Days-Based Trials

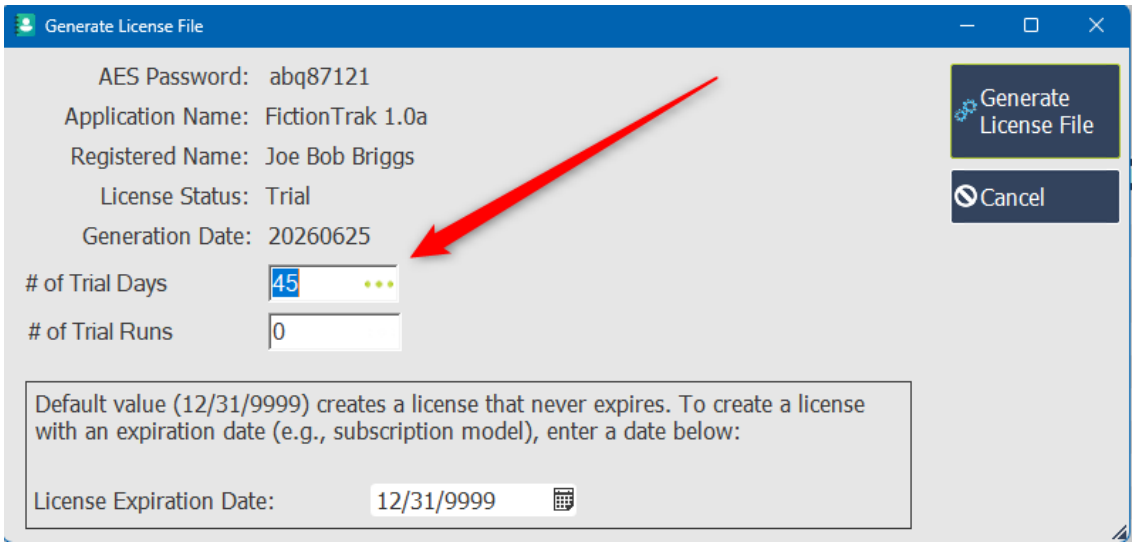
Days-Based Trials are the most common software evaluation model.

A Days-Based Trial permits application usage for a specified number of calendar days beginning with the first successful execution of the application.

For **example:**

- 15-day trial
- 30-day trial
- 45-day trial
- 60-day trial

Days-Based Trials are configured through the Trial Days field within the Generate License File window.



When a Days-Based Trial license is generated, the LicenseTrak Runtime Library records the application's first execution date and calculates the remaining trial period during subsequent application launches.

Days-Based Trials are often preferred when developers wish to provide unrestricted access to application functionality for a limited period of time.

Typical uses include:

- Shareware applications.
- Demonstration software.
- Evaluation versions.
- Beta testing programs.

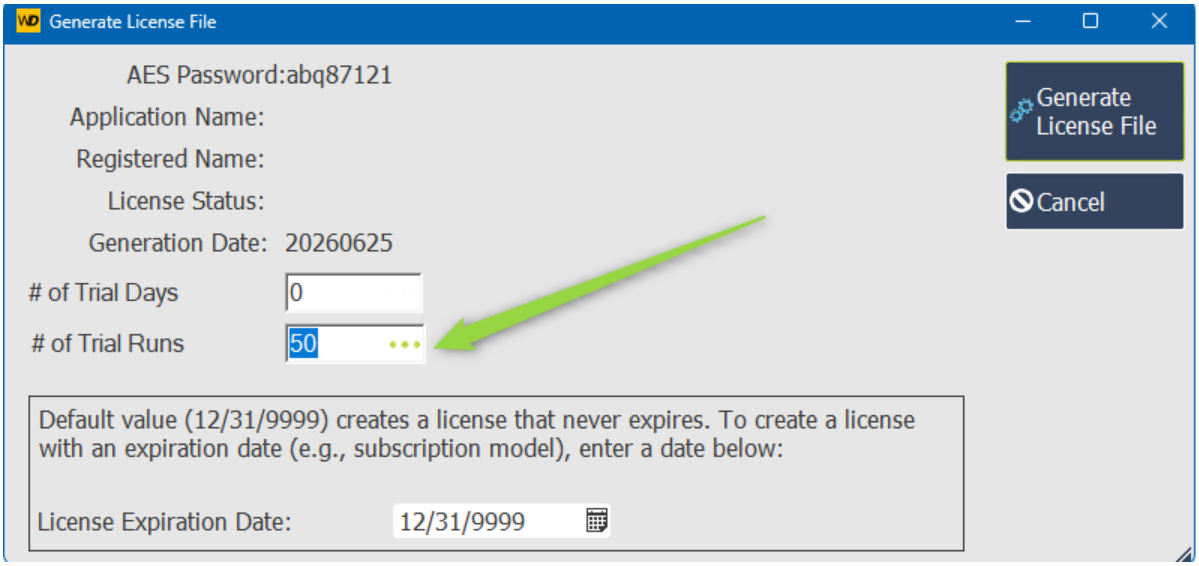
Run-Based Trials

Run-Based Trials restrict application usage based upon the number of application executions rather than the number of elapsed calendar days.

For **example:**

- 10 executions
- 25 executions
- 50 executions
- 100 executions

Run-Based Trials are configured through the Trial Runs field within the Generate License File window.



Each successful application launch increments the Execution Count value stored within the License.LIC file.

Once the maximum number of permitted executions has been reached, the LicenseTrak Runtime Library may prevent further application operation depending upon the developer's implementation.

Run-Based Trials are often useful when application usage occurs infrequently or when developers wish to provide a specific number of evaluation opportunities regardless of elapsed time.

Typical uses include:

- Specialized business applications.
- Engineering software.
- Utility programs.
- Applications used intermittently by customers.

Hybrid Trials

Hybrid Trials combine both Days-Based and Run-Based restrictions within a single License.LIC file.

For **example:**

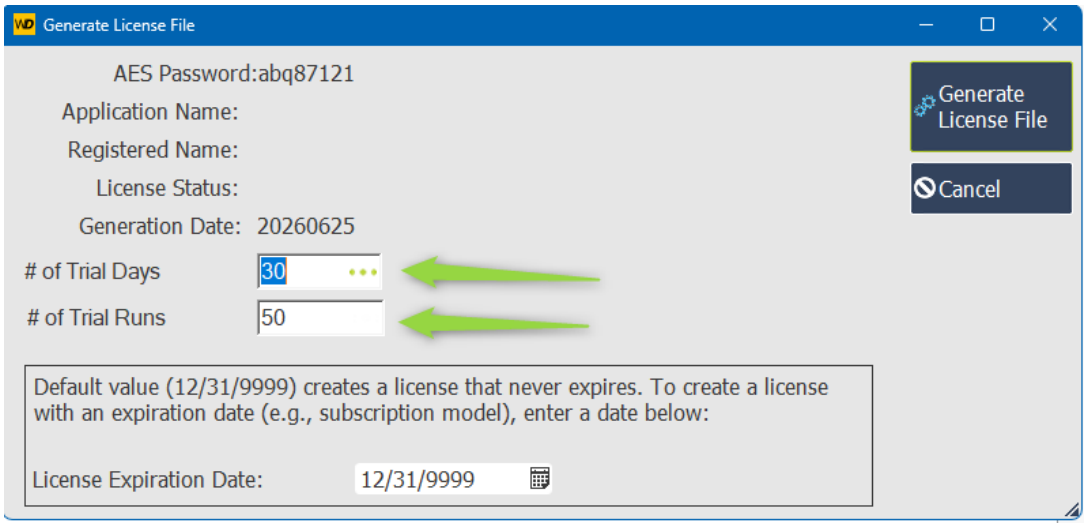
- 30 days and 50 executions
- 45 days and 100 executions
- 60 days and 25 executions

A Hybrid Trial expires when either restriction is exceeded.

For **example**, a 30-day/50-run trial becomes invalid if:

- 30 calendar days have elapsed **OR**
- 50 executions have occurred

whichever occurs first.



Hybrid Trials provide the highest level of evaluation control and are often used when developers wish to limit both the duration and frequency of application usage.

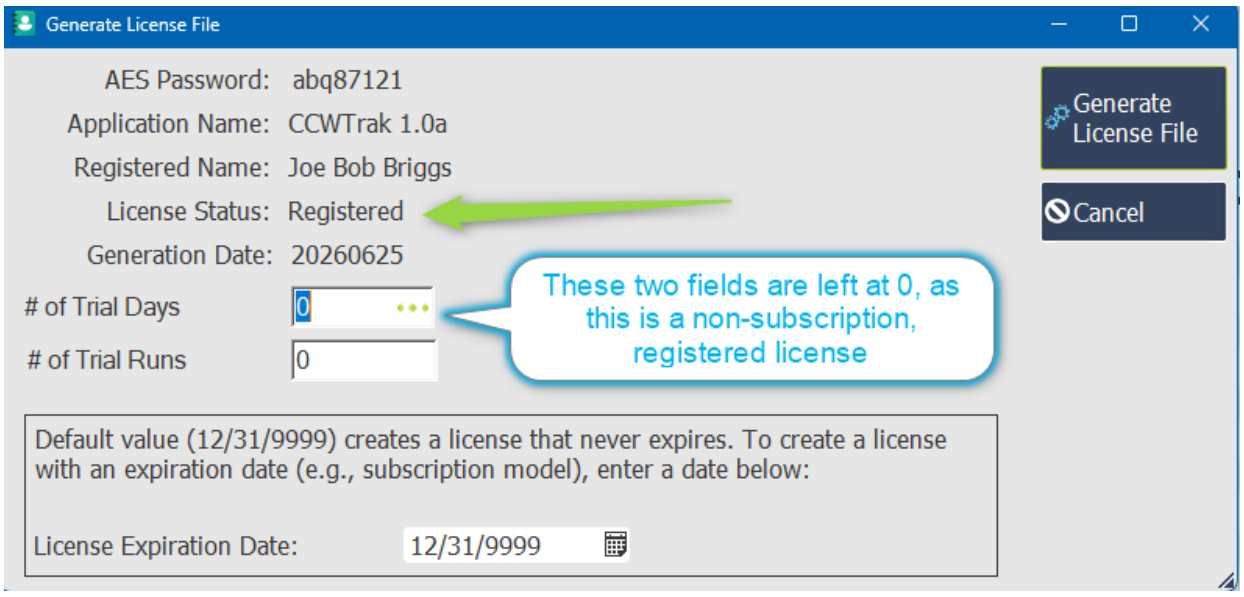
This licensing model can help ensure that evaluation copies are used for legitimate product evaluation purposes while still providing prospective customers with sufficient opportunity to assess the software.

Registered Licenses

Registered Licenses are issued to customers who have purchased the software application.

A Registered License indicates that the customer is authorized to use the application according to the developer's licensing policies.

Unlike Trial Licenses, Registered Licenses typically do not contain Trial Day or Trial Run restrictions.



Registered Licenses are commonly generated after:

- Software purchases.

- Upgrade purchases.
- Maintenance renewals.
- License replacements.

When a Registered License is detected, the LicenseTrak Runtime Library may permit unrestricted application operation according to the developer's implementation.

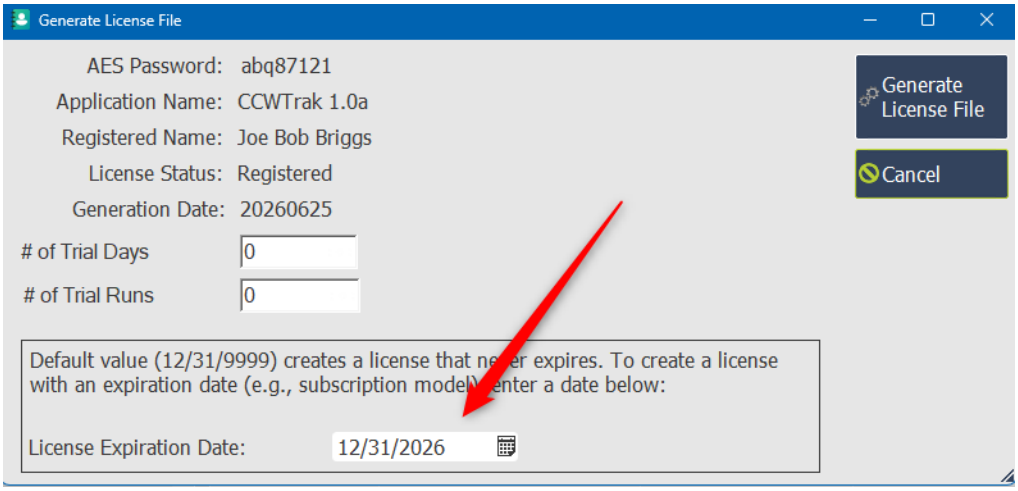
Registered Licenses represent the most common licensing model used by commercial software publishers.

Subscription Licenses

Although LicenseTrak does not contain a dedicated Subscription License status, subscription-based licensing models may be implemented using Registered Licenses in combination with License Expiration Dates.

For **example:**

- 30-day subscription
- 90-day subscription
- Annual subscription
- Multi-year subscription



When a License Expiration Date is specified, the LicenseTrak Runtime Library may evaluate the expiration date and determine whether the license remains valid.

This approach allows developers to implement recurring licensing models without requiring Internet activation servers, cloud-based licensing services, or online account validation.

Subscription Licenses are commonly used for:

- SaaS-style desktop applications.
- Annual maintenance programs.
- Educational licensing.
- Membership-based applications.

The specific enforcement behavior is determined by the developer's implementation of the Runtime Library

validation functions.

License Violations and Tamper Detection

LicenseTrak provides mechanisms that assist developers in detecting license-related violations and potential tampering activities.

Rather than enforcing a fixed response, LicenseTrak records violation information and allows the protected application to determine the appropriate course of action.

Violation information is maintained within the License.LIC file and may be examined through the LicenseTrak Runtime Library.

Examples of activities that may result in a recorded violation include:

- Invalid license validation attempts.
- License file modification attempts.
- Application identity mismatches.
- Other developer-defined validation failures.

The LicenseTrak Runtime Library provides procedures that allow applications to record and evaluate violation information during program execution.

This design provides maximum flexibility because enforcement decisions remain under the control of the software developer rather than the licensing system itself.

Developer-Defined Enforcement

LicenseTrak intentionally separates violation detection from enforcement actions.

When a licensing violation is detected, the protected application may choose how to respond based upon the developer's business requirements.

Possible responses include:

- Displaying a warning message.
- Recording the violation for future review.
- Restricting selected application features.
- Preventing application startup.
- Requiring customer contact with the software publisher.
- Other developer-defined actions.

For **example**, an application might permit several validation failures before restricting functionality, while another application may terminate immediately after the first detected violation.

Because enforcement policies vary widely between software products, LicenseTrak leaves these decisions to the developer.

The Runtime Library provides the information required to make enforcement decisions without imposing a specific licensing strategy.

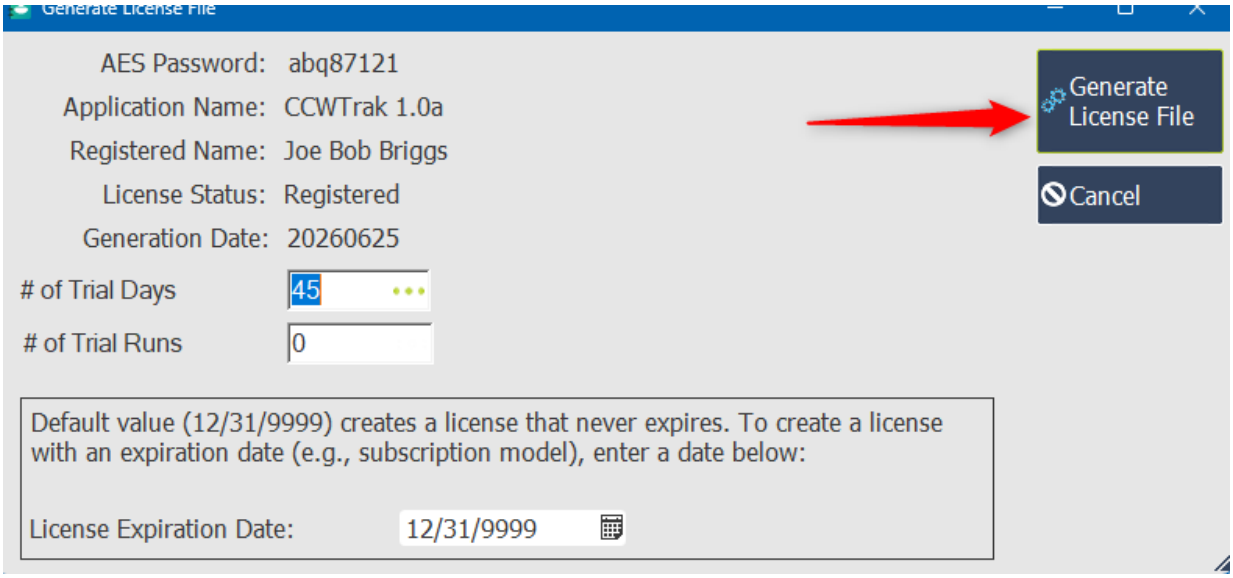
License File Generation

LicenseTrak generates License.LIC files using information stored within the Application Library, Customer Library, License records, and Developer Seed configuration.

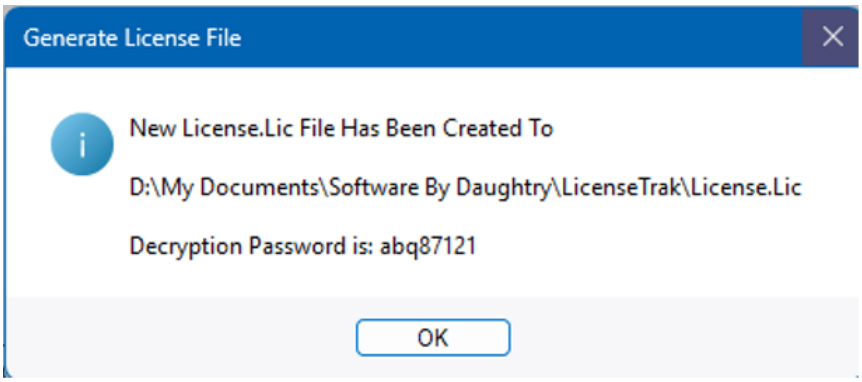
The generated License.LIC file contains the information required by the LicenseTrak Runtime Library to determine licensing status and application authorization.

During generation, LicenseTrak incorporates information including:

- Developer Seed
- Application Name
- Version Number
- Registered Customer Name
- License Status
- Trial Restrictions
- License Expiration Information
- Validation Information



Clicking the **Generate License File** button results in a new License.LIC file to be generated in the LicenceTrak folder and display a popup message after the file is created (note: a previously generated License.LIC will will be overwritten if it exists in the LicenseTrak folder):



The resulting License.LIC file may be distributed with software installation packages, delivered directly to customers, or incorporated into other software distribution workflows.

Because all required licensing information is contained within the License.LIC file, no Internet activation, online account validation, cloud-based licensing services, or external license servers are required.

License Regeneration

One of the most valuable features provided by LicenseTrak is the ability to regenerate replacement License.LIC files.

Customers occasionally lose license files due to:

- Hard drive failures.
- Computer replacements.
- Operating system reinstalls.
- Accidental file deletion.
- Data migration activities.

When this occurs, the software publisher may simply locate the customer's existing License record and generate a replacement License.LIC file.

Because LicenseTrak maintains historical Application, Customer, and License information, replacement licenses may be generated years after the original purchase.

This capability significantly reduces customer support effort and eliminates the need to recreate licensing information from memory.

For this reason, many developers choose to preserve historical Application, Customer, and License records indefinitely whenever practical.

Maintaining complete historical information simplifies future license replacement activities and improves long-term customer support.

Detecting License Abuse

No software licensing system can guarantee absolute protection against a sufficiently skilled and determined attacker.

LicenseTrak is designed to assist developers in identifying suspicious licensing activity and potential tampering

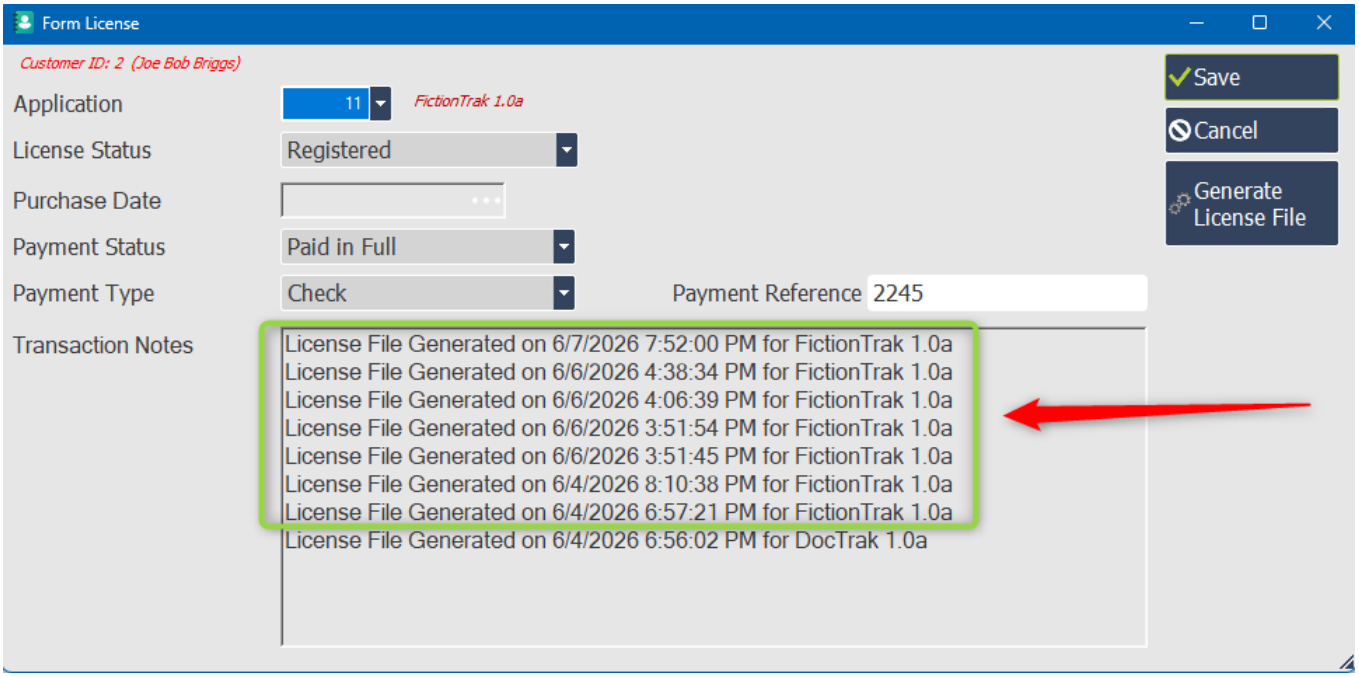
attempts while providing flexible mechanisms for responding to those events.

LicenseTrak records violation information within the License.LIC file and makes that information available through the Runtime Library.

Potential indicators of suspicious activity may include:

- Repeated license validation failures.
- Application identity mismatches.
- License file modification attempts.
- Other developer-defined validation exceptions.

The LicenseTrak Runtime Library allows developers to examine violation information and determine whether additional action should be taken.



Because each software product has unique requirements, LicenseTrak does not impose a predefined enforcement policy.

Instead, developers may evaluate violation information and determine the most appropriate response for their applications.

Reasonable Expectations

Software licensing should be viewed as a business control rather than an impenetrable security barrier.

Most software piracy and unauthorized usage result from convenience, misunderstanding, or opportunity rather than highly sophisticated technical attacks.

LicenseTrak is designed to discourage unauthorized use, identify suspicious activity, and provide developers with information that can be used to make informed licensing decisions.

Developers should establish licensing policies that balance protection, customer convenience, and administrative effort.

In many cases, a practical licensing solution that serves legitimate customers well is more valuable than an overly restrictive system that creates support challenges for paying customers.

LicenseTrak provides the tools necessary to implement a wide variety of licensing strategies while allowing developers to determine the level of enforcement appropriate for their software products.

Chapter 8 - Support Tracking

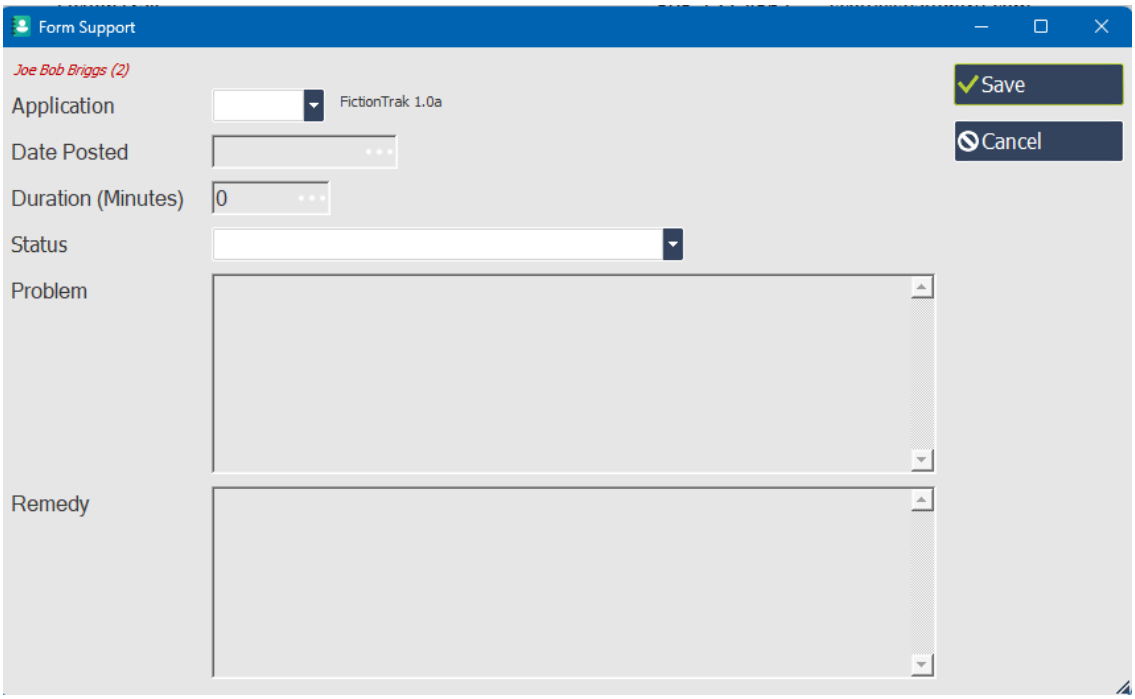
Creating Support Records

LicenseTrak includes an integrated Support Tracking system that allows developers to record and maintain customer support interactions.

Support Records provide a historical record of assistance provided to customers and are linked directly to Customer records within the LicenseTrak database.

To create a new Support Record:

1. Open the Browse Customers window.
2. Select the desired Customer record.
3. Click the New button located within the Support Calls section.
4. Complete the Support Call Form.
5. Click Save.



A Support Record may be created for any customer regardless of the number or type of licenses associated with that customer.

Typical reasons for creating a Support Record include:

- Answering customer questions.
- Troubleshooting application problems.
- Documenting installation assistance.
- Recording licensing issues.
- Tracking feature requests or enhancement suggestions.

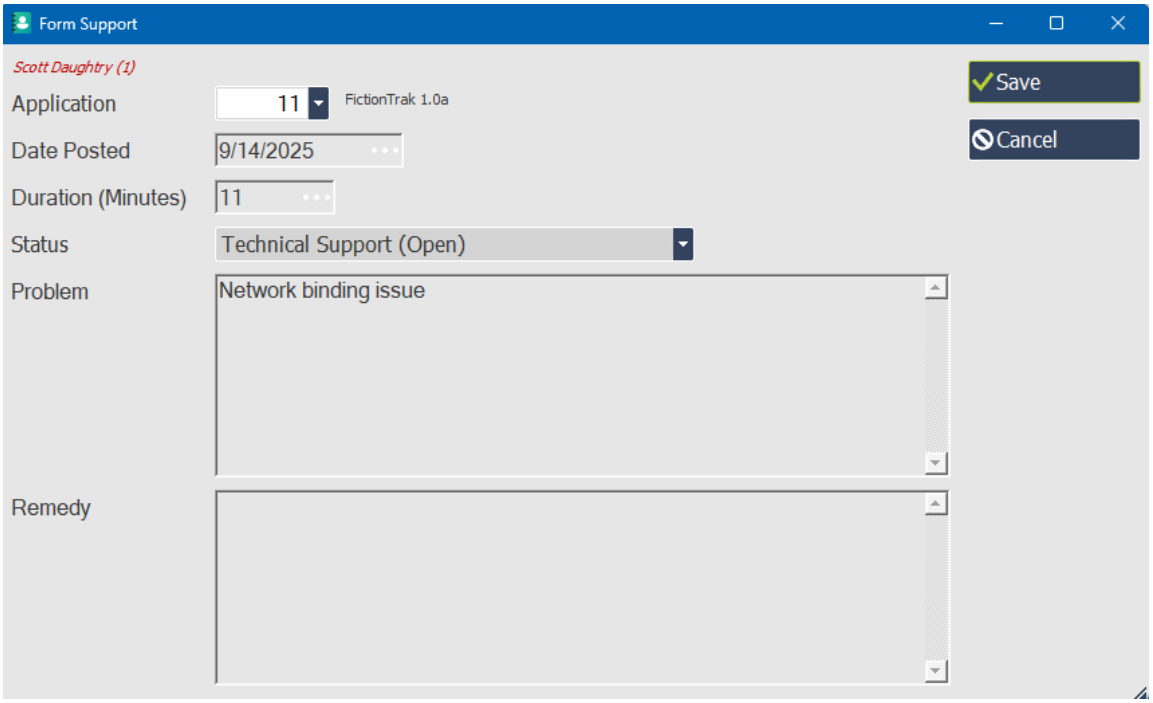
Maintaining accurate Support Records can simplify future customer interactions and provide valuable historical information for the software publisher.

Updating Support Records

Support Records may be updated at any time to reflect the current status of a customer issue or support request.

To update an existing Support Record:

1. Open the Browse Customers window.
2. Select the desired Customer record.
3. Select the appropriate Support Record within the Support Calls section.
4. Click the Edit button.
5. Update the desired information.
6. Click Save.



Support Records are often updated as additional information becomes available or as troubleshooting efforts progress.

Typical reasons for updating a Support Record include:

- Recording additional troubleshooting steps.

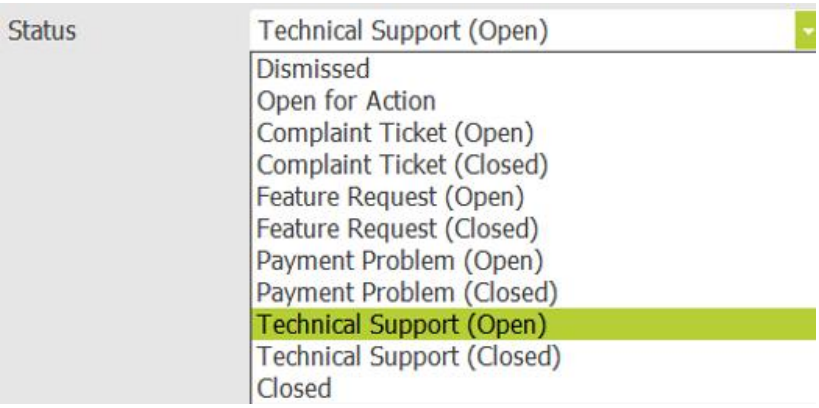
- Updating the current support status.
- Documenting customer follow-up information.
- Recording the final resolution of an issue.
- Adding notes discovered during the support process.

Maintaining complete and up-to-date Support Records provides a valuable historical reference for future customer interactions.

Support Status Values

Each Support Record includes a Support Status value that indicates the current state of the support activity.

The Support Status field allows developers to quickly identify whether a customer issue remains active or has been resolved.



Typical Support Status values may include:

- Open
- Pending
- Closed

The specific values available depend upon the configuration of the LicenseTrak Administration Program.

An Open status generally indicates that work is actively being performed.

A Pending status may indicate that additional information is required from the customer or that a response is awaiting further action.

A Closed status indicates that the support issue has been resolved and no additional action is currently required.

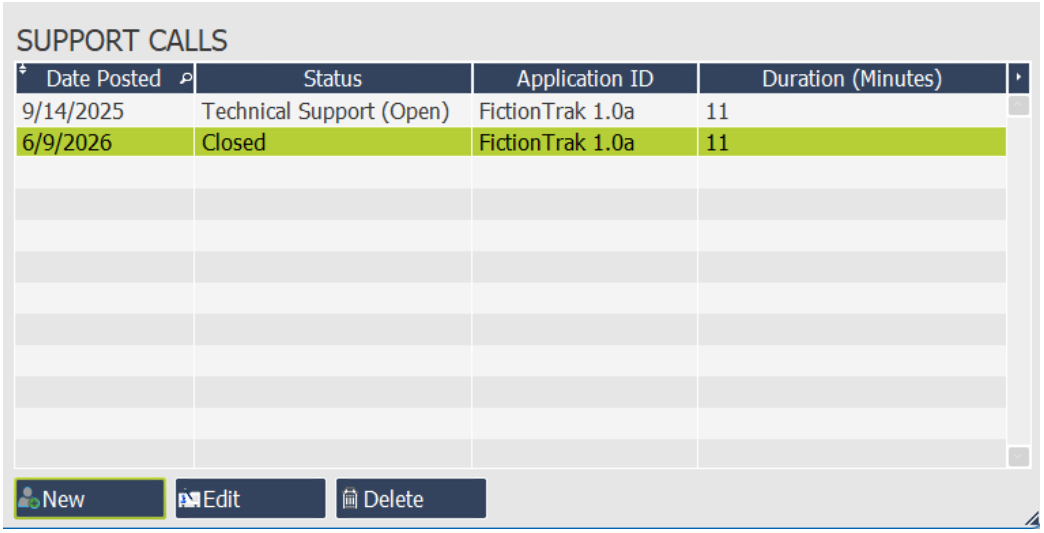
Maintaining accurate Support Status values allows developers to quickly identify outstanding customer issues and monitor support workload.

Reporting and History

One of the primary benefits of maintaining Support Records within LicenseTrak is the ability to review historical customer interactions.

Over time, Support Records become a valuable knowledge base that documents common problems, troubleshooting techniques, and successful resolutions.

Because Support Records are linked directly to Customer and License information, developers may quickly review a customer's support history alongside their licensing history.



Historical Support Records may assist developers in:

- Reviewing previous customer interactions.
- Identifying recurring application issues.
- Reusing successful troubleshooting procedures.
- Tracking feature requests and enhancement suggestions.
- Understanding long-term customer support trends.

Well-maintained Support Records often reduce the amount of time required to resolve future issues by providing immediate access to information gathered during previous support activities.

Best Practices

The following recommendations have proven useful when maintaining Support Records within LicenseTrak.

Record Information Promptly

Create or update Support Records as soon as practical after interacting with a customer. Important details are less likely to be forgotten when documented immediately.

Document The Resolution

Whenever possible, record the final solution to a customer's issue. Similar problems often reoccur, and previous resolutions can significantly reduce future troubleshooting time.

Use Consistent Status Values

Maintain Support Status values consistently to provide an accurate view of outstanding and completed support activities.

Include Meaningful Notes

Document error messages, configuration details, troubleshooting steps, and other information that may be valuable during future customer interactions.

Preserve Historical Records

Avoid deleting Support Records whenever practical. Historical support information frequently becomes more valuable over time and may assist with future troubleshooting efforts.

View The Customer Holistically

Use the integrated Customer, License, and Support history available within LicenseTrak to develop a complete understanding of the customer's relationship with your software products.

Think Like Future You

A few minutes spent documenting a support interaction today may save hours of research and troubleshooting months or years later.

Chapter 9 - Daily Workflow Examples

Chapter 9 presents several common software publishing scenarios and demonstrates how LicenseTrak may be used to manage each situation.

The examples provided in this chapter are intended to illustrate practical LicenseTrak workflows rather than prescribe a single licensing strategy. Developers should adapt these examples to meet the specific requirements of their applications and customers.

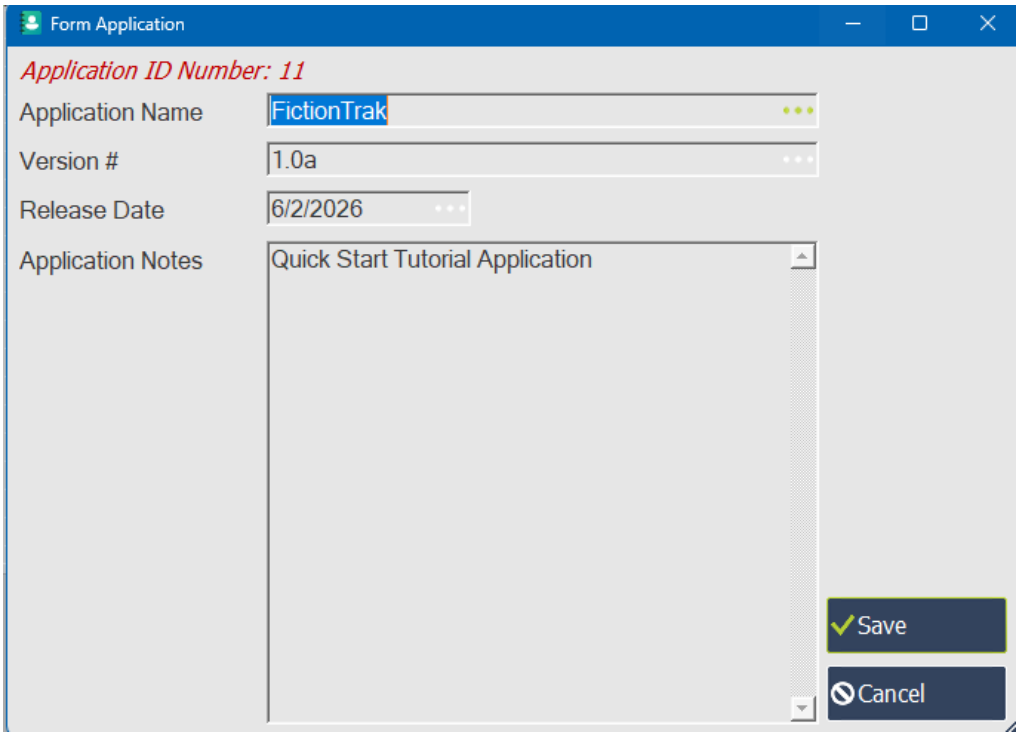
The following sections demonstrate typical day-to-day activities performed by software publishers using the LicenseTrak Administration Program.

Releasing a New Application

One of the first tasks performed by a software publisher is creating an Application record for a new software product.

The following **example** demonstrates a typical workflow for preparing a newly developed application for distribution.

1. Open the LicenseTrak Administration Program.
2. Select File, then Browse Application Picklist.
3. Click the New button.
4. Enter the Application Name, Version Number, Release Date, and any desired Application Notes.
5. Click Save to create the new Application record.



Once the Application record has been created, it becomes available for Trial and Registered license generation activities.

Many developers also use this opportunity to record version-specific notes, release information, or upgrade policies within the Application Notes field for future reference.

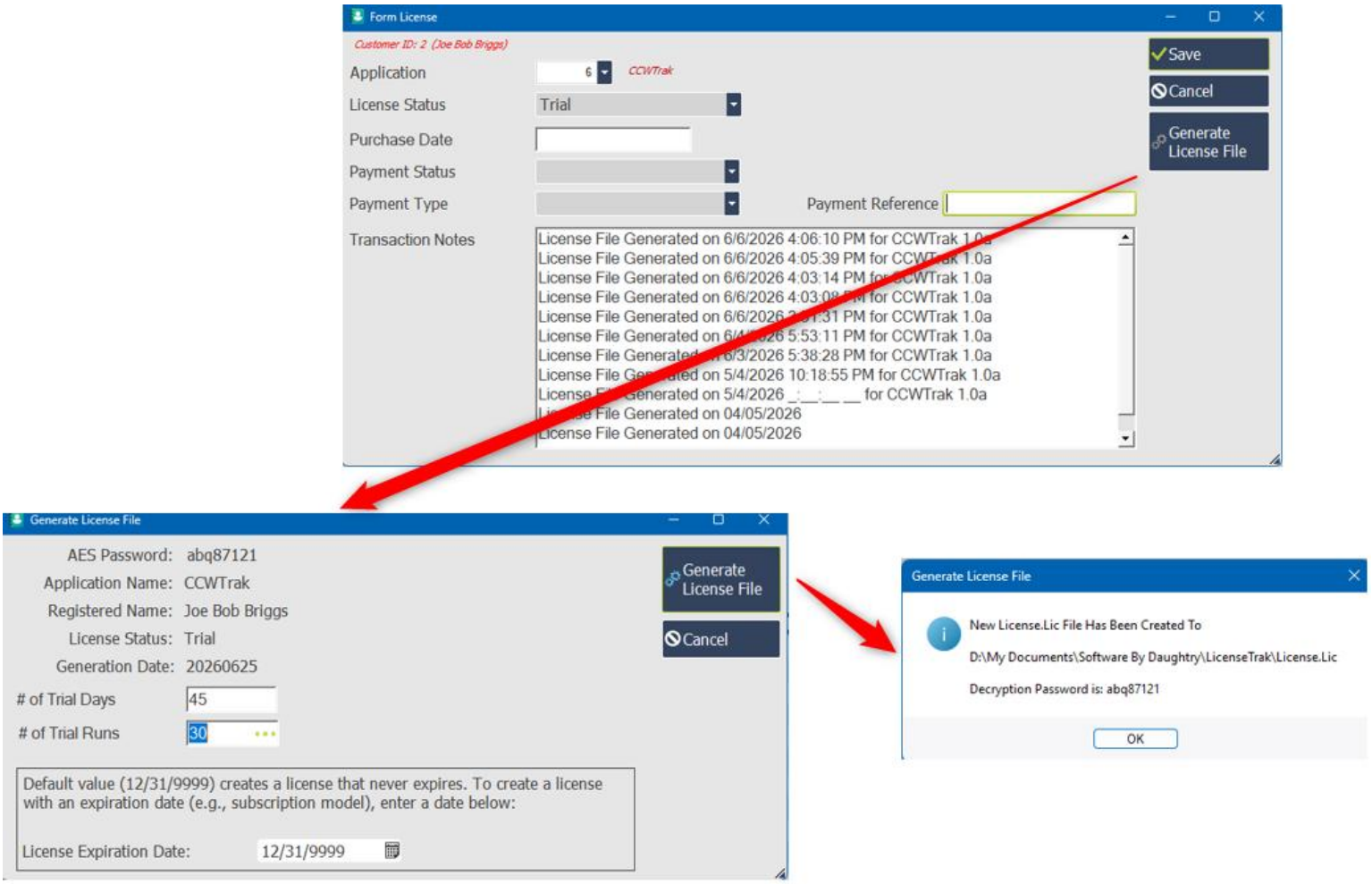
Publishing a Trial Version

After an Application record has been created, many software publishers choose to distribute a Trial version of the application to allow prospective customers to evaluate the product before making a purchase.

A common LicenseTrak workflow is to generate a single Trial License.LIC file that is included within the application's installation package.

The following **example** demonstrates a typical Trial distribution process:

1. Create or select the Application record.
2. Create or select the Trial User customer record.
3. Create a new License record associated with the Trial User.
4. Select Trial as the License Status.
5. Specify the desired Trial Days and/or Trial Runs values.
6. Generate the Trial License.LIC file.
7. Include the generated License.LIC file within the application's Setup.exe installation package.



When a prospective customer downloads and installs the application, the included Trial License.LIC file allows immediate application evaluation without requiring Internet activation, online registration, or communication with a licensing server.

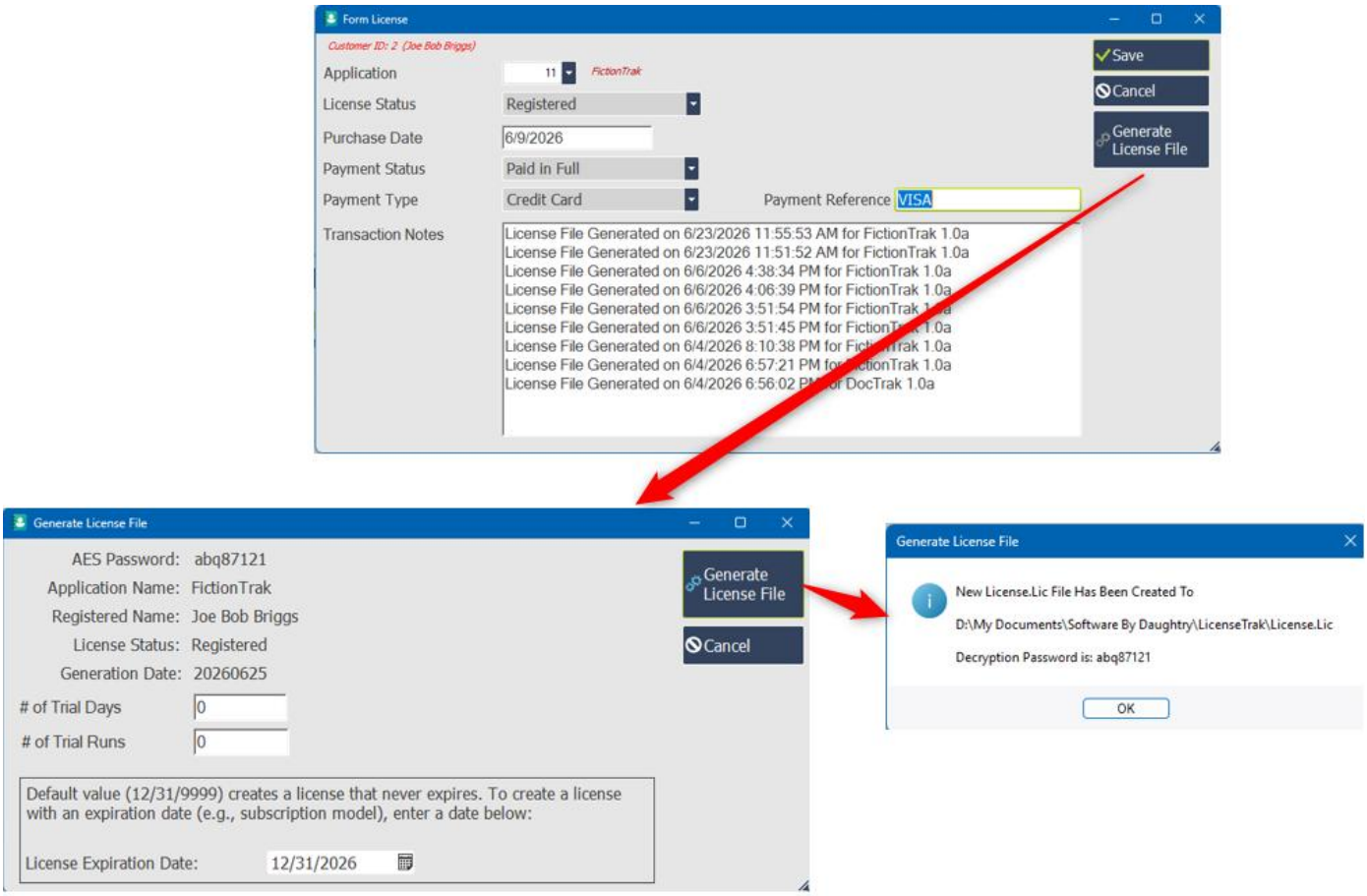
Many developers use a generic customer record such as "Trial User" for this **purpose**, allowing the same Trial License.LIC file to be distributed with every evaluation copy of the application.

Processing a Customer Purchase

When a customer decides to purchase the software, the Trial License.LIC file may be replaced with a Registered License.LIC file generated specifically for that customer.

A typical purchase workflow is as follows:

1. Open the Browse Customers window.
2. Create a new Customer record if one does not already exist.
3. Create a new License record associated with the customer and the purchased application.
4. Select Registered as the License Status.
5. Record any desired payment or transaction information.
6. Generate the Registered License.LIC file.
7. Deliver the License.LIC file to the customer by email or other preferred distribution method.



The customer simply replaces the existing Trial License.LIC file with the newly provided Registered License.LIC file.

The next time the application starts, the LicenseTrak Runtime Library detects the Registered License and the application operates according to the developer's licensing policies.

This workflow allows developers to transition customers from evaluation copies to fully licensed software without requiring software reinstallation or online activation procedures.

Renewing a Subscription

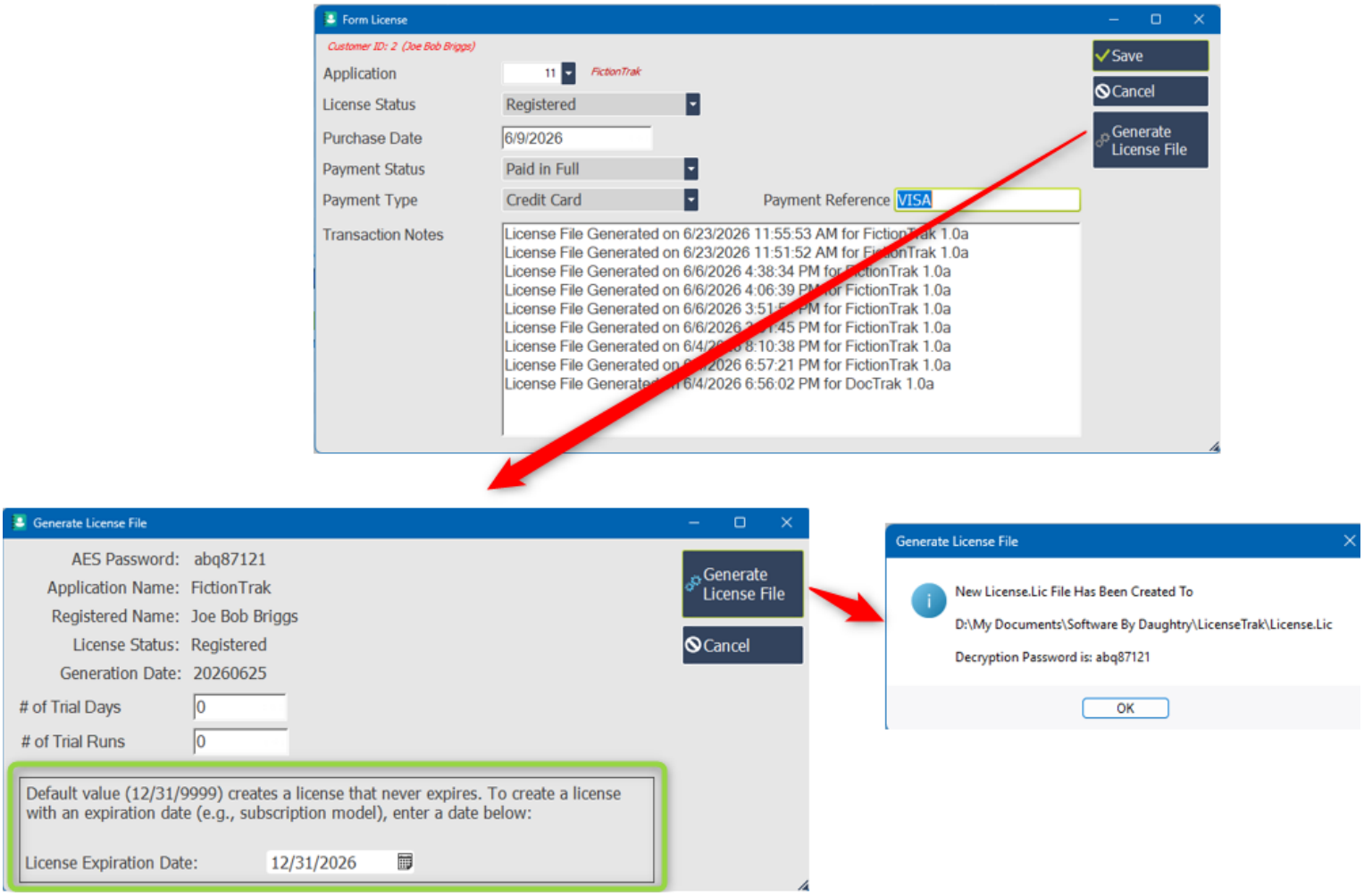
Developers who implement subscription-based licensing may periodically need to extend a customer's licensed usage period.

In LicenseTrak, subscription models are typically implemented by generating a Registered License.LIC file that contains an updated License Expiration Date.

A typical subscription renewal workflow is as follows:

1. Open the Browse Customers window.
2. Locate and select the customer.
3. Open the customer's existing License record.
4. Click the Generate License File button.
5. Specify the new License Expiration Date.
6. Generate the updated License.LIC file.

7. Deliver the replacement License.LIC file to the customer.



The customer simply replaces the previous License.LIC file with the updated version.

Because LicenseTrak stores the customer's licensing history, subscription renewals may be performed quickly without recreating Application, Customer, or License records.

This approach allows developers to implement subscription licensing without requiring Internet activation servers or recurring online validation.

Replacing a Lost License

Occasionally, a customer may lose a License.LIC file due to circumstances beyond their control. Common situations include:

- Hard drive or SSD failures.
- Computer replacement.
- Operating system reinstallation.
- Accidental file deletion.
- Data migration to a new computer.

One of the primary advantages of LicenseTrak is the ability to quickly generate a replacement License.LIC file using information already stored within the LicenseTrak database.

A typical replacement workflow is as follows:

1. Open the Browse Customers window.
2. Locate and select the customer.
3. Review the customer's existing License record.
4. Click the Generate License File button.
5. Generate a replacement License.LIC file.
6. Deliver the replacement file to the customer by email or another preferred method.

Because the original Application, Customer, and License records are preserved within LicenseTrak, replacement licenses can often be generated years after the original software purchase.

This capability significantly reduces customer support effort and reinforces the importance of maintaining complete historical records within the LicenseTrak database.

Customer Changes Computers

A common support scenario occurs when a customer replaces or upgrades their computer and needs to transfer a licensed application to the new system.

Because LicenseTrak uses a portable License.LIC file rather than an Internet activation server, the transition process is straightforward.

A typical workflow is as follows:

1. Confirm the customer's identity and existing license information.
2. Determine whether the customer still possesses a copy of the current License.LIC file.
3. If necessary, locate the customer's License record within LicenseTrak.
4. Generate a replacement License.LIC file.
5. Provide the customer with the replacement file and instructions for copying it into the application's installation folder.

In many cases, the customer may simply copy the existing License.LIC file from the old computer to the new computer. However, when the original file is unavailable, LicenseTrak allows the software publisher to quickly generate a replacement using the information already stored in the LicenseTrak database.

This approach minimizes customer downtime and avoids the need for online activation resets or manual license recovery procedures.

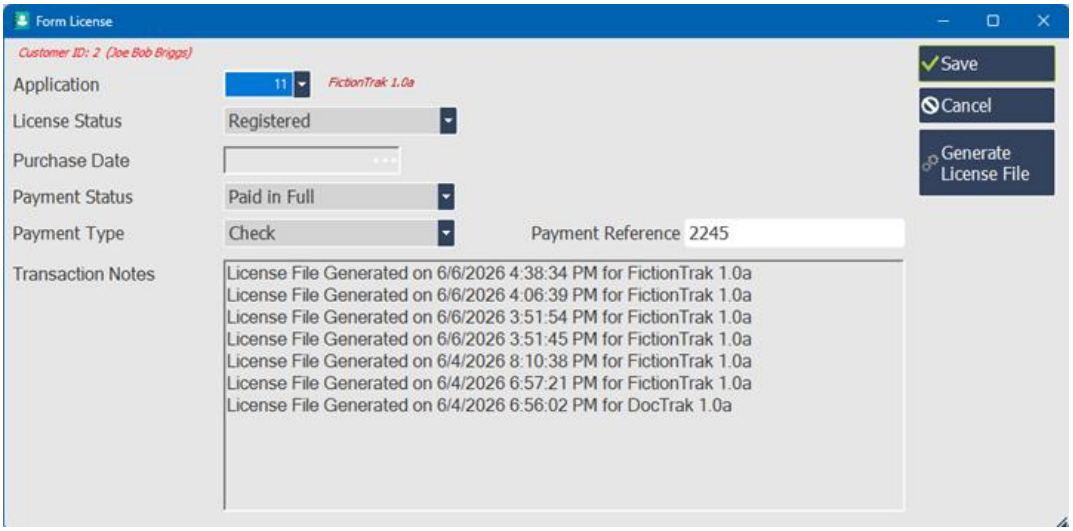
Investigating License Abuse

Occasionally, a software publisher may encounter circumstances suggesting that a License.LIC file has been modified, duplicated, or otherwise used in a manner inconsistent with the developer's licensing policies.

LicenseTrak assists developers by maintaining licensing information and providing Runtime Library functions that can record and evaluate license validation events and potential tampering activity.

When investigating a suspected licensing issue, a typical workflow may include:

1. Review the customer's Application, License, and Support history.
2. Examine any violation or tamper information recorded by the protected application.
3. Determine whether the observed behavior is the result of a legitimate customer issue or a potential licensing violation.
4. Decide upon an appropriate course of action based upon the developer's licensing policies.



Many apparent licensing problems are ultimately caused by routine events such as hardware failures, operating system reinstalls, accidental file deletion, or customer misunderstanding rather than intentional misuse.

For this reason, developers should review all available information before taking enforcement action.

LicenseTrak provides the information necessary to support these decisions while leaving the final enforcement policy under the control of the software publisher.

PART II - INTEGRATING LICENSETRAK

Chapter 10 - Getting Started

Part II of this manual describes the process of integrating the LicenseTrak Runtime Library into a WinDev application.

Unlike the LicenseTrak Administration Program, which is used by the software publisher to manage applications, customers, and licenses, the Runtime Library is incorporated directly into the protected application.

The examples presented in this section assume that the developer has:

1. Installed the LicenseTrak Administration Program.
2. Defined a Developer Seed.
3. Created an Application record.
4. Generated a Trial or Registered License.LIC file.

The goal of this chapter is to prepare a WinDev application to leverage the LicenseTrak Runtime Library for license

validation and enforcement.

The integration process consists of the following steps:

1. Configuring the imported License File analysis.
2. Importing the LicenseTrak Runtime Library.
3. Verifying the Runtime Library installation.
4. Deploying the License.LIC file.
5. Performing the first Runtime Library initialization.
6. Each of these steps is described in detail within the sections that follow.

Preparing Your Application

Before integrating the LicenseTrak Runtime Library, it is recommended that developers begin with a stable, compilable WinDev project.

LicenseTrak is designed to be integrated into existing applications with minimal disruption to the developer's existing database structures and application architecture.

The Runtime Library does not require developers to adopt the internal Application, Customer, License, or Support Tracking databases used by the LicenseTrak Administration Program.

Instead, the protected application imports only the LicenseTrak License File definition and the LicenseTrak Runtime Library required for license management.

For the examples presented in this chapter, a simple WinDev application will be used to demonstrate the integration process which has a single database defined (Customer.Fic):

The same procedures may be applied to both new and existing WinDev projects.

Importing the LicenseTrak License File Definition

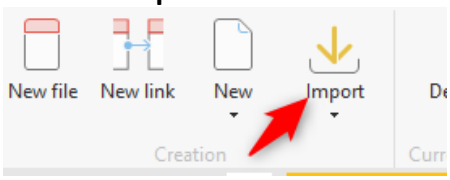
The LicenseTrak Runtime Library stores licensing information within a HyperFile data file named License.LIC. Before the Runtime Library can be used, the associated License File definition must be added to the application's Analysis.

The License File definition is supplied with LicenseTrak as the file **License.LIC**.

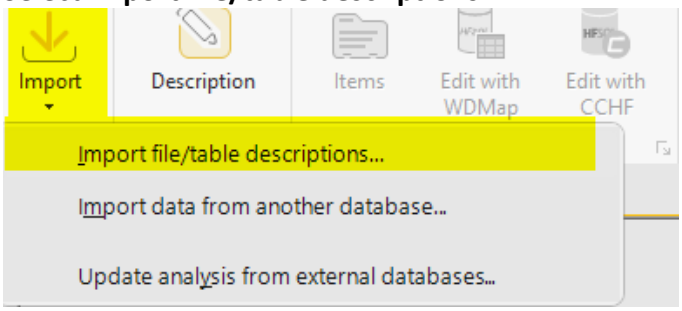
To import the License File definition, change the file selection filter from *.FIC to *.*; the License.LIC file will now

appear. Select the file:

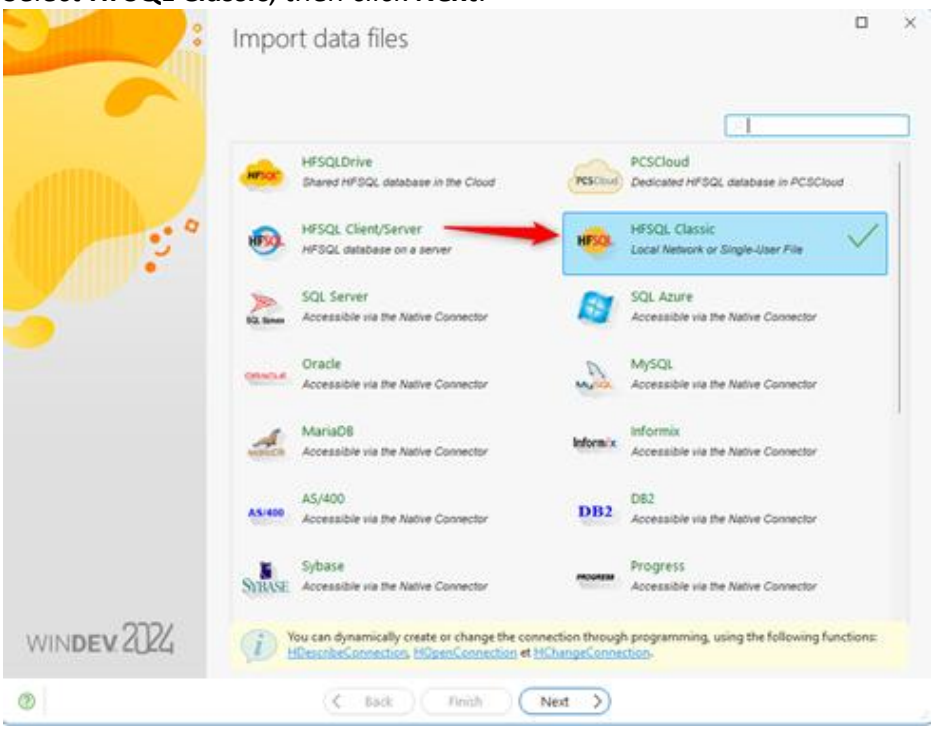
1. Copy the supplied **License.FIC** file into the target application's project folder.
2. Open the WinDev project.
3. Open the project's Analysis.
4. Click the **Import** toolbar button.



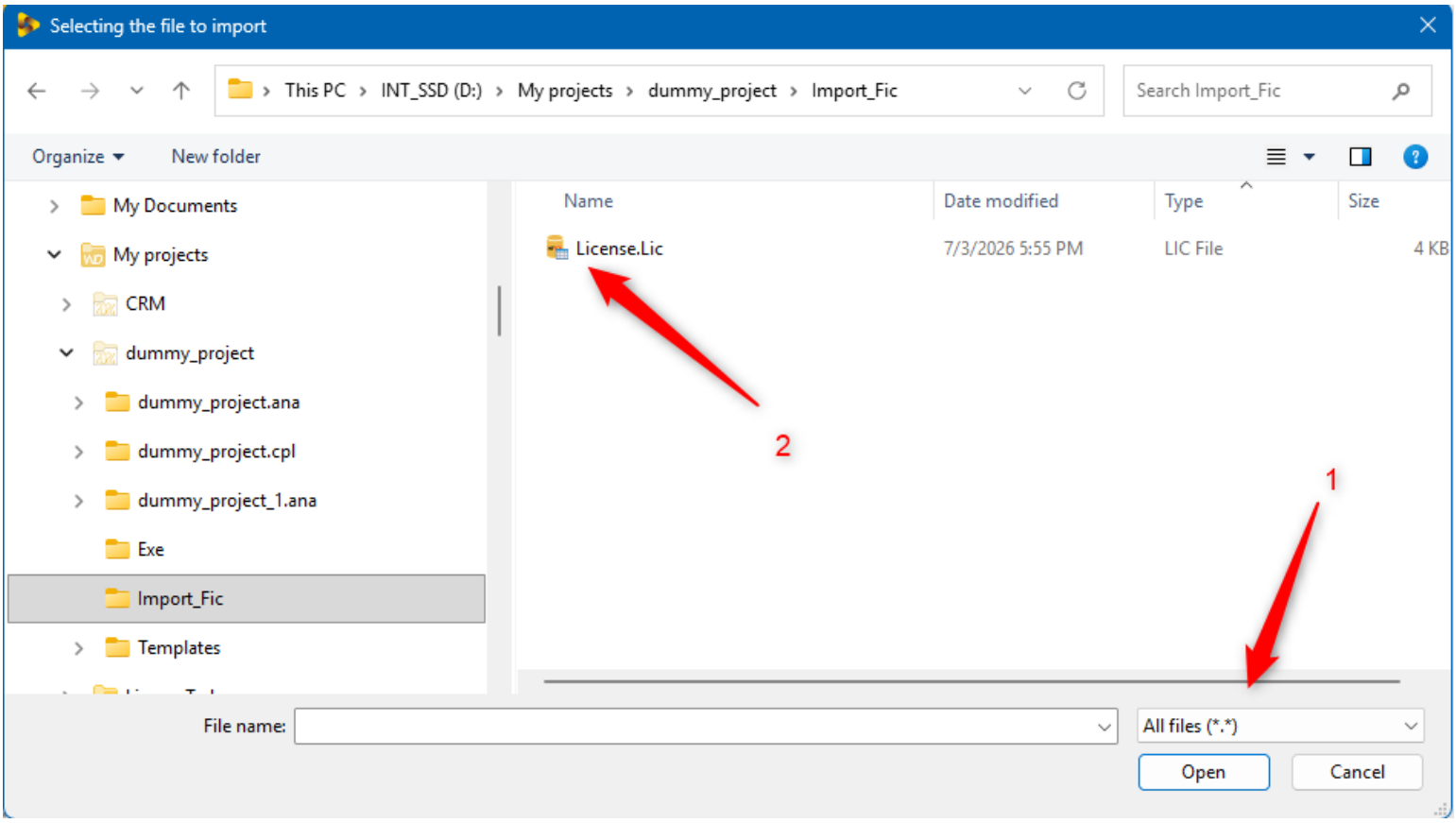
5. Select **Import file/table descriptions**.



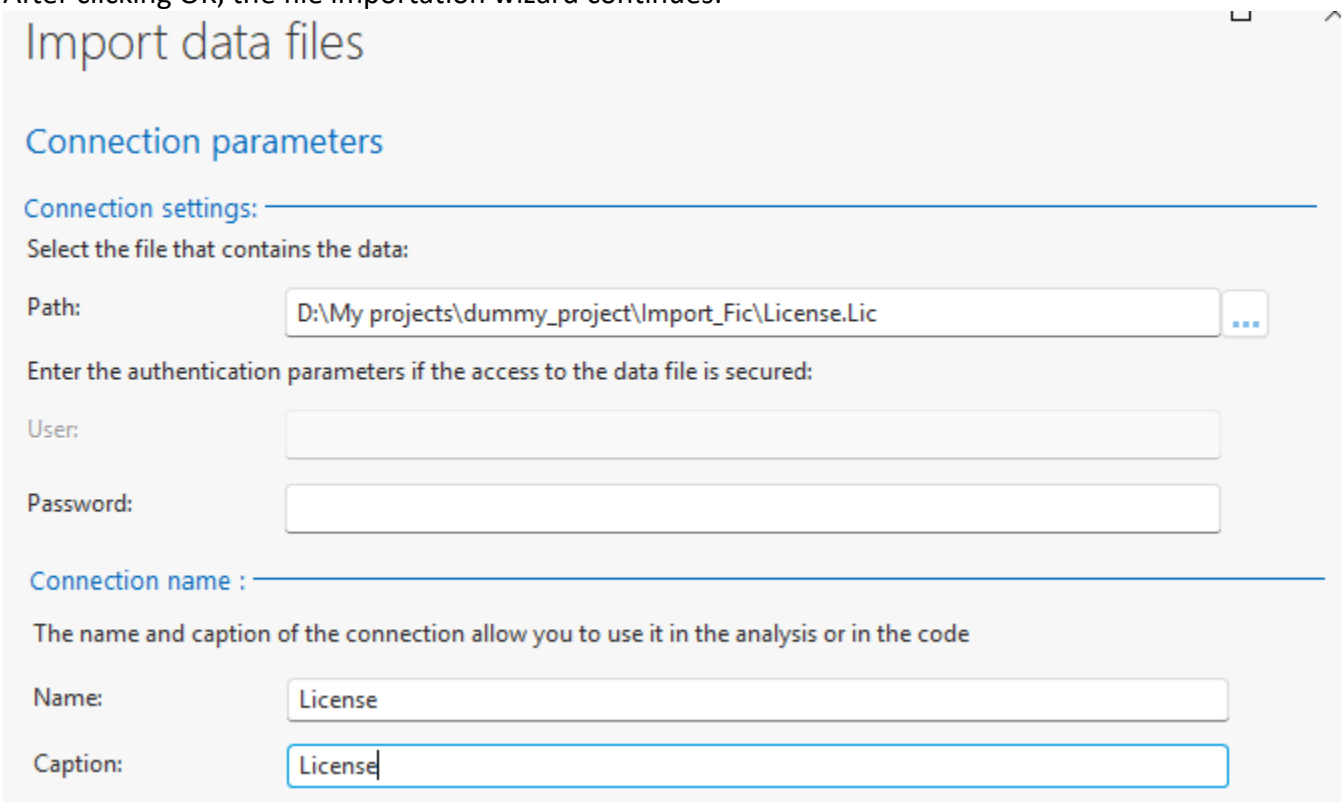
6. Select **HFSQL Classic**, then click **Next**.



7. Browse to and select the supplied **License.LIC** file.

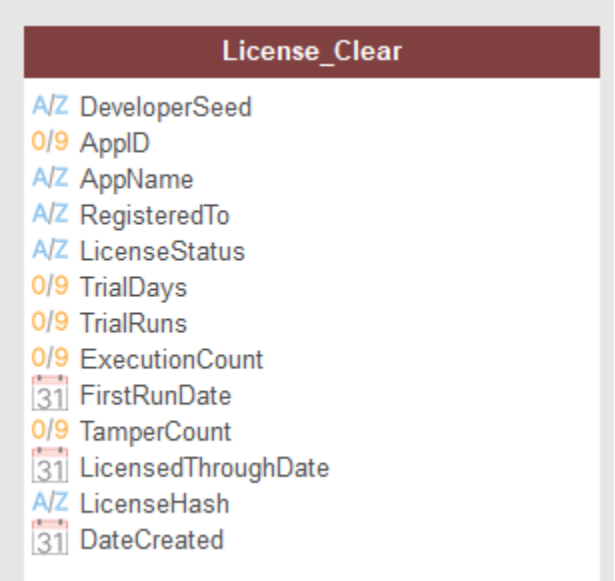


After clicking OK, the file importation wizard continues:



Click FINISH to import the file structure into the WinDev analysis file. The analysis is updated and will show the newly

imported License_Clear file:

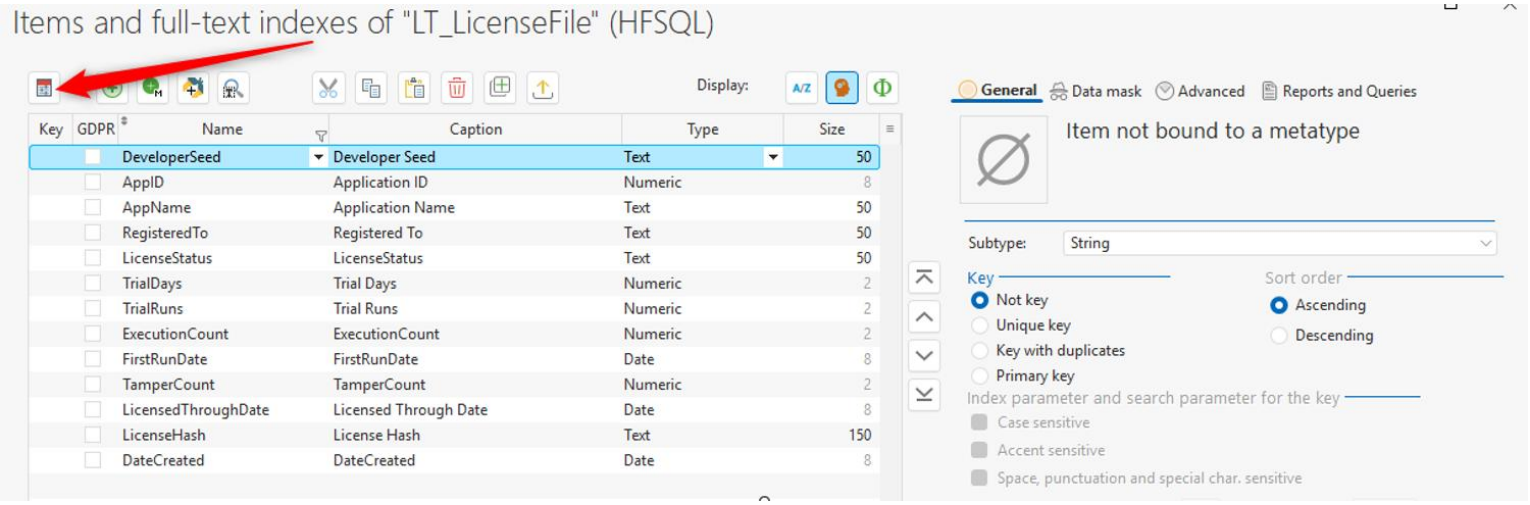


Configuring the License File Analysis

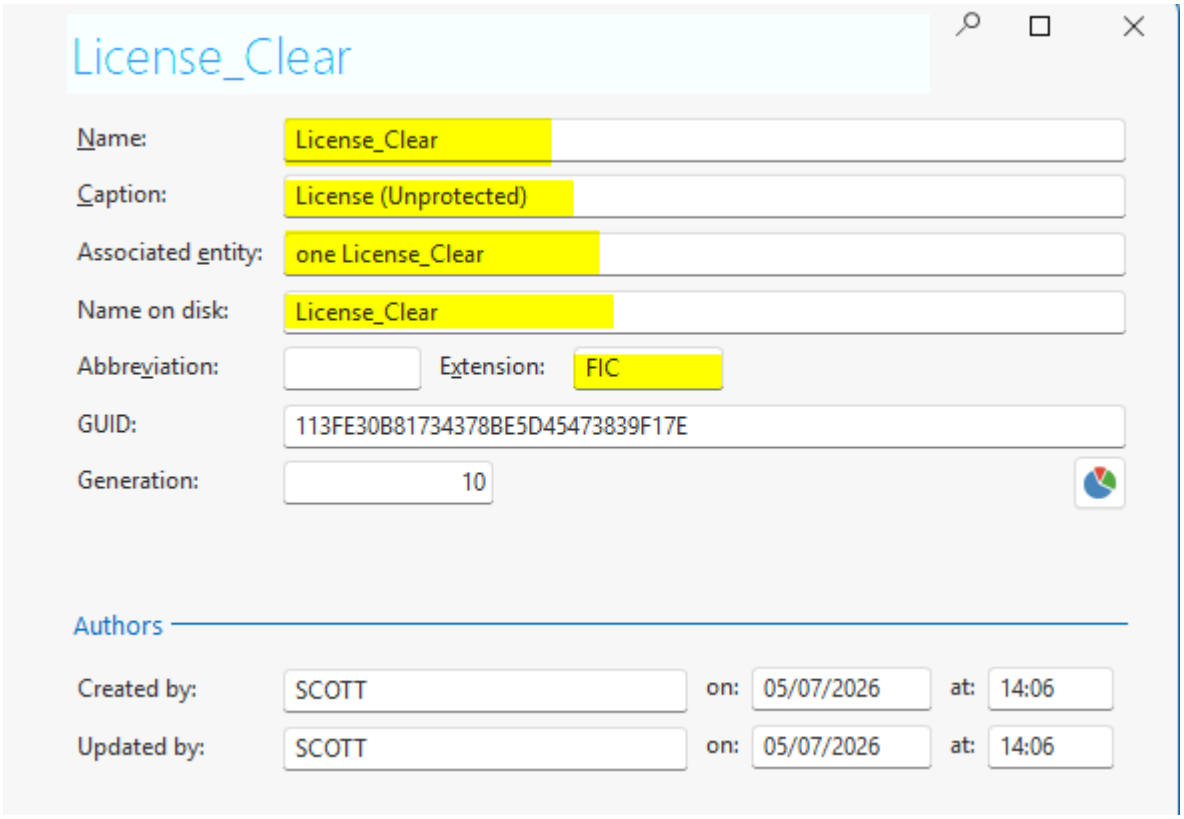
After importing the License File definition, several Analysis properties should be reviewed and updated to ensure compatibility with the LicenseTrak Runtime Library.

To modify the imported License File definition:

1. Open the application's Analysis.
2. Open the imported **License_Clear** file definition.
3. Click the **Properties of the Data File** button.



Windev assigns undesired properties from the license.fic importation; the highlighted property fields require modification:

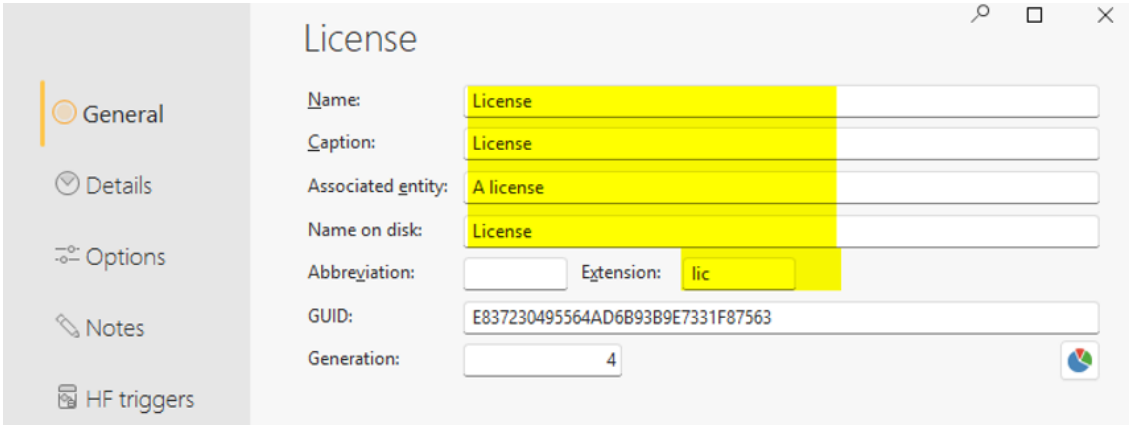


Review the imported properties and update the highlighted fields as described in the following examples.

The recommended values are:

- Name: **License**
- Caption: **License**
- Associated Entity: **A license**
- Name on Disk: **License**
- Extension: **LIC**

The corrected property screen now resembles this:



After the changes have been completed, save the modified Analysis and regenerate the project if prompted.

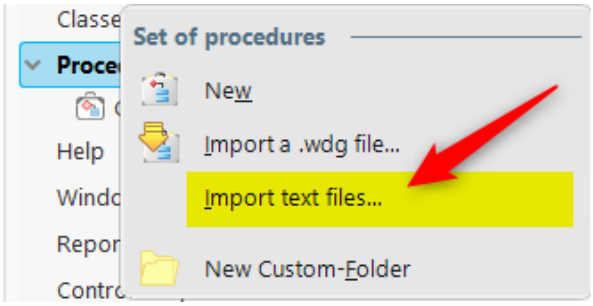
The application is now prepared for importing the LicenseTrak Runtime Library.

Importing the LicenseTrak Runtime Library

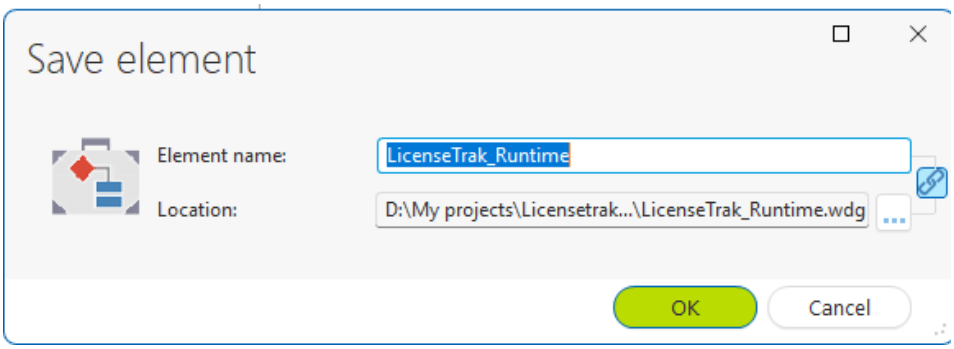
The LicenseTrak Runtime Library is supplied as a WLanguage source file named LicenseTrak_Runtime.WL. This file contains the collection of global procedures/functions required to integrate LicenseTrak licensing services into a WinDev application.

To import the Runtime Library:

1. Copy the supplied **LicenseTrak_Runtime.WL** file into the target application's project folder (it is recommeded to create a subfolder and copy the LicenseTrak_Runtime.WL file there).
2. Open the WinDev project.
3. Display the **Project Explorer** pane.
4. Right-click on the **Procedures** folder.
5. Select **Import Text Files** from the popup menu.
6. Browse to and select the **LicenseTrak_Runtime.WL** file.
7. Complete the import process.



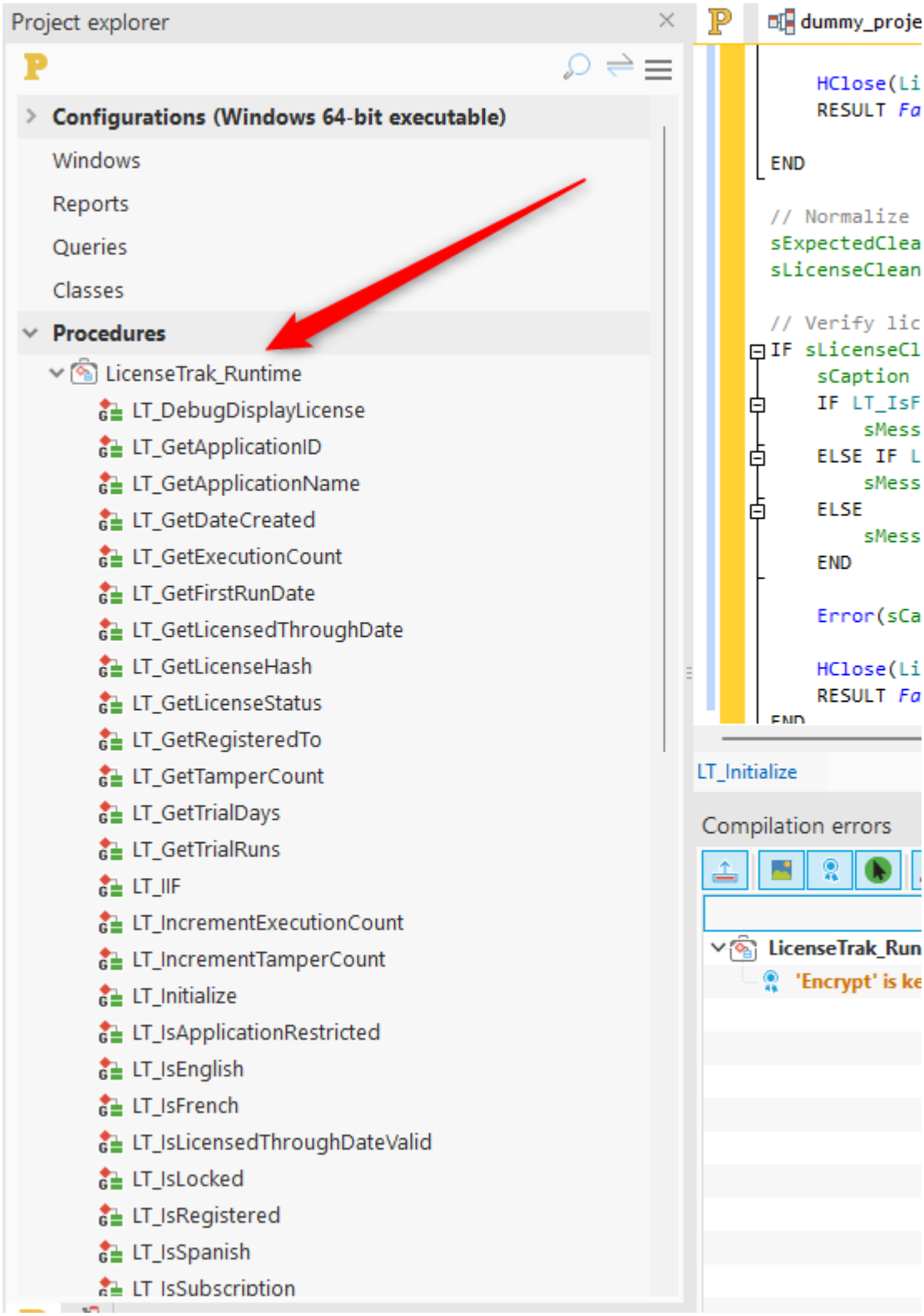
A popup window appears; accept the default values and click OK to continue:



After the import has completed successfully, the LicenseTrak Runtime Library procedures will appear within the Project Explorer and become available for use by the application.

The Runtime Library is supplied as source code, allowing developers to review, examine, and understand the procedures that provide LicenseTrak functionality.

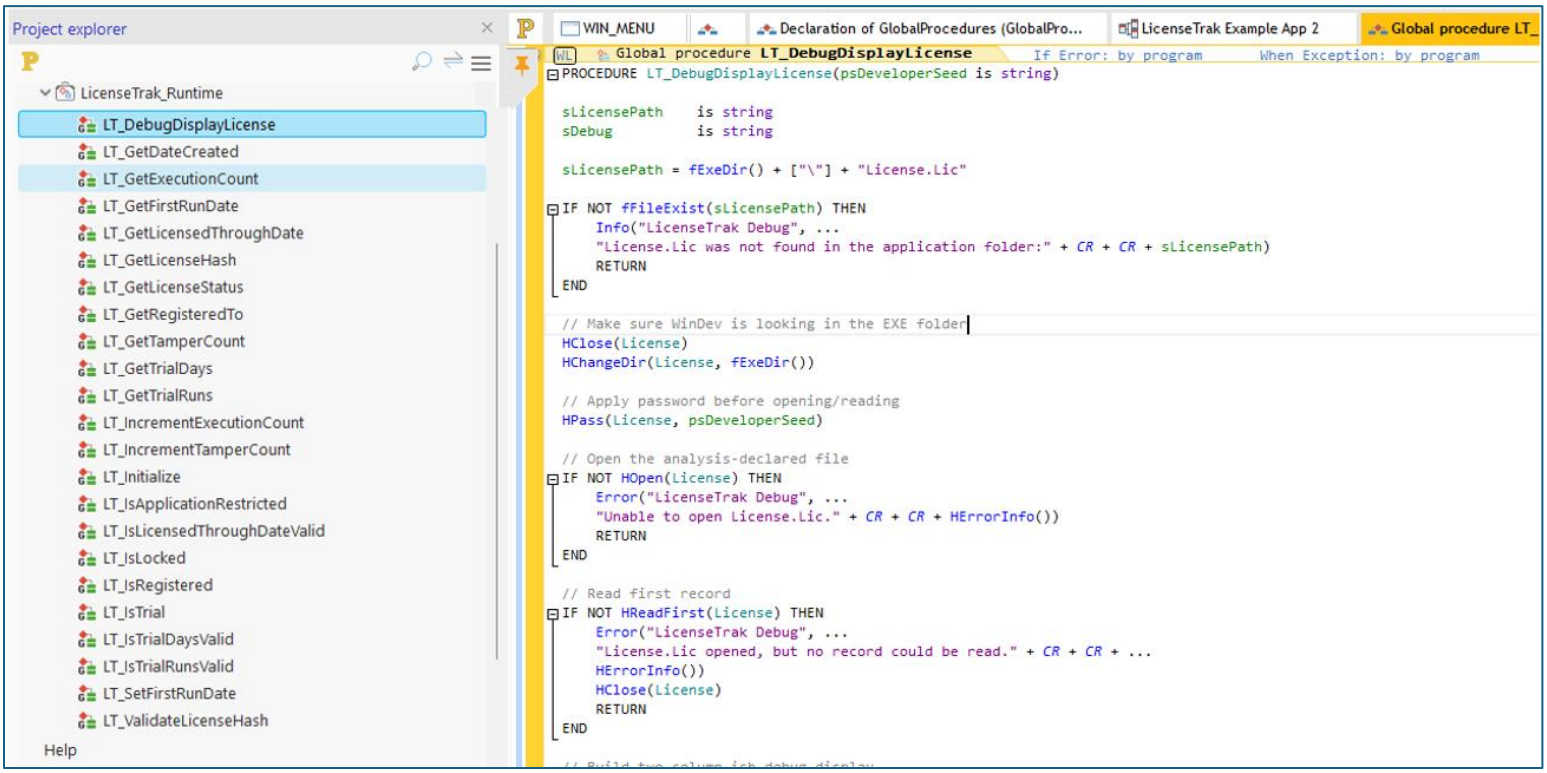
The WinDev IDE will now display the full global LicenseTrak function library that is visible anywhere in your application:


Project explorer
Configurations (Windows 64-bit executable)
Windows
Reports
Queries
Classes
Procedures
LicenseTrak_Runtime
LT_DebugDisplayLicense
LT_GetApplicationID
LT_GetApplicationName
LT_GetDateCreated
LT_GetExecutionCount
LT_GetFirstRunDate
LT_GetLicensedThroughDate
LT_GetLicenseHash
LT_GetLicenseStatus
LT_GetRegisteredTo
LT_GetTamperCount
LT_GetTrialDays
LT_GetTrialRuns
LT_IIF
LT_IncrementExecutionCount
LT_IncrementTamperCount
LT_Initialize
LT_IsApplicationRestricted
LT_IsEnglish
LT_IsFrench
LT_IsLicensedThroughDateValid
LT_IsLocked
LT_IsRegistered
LT_IsSpanish
LT_IsSubscription
dummy_proje
HClose(Li
RESULT Fa
END
// Normalize
sExpectedClea
sLicenseClean
// Verify lic
IF sLicenseCl
sCaption
IF LT_IsF
sMess
ELSE IF L
sMess
ELSE
sMess
END
Error(sCa
HClose(Li
RESULT Fa
END
LT_Initialize
Compilation errors
LicenseTrak_Run
'Encrypt' is ke

Verifying the Runtime Library Installation

After importing the LicenseTrak Runtime Library, it is recommended that developers verify the installation before proceeding with application integration.

Begin by expanding the **Procedures** section of the Project Explorer. The imported LicenseTrak Runtime Library procedures should now be visible.

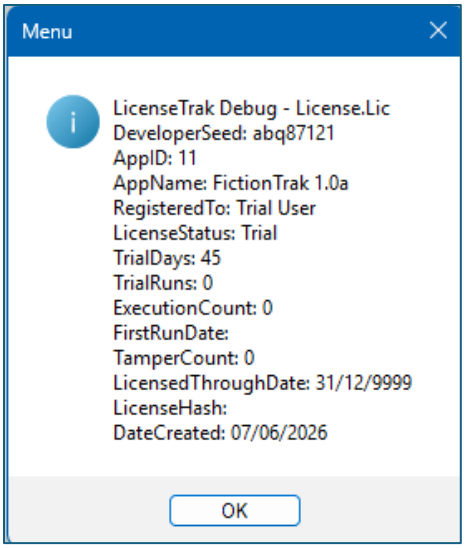


Next, perform a project compilation to verify that the Runtime Library has been successfully integrated into the application.

A successful compilation confirms that:

- The LicenseTrak Runtime Library has been imported correctly.
- The imported License File definition is available to the Runtime Library.
- The application is ready for LicenseTrak initialization and testing.

As an additional verification step, developers may create a temporary menu option or button that calls the **`LT_DebugDisplayLicense()`** procedure. When executed, this procedure displays the contents of the currently deployed License.LIC file and confirms that the Runtime Library can successfully access and interpret licensing information.



Successful execution of the debug display confirms that the Runtime Library installation has been completed correctly and that the application is ready for the remaining integration steps.

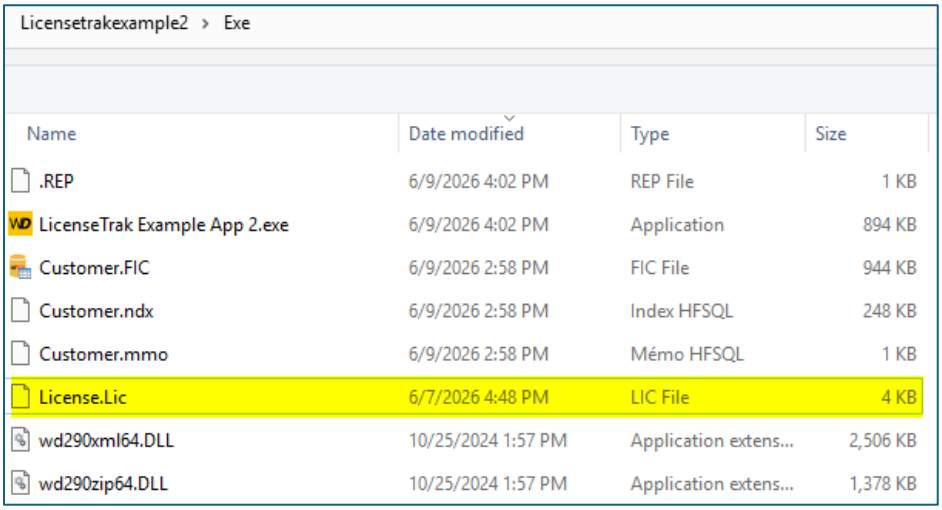
Deploying License.LIC

After the LicenseTrak Runtime Library has been successfully imported and verified, a License.LIC file must be deployed with the protected application.

The License.LIC file generated by the LicenseTrak Administration Program contains the licensing information that will be evaluated by the Runtime Library during application execution.

To deploy the License.LIC file:

1. Generate a Trial or Registered License.LIC file using the LicenseTrak Administration Program.
2. Copy the generated License.LIC file into the application's execution folder.
3. Rebuild or launch the application.



During application startup, the LicenseTrak Runtime Library automatically accesses the License.LIC file and retrieves the licensing information stored within it.

For Trial distributions, the License.LIC file may be included directly within the application's installation package.

For Registered customers, the License.LIC file may be distributed separately by email, download portal, or other preferred delivery method.

Because LicenseTrak uses a portable license file architecture, no Internet activation, cloud licensing service, or external validation server is required.

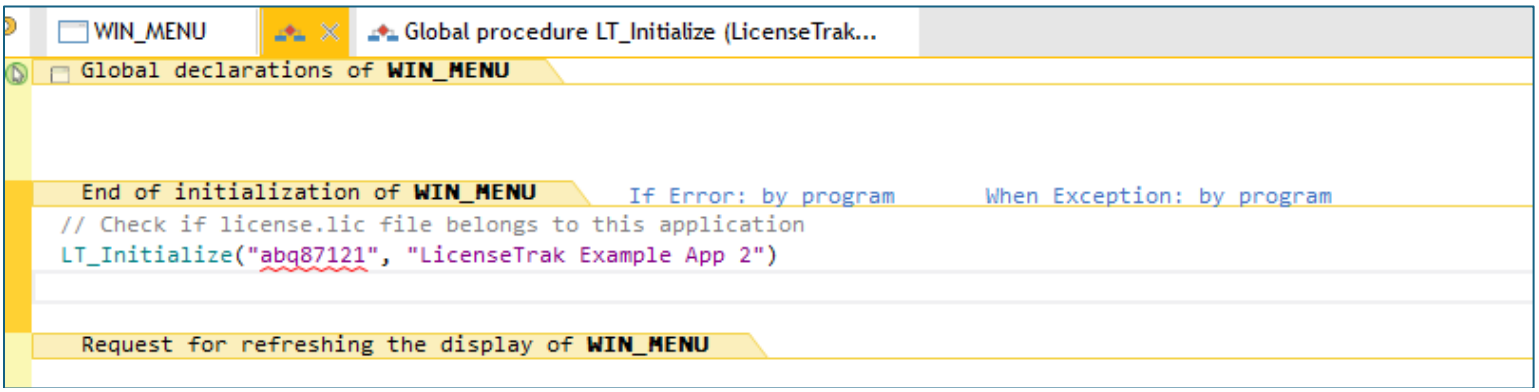
First Application Initialization

With the LicenseTrak License File definition imported, the Runtime Library installed, and a License.LIC file deployed, the application is ready to perform its first LicenseTrak initialization.

LicenseTrak initialization is performed by calling the LT_Initialize() procedure during application startup. A common location for this call is within the Main window's initialization code, although developers may invoke the procedure at any point before licensing decisions are required.
The LT_Initialize() procedure performs several important tasks automatically:

- Locates the License.LIC file.
- Reads the licensing information stored within the file.
- Validates the application identity.
- Prepares the Runtime Library for subsequent licensing operations.

The following code was added to the Main window's **End of initialization of WIN_MENU** event:



After successful execution of LT_Initialize(), the application may use the remaining LicenseTrak Runtime Library procedures to determine licensing status and implement developer-defined licensing policies. Typical examples include:

- Determining whether the application is operating in Trial mode.
- Displaying the Registered User name.
- Evaluating Trial Day or Trial Run limitations.
- Checking license expiration dates.
- Recording and evaluating license validation events.

Testing Application Identity Validation

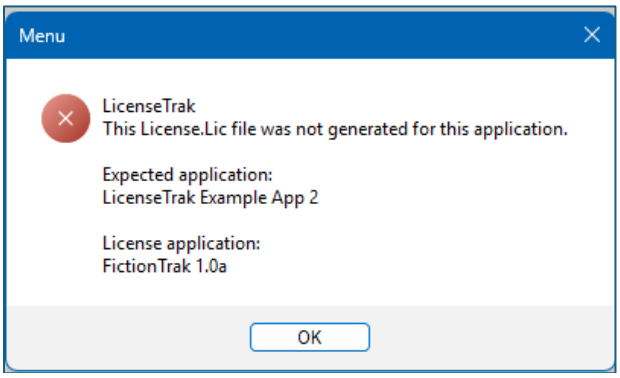
As a final integration test, the **example** application may be executed using a License.LIC file that was intentionally

generated for a different application.

For this **example**, the sample application is named **LicenseTrak Example App 2**, while the deployed License.LIC file was originally generated for **FictionTrak 1.0a**.

When the **example** application is executed, the LicenseTrak Runtime Library compares the expected application identity against the application information stored within the License.LIC file.

Because the License.LIC file was generated for a different application, the Runtime Library detects the mismatch and displays the following validation message:

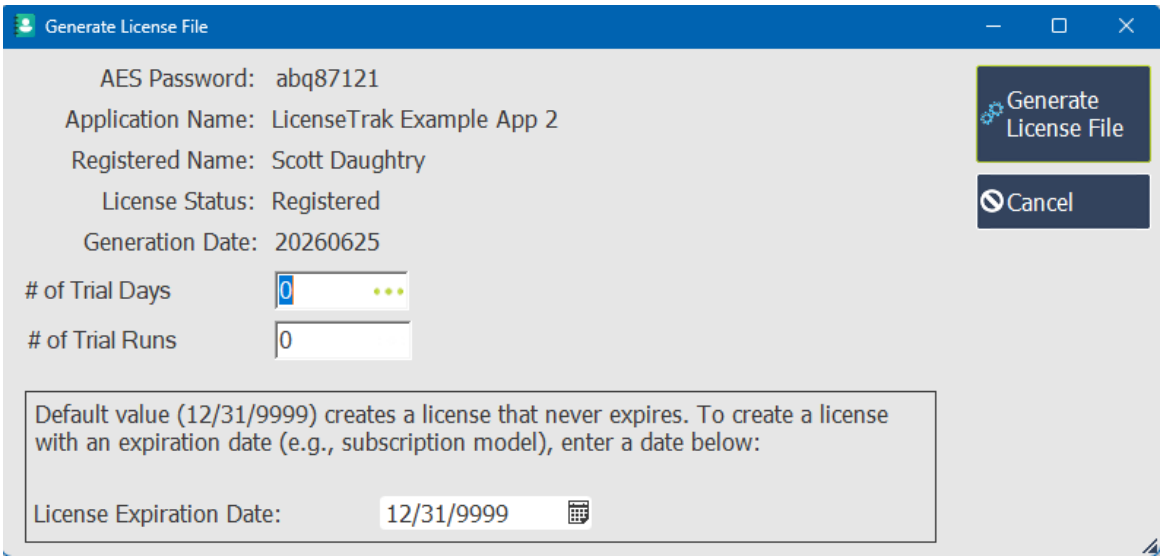


This behavior confirms that the Runtime Library correctly prevents a License.LIC file generated for one protected application from being used by another application.

Verifying Successful Initialization

Next, generate a new License.LIC file using the LicenseTrak Administration Program for the **example** application itself.

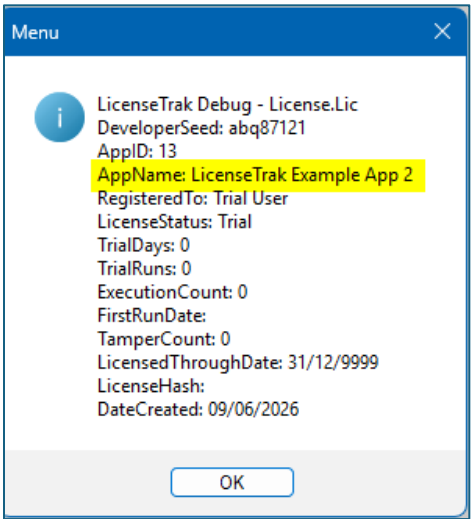
In this **example**, a Trial License.LIC file was generated for the application **LicenseTrak Example App 2**.



Replace the previous License.LIC file with the newly generated version and execute the application again.

Because the License.LIC file now contains the correct application identity information, no LicenseTrak validation message is displayed and the Runtime Library completes initialization successfully.

As a final verification step, execute the `LT_DebugDisplayLicense()` procedure. The displayed information should correspond to the newly generated License.LIC file, including the correct Application Name, License Status, Registered User, Trial configuration, and other licensing information.



Successful execution of LT_Initialize() and LT_DebugDisplayLicense() confirms that:

- The LicenseTrak License File definition has been imported correctly.
- The LicenseTrak Runtime Library has been integrated successfully.
- The deployed License.LIC file is valid for the protected application.
- Application identity validation is functioning correctly.

At this point, the application has been successfully integrated with the LicenseTrak Runtime Library and is ready to implement developer-defined Trial, Registered, and Subscription licensing policies.

Chapter 11 - License File Structure

License.LIC Overview

The LicenseTrak Runtime Library stores licensing information within a HyperFile data file named **License.LIC**.

As described in Chapter 10, the License File definition is supplied with LicenseTrak and is imported directly into the protected application's Analysis. Developers are not required to manually create the License File structure.

The Runtime Library expects the imported License File definition to remain unchanged. The field names, data types, and field lengths must exactly match the supplied definition in order for the Runtime Library procedures to operate correctly.

The License.LIC file is a single-record HyperFile data file containing the information required by the Runtime Library to determine the current licensing status of the protected application.

Unlike a traditional application database, the License.LIC file does not store multiple records. Instead, it functions as a

portable container that travels with the protected application and contains the licensing information associated with that installation.

The License.LIC file uses the file extension **.LIC** rather than the standard HyperFile **.FIC** extension. This distinction helps identify the file as a LicenseTrak license file rather than a conventional application database.

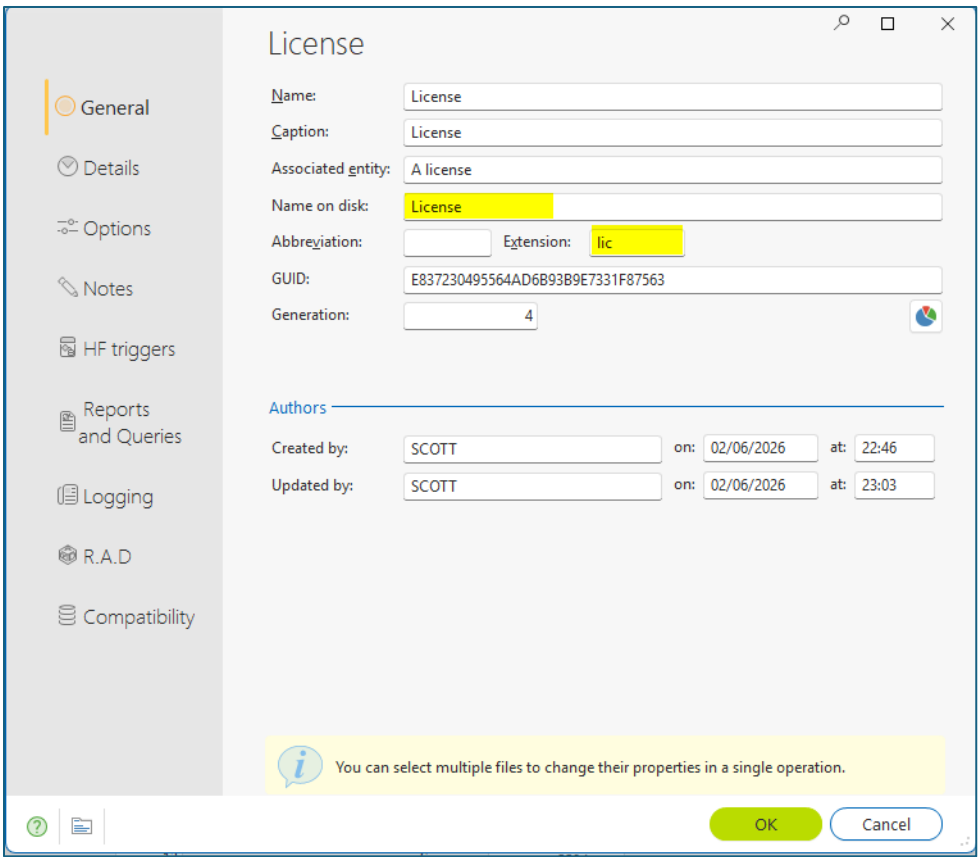
Field Definitions

A Hyperfile database file must be added to each Windev application that LicenseTrak is used. The structure must EXACTLY match this definition, else a runtime error will be generated within your application when a LicenseTrak function call is made. The schema is:

Key	GDPR	Name	Caption	Type	Size
		DeveloperSeed	Developer Seed	Text	50
		AppID	Application ID	Numeric	8
		AppName	Application Name	Text	50
		RegisteredTo	Registered To	Text	50
		LicenseStatus	LicenseStatus	Text	50
		TrialDays	Trial Days	Numeric	2
		TrialRuns	Trial Runs	Numeric	2
		ExecutionCount	ExecutionCount	Numeric	2
		FirstRunDate	FirstRunDate	Date	8
		TamperCount	TamperCount	Numeric	2
		LicensedThroughDate	Licensed Through Date	Date	8
		LicenseHash	License Hash	Text	150
		DateCreated	DateCreated	Date	8

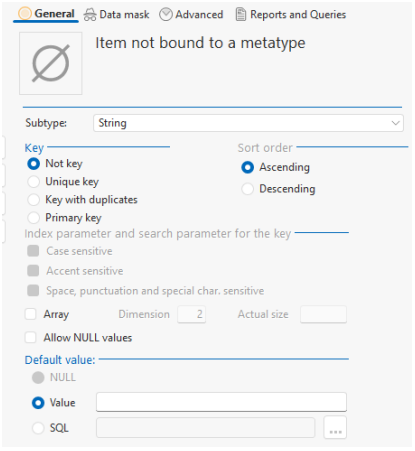
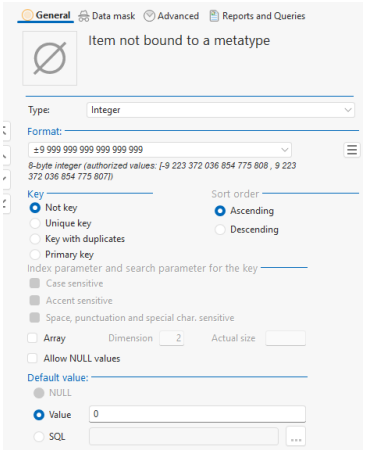
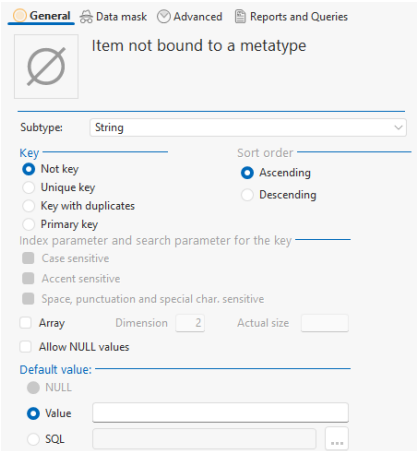
Notice the default Windev-added field has been manually removed from the schema. There are no keys used within this Windev file. It is a single-record file which contains LicenseTrak licensing information. It will use a file extension of (.lic) instead of the standard hyperfile file extension (.fic).

Depicted below is the properties of the License.LIC file; critical configuration settings are highlighted:

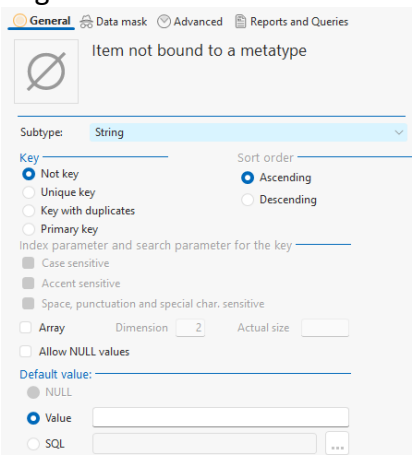


The Runtime Library automatically maintains and updates selected fields as the application executes. Developers should not manually modify the contents of the License.LIC file outside of the LicenseTrak Administration Program or the supplied Runtime Library procedures.

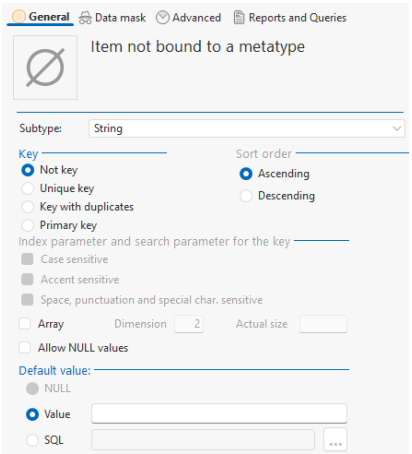
The following pages provide the detailed WinDev field definitions associated with each License.LIC field.

DeveloperSeed 	AppID 	AppName 

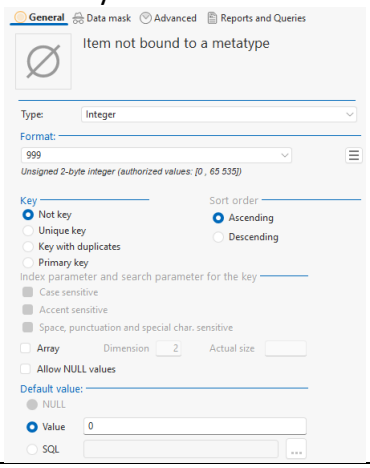
RegisteredTo



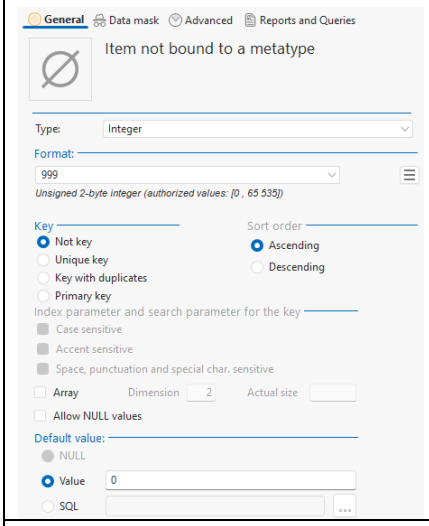
LicenseStatus



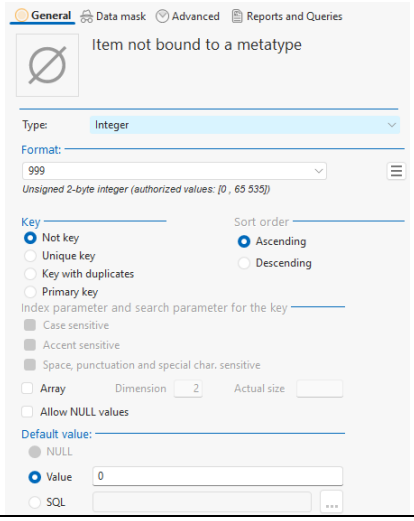
TrialDays



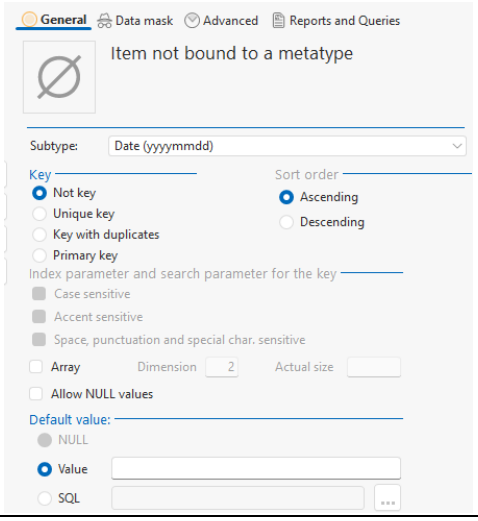
TrialRuns



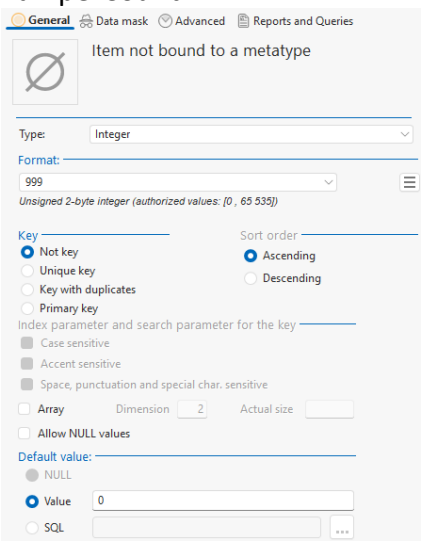
ExecutionCount



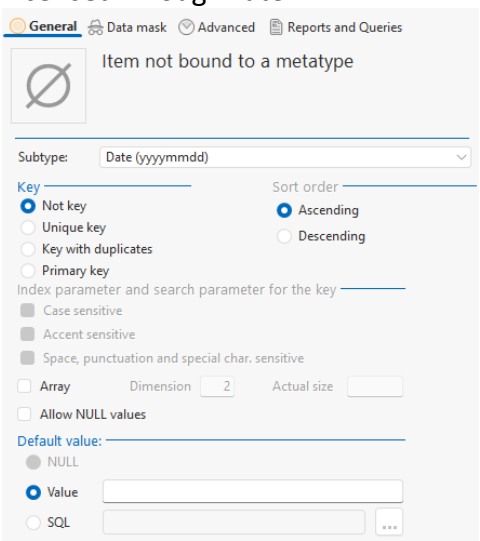
FirstRunDate



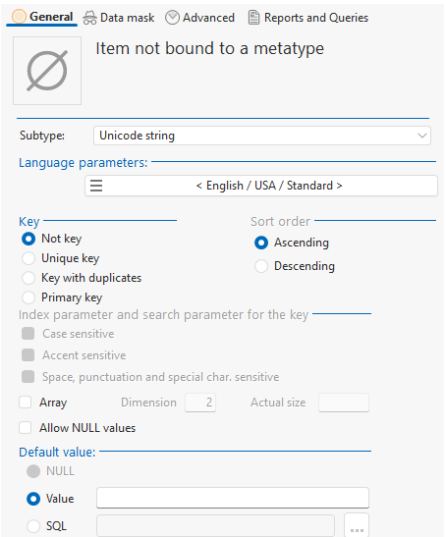
TamperCount

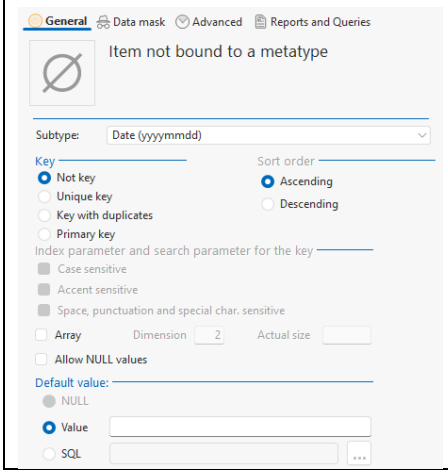


LicensedThroughDate



LicenseHash

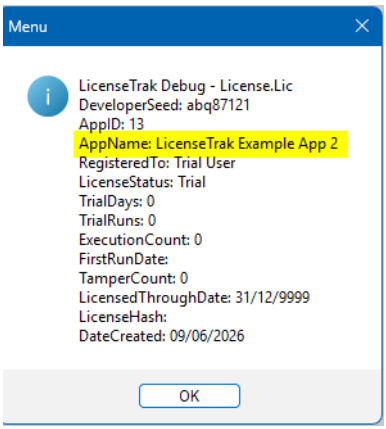


DateCreated 		

Sample License Record

A typical License.LIC file contains a single record representing the current licensing state of the protected application.

For diagnostic and development purposes, the LicenseTrak Runtime Library includes the `LT_DebugDisplayLicense()` procedure, which displays the contents of the active License.LIC file in a human-readable format.



The displayed information includes the Application Name, Registered User, License Status, Trial configuration, execution counters, tamper information, and license dates currently stored within the License.LIC file.

The debug display is intended primarily as a development and integration aid. It allows developers to quickly verify that the Runtime Library is reading the expected License.LIC file and that the licensing information contained within the file matches the values generated by the LicenseTrak Administration Program.

During normal application operation, developers will typically use the individual `LT_` Runtime Library procedures to retrieve only the specific licensing information required by the protected application.

Chapter 12 - Implementing License Strategies

The LicenseTrak Runtime Library provides developers with the flexibility to implement a wide variety of software licensing models.

Earlier chapters described the process of generating Trial, Registered, and Subscription License.LIC files using the LicenseTrak Administration Program. This chapter focuses on the protected application itself and demonstrates how the LicenseTrak Runtime Library procedures may be used to enforce developer-defined licensing policies.

LicenseTrak intentionally does not impose a single licensing strategy. Instead, the Runtime Library provides information about the current licensing state of the application, allowing the developer to determine the most appropriate response for their software product and customer base.

The examples presented in this chapter are intended to illustrate common implementation approaches. Developers are encouraged to adapt these examples to meet the specific requirements of their own applications.

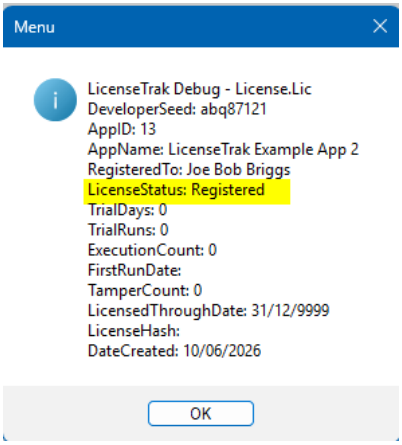
Unlimited Registration Model

The Unlimited Registration Model is the simplest and most common licensing strategy used by traditional desktop applications.

Under this model, prospective customers evaluate the application using a Trial License.LIC file. Once the customer purchases the software, the Trial License.LIC file is replaced with a Registered License.LIC file generated by the LicenseTrak Administration Program.

The protected application uses the LicenseTrak Runtime Library to determine whether the currently installed License.LIC file represents a Trial or Registered installation.

A typical implementation may simply check the current licensing status during application startup and enable all application features when a Registered License.LIC file is present.



This licensing model is well suited for traditional commercial desktop applications where customers purchase a perpetual license and continue using the application without periodic renewal requirements.

Because the License.LIC file is portable, replacement Registered licenses may be generated and redistributed at any time using the LicenseTrak Administration Program if the customer loses the original license file due to hardware

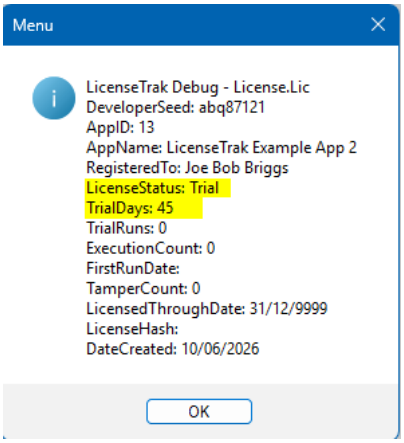
failure or computer replacement.

Trial Day Model

The Trial Run Model limits software evaluation based upon the number of times the application has been executed rather than the number of elapsed calendar days.

This licensing strategy is often useful for applications that may be used infrequently, such as engineering tools, utilities, or specialized business software. By limiting the number of permitted executions instead of the number of days, prospective customers are provided with a predictable number of evaluation opportunities regardless of how much time passes between application launches.

A Trial Run License.LIC file is generated by specifying a Trial Run value within the LicenseTrak Administration Program. The generated value is stored within the License.LIC file and becomes available to the protected application through the LicenseTrak Runtime Library.



During normal application operation, the Runtime Library may be used to retrieve the permitted Trial Run value and the current application Execution Count. The protected application may then compare these values and determine the appropriate course of action based upon the developer's licensing policy.

Typical responses when the permitted Trial Run limit has been reached include:

- Displaying a notification message.
- Requesting software registration.
- Disabling selected application features.
- Preventing further application execution.

The specific enforcement behavior is determined entirely by the developer. LicenseTrak provides the licensing information required to support these decisions while allowing each software publisher to implement the strategy most appropriate for their application.

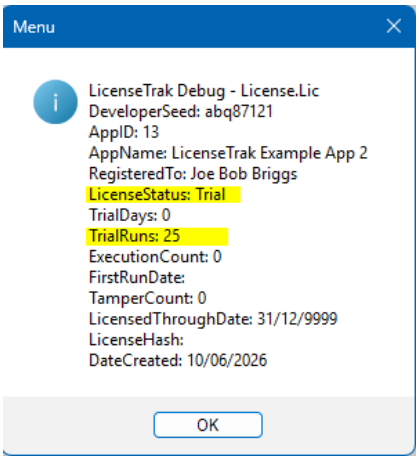
Trial Run Model

The Trial Run Model limits software evaluation based upon the number of times the protected application has been executed rather than the number of elapsed calendar days.

This licensing strategy is particularly well suited for applications that are used occasionally or intermittently. Unlike a

Days-Based Trial, which expires after a predetermined number of calendar days, a Trial Run license allows the prospective customer to evaluate the application for a fixed number of executions regardless of the amount of time between application launches.

To create a Trial Run license, generate a Trial License.LIC file using the LicenseTrak Administration Program and specify the desired Trial Run value.



In the **example** above, the License.LIC file contains a Trial Run limit while the Execution Count indicates the number of times the protected application has been successfully executed.

During application startup, the LicenseTrak Runtime Library may be used to retrieve the Trial Run limit and the current Execution Count. The protected application may then determine whether the permitted number of evaluation runs has been exceeded and respond according to the developer's licensing policy.

Possible responses include:

- Displaying a reminder that the evaluation period is nearing completion.
- Encouraging the user to purchase a Registered license.
- Restricting selected application functionality.
- Preventing further application execution.

The specific enforcement behavior is determined entirely by the developer. LicenseTrak provides the licensing information and Runtime Library procedures necessary to implement a Trial Run licensing strategy while allowing each software publisher to define the most appropriate customer experience for their application.

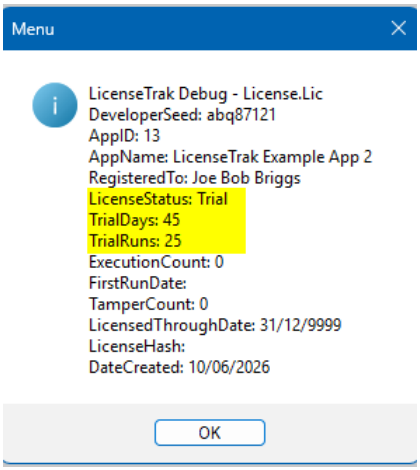
Hybrid Trial Model

The Hybrid Trial Model combines both the Trial Day and Trial Run licensing strategies into a single License.LIC file.

Under this approach, the protected application may be evaluated for a specified number of calendar days and a specified number of application executions. The evaluation period ends when either limit is reached.

This licensing model is particularly effective for commercial desktop applications because it provides prospective customers with a reasonable evaluation opportunity while reducing the likelihood that the Trial period can be artificially extended by infrequent application use.

To create a Hybrid Trial license, generate a Trial License.LIC file using the LicenseTrak Administration Program and specify values for both the **Trial Days** and **Trial Runs** fields.



In the **example** above, the License.LIC file contains both a Trial Day limit and a Trial Run limit. The protected application may use the LicenseTrak Runtime Library to retrieve these values, compare them against the current application state, and determine whether the evaluation period should continue.

A typical implementation permits application execution only while both Trial conditions remain valid. If either the permitted number of Trial Days or the permitted number of Trial Runs has been exceeded, the developer may elect to notify the user, restrict selected features, or terminate the application.

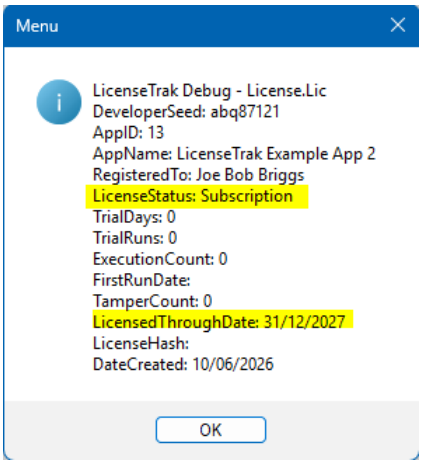
The Hybrid Trial Model offers an excellent balance between customer convenience and software protection and is well suited for many commercial shareware and evaluation applications.

Subscription Model

The Subscription Model permits software use through a developer-defined expiration date stored within the License.LIC file.

Unlike the Unlimited Registration Model, which grants perpetual use of the application, a Subscription License.LIC file contains a **Licensed Through Date** that defines the period during which the application remains licensed.

To create a Subscription license, generate a License.LIC file using the LicenseTrak Administration Program and specify the desired **Licensed Through Date**.



In the **example** above, the License.LIC file contains a populated Licensed Through Date that is used by the protected application to determine whether the subscription period remains valid.

During application startup, the LicenseTrak Runtime Library may be used to retrieve the Licensed Through Date and compare it to the current system date. The protected application may then determine the appropriate response when the subscription period approaches expiration or has expired.

Typical developer-defined responses include:

- Displaying a reminder that the subscription period is nearing expiration.
- Encouraging the customer to renew their subscription.
- Restricting access to selected application features.
- Preventing further application execution after the subscription period has expired.

The specific enforcement behavior is determined entirely by the developer. LicenseTrak provides the Runtime Library procedures and licensing information necessary to implement a subscription-based licensing strategy while allowing each software publisher to define the renewal process and customer experience that best fits their application.

A significant advantage of the Subscription Model is that subscription renewals may be performed simply by generating and distributing an updated License.LIC file containing a new Licensed Through Date. No Internet connection, activation server, or cloud-based licensing infrastructure is required.

Feature Restriction Model

The Feature Restriction Model allows the protected application to selectively enable or disable functionality based upon the information contained within the License.LIC file.

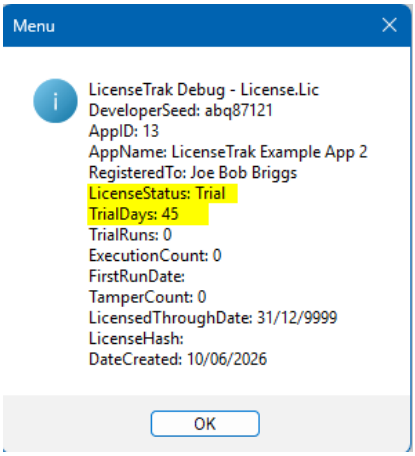
Unlike other licensing strategies that may prevent the application from executing entirely, a Feature Restriction Model permits the developer to create multiple levels of application functionality while allowing all users to continue running the software.

This approach is commonly used in commercial desktop applications that offer both Standard and Professional editions, limited Trial versions, or optional premium features.

Examples of Feature Restriction strategies include:

- Permitting data entry while disabling printing or export functions.
- Limiting the number of records that may be created or stored.
- Restricting advanced reporting or analysis capabilities.
- Disabling selected administrative or configuration features.
- Providing read-only access after a Trial or Subscription period has expired.

The LicenseTrak Runtime Library provides the licensing information required to support these decisions. The protected application may examine the current License Status, Trial limitations, Subscription expiration date, or other License.LIC fields and determine which application features should be made available to the current user.



For **example**, a developer may permit a Trial user to evaluate the application's primary functionality while disabling advanced reporting features. Similarly, a Subscription customer whose Licensed Through Date has expired might be permitted to view existing data while being prevented from adding or modifying records until the subscription is renewed.

The Feature Restriction Model provides software publishers with significant flexibility and encourages a customer-friendly licensing experience. Rather than simply terminating application execution when a licensing condition is encountered, developers may choose to present upgrade opportunities while continuing to provide limited access to the application's existing functionality.

Because the Runtime Library provides the underlying licensing information without imposing a predefined enforcement policy, LicenseTrak may be adapted to support virtually any feature-based licensing strategy required by the protected application.

The licensing models presented in this chapter are intended as examples rather than limitations. The flexibility of the LicenseTrak Runtime Library allows developers to combine and adapt these strategies to create licensing policies that best meet the needs of their applications and customers.

Chapter 13 - Integration Examples

The LicenseTrak Runtime Library is designed to provide licensing information to the protected application while allowing the developer complete flexibility in determining how that information is presented and enforced.

The examples presented in this chapter are intended to illustrate common integration techniques that may enhance the user experience or simplify application administration. These examples are not mandatory components of the

LicenseTrak Runtime Library, but rather suggestions that developers may adapt to meet the requirements of their own software products.

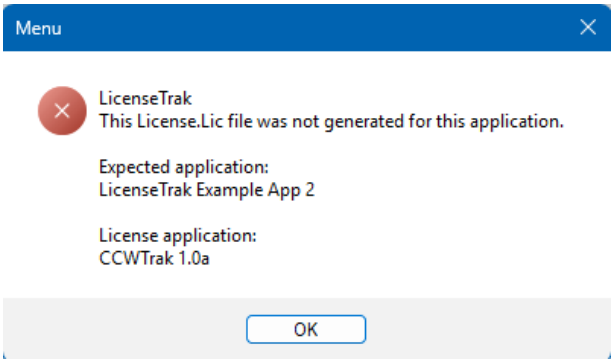
Some applications may choose to display licensing information prominently, while others may use the Runtime Library solely to determine whether selected features should be enabled or disabled. The flexible architecture of LicenseTrak allows each software publisher to implement the licensing strategy and user interface that best fits their application.

Startup Validation

One of the most common integration points for the LicenseTrak Runtime Library is during application startup.

A typical implementation performs LicenseTrak initialization immediately after the application's main window is displayed or during the project's initialization code. Performing the validation process early ensures that licensing information is available before application features are enabled or restricted.

The following **example** demonstrates a simple application startup sequence that initializes the LicenseTrak Runtime Library but experienced a failure (i.e. the License.LIC file was for another LicenseTrak protected application):



When the application starts, the Runtime Library locates and reads the License.LIC file, validates the application identity, and prepares the licensing environment for use by the remaining Runtime Library procedures.

If the deployed License.LIC file does not belong to the protected application, or if another initialization problem is encountered, the Runtime Library may notify the user and allow the protected application to respond according to the developer's preferred licensing policy.

Performing LicenseTrak initialization during application startup helps ensure that licensing information is available throughout the remainder of the application's execution.

Startup Status Display

Many commercial applications provide the user with a brief indication of the current licensing status during application startup. This information may be displayed within a splash screen, a status dialog, or a simple notification message.

While not required, presenting licensing information during application initialization can reassure the user that the application has successfully recognized the installed License.LIC file and may also serve as a reminder when a Trial or Subscription license is nearing expiration.

Typical information that may be presented includes:

- Registered User name.
- Current License Status.
- Trial Days or Trial Runs remaining.
- Subscription expiration date.
- Version or Application Name information.

The LicenseTrak Runtime Library provides procedures that allow the protected application to retrieve this information and present it in a manner consistent with the application's overall user interface design.

For **example**, a Trial version might display a brief message indicating the number of evaluation days remaining, while a Registered installation might simply acknowledge the registered owner before continuing to the main application window. Another popular option is to display registration/trial information within the applications “About” window.

Because the Runtime Library separates license management from application presentation, developers are free to display as much or as little licensing information as they feel is appropriate for their users.

Trial Expiration Handling

A Trial License.LIC file eventually reaches the limits established by the software publisher. This may occur because the permitted number of Trial Days has elapsed, the maximum number of Trial Runs has been reached, or a combination of both conditions within a Hybrid Trial implementation.

When a Trial period expires, the protected application may respond in any manner the developer considers appropriate. LicenseTrak intentionally does not impose a mandatory enforcement policy, allowing each software publisher to design the customer experience that best matches their application's licensing model.

Common approaches include:

- Displaying a Trial expiration notification.
- Encouraging the user to purchase a Registered license.
- Providing a convenient link to the developer's web site or online ordering page.
- Entering a limited functionality or read-only operating mode.
- Preventing further application execution until a valid License.LIC file is installed.

Many developers find that a friendly and informative Trial expiration message encourages registration more effectively than an abrupt program termination. The flexibility of the LicenseTrak Runtime Library allows developers to implement either approach while using the same underlying licensing information.

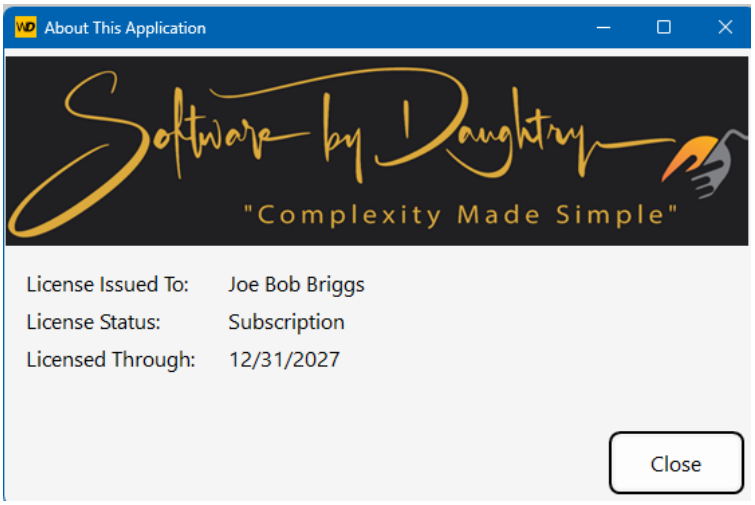
About Box Integration

The application's **About** dialog provides a convenient location to display licensing information obtained from the LicenseTrak Runtime Library.

In addition to presenting the application's version information and publisher details, many developers choose to display selected licensing information to reassure the user that the correct License.LIC file has been installed.

Commonly displayed information includes:

- Registered User name.
- Current License Status.
- Subscription expiration date, if applicable.
- Application version information.



This window's text strings were built like this:

```
End of initialization of WIN_About_This_Application
// Grab LicenseTrak values
sDate is string
sDate = LT_GetLicensedThroughDate("abq87121")
STC_LicensedThrough = DateToString(sDate,maskDateSystem)
STC_LicenseStatus = LT_GetLicenseStatus("abq87121")
STC_RegisteredTo = LT_GetRegisteredTo("abq87121")
```

Displaying licensing information within the About dialog offers several advantages. Users can quickly confirm that the application is operating under the expected license, while technical support personnel can easily verify the installed licensing status when assisting a customer.

The LicenseTrak Runtime Library provides procedures that allow the protected application to retrieve this information and present it in a format that is consistent with the application's existing user interface design. Developers may choose to display all available licensing information or only the values most relevant to their customers.

The **example** shown above displays the Registered User, current License Status, and Licensed Through Date for a Subscription License.LIC file. Similar techniques may be used to display Trial or Registered licensing information as appropriate for the protected application.

Report Header Integration

Many business applications generate printed reports, invoices, summaries, or exported documents that may be distributed outside the organization. Integrating selected licensing information into these reports can provide a subtle but useful indication of application ownership and licensing status.

The LicenseTrak Runtime Library provides procedures that allow the protected application to retrieve information that may be incorporated into report headers or footers during report generation.

Examples of information that developers may choose to display include:

- Registered User name.
- Company or Organization name.
- Application version information.
- Subscription expiration date.

A common implementation is to display the Registered User information in a small footer or header line, such as:

Licensed to: Joe Bob Briggs

For subscription-based applications, some developers may also choose to include the current Licensed Through Date for internal administrative or audit purposes.

Including licensing information within generated reports can provide several practical benefits. It helps confirm that the correct License.LIC file is installed, simplifies technical support, and may discourage the unauthorized distribution of application-generated output by clearly identifying the licensed owner.

The exact format and placement of licensing information within printed reports is entirely at the discretion of the developer. LicenseTrak simply provides the Runtime Library procedures necessary to retrieve the required information and integrate it into the application's existing reporting framework.

Status Bar Integration

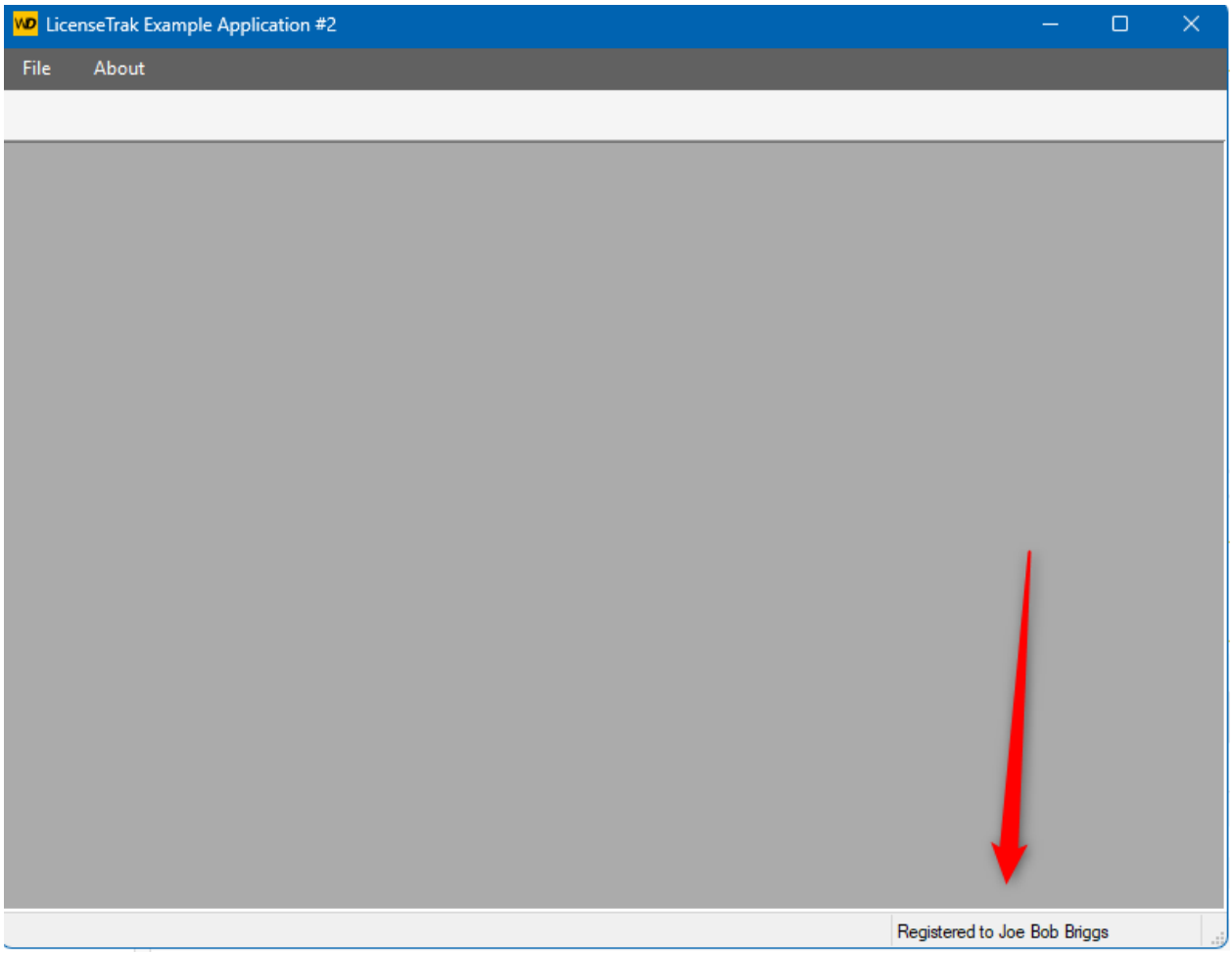
The application's status bar provides another convenient location to display selected licensing information obtained from the LicenseTrak Runtime Library.

Unlike a startup notification or About dialog, a status bar allows licensing information to remain visible throughout normal application operation without interrupting the user's workflow. This approach can provide a subtle reminder of the current licensing status while consuming very little screen space.

The LicenseTrak Runtime Library procedures may be used to populate one or more status bar cells with information such as:

- Registered User name.
- Current License Status.
- Trial Days remaining.
- Subscription expiration date.

The following **example** demonstrates a simple implementation that retrieves the Registered User information from the installed License.LIC file and displays it within the application's status bar.



The Windev code used to create the status bar message:

```
End of initialization of WIN_MENU    If Error: by program    When Exception: by program
// Check if license.lic file belongs to this application
LT_Initialize("abq87121", "LicenseTrak Example App 2")

// Activate the window's status bar
WIN_MENU.StatusBar = True


// Build the Status Bar string
sMessage is string
sMessage = "Registered to " + LT_GetRegisteredTo("abq87121")

// Add a new cell "MyCell"
sMyCell is string
StatusBarAddCell("sMyCell",200)

// Display a message in the newly created cell
Message("sMyCell", sMessage)
```

In this **example**, the Runtime Library retrieves the Registered User value from the License.LIC file using the **`LT_GetRegisteredTo()`** procedure. The application then formats the information and displays it within a dedicated status bar cell.

Developers may adapt this approach to display any information available through the LicenseTrak Runtime Library. For **example**, a Trial version might display the number of evaluation days remaining, while a Subscription installation

could display the current Licensed Through Date.

Status Bar integration is entirely optional, but it provides a simple and unobtrusive method for presenting licensing information to the user during normal application operation.

Feature Restriction Features

One of the greatest advantages of the LicenseTrak Runtime Library is that it does not impose a predefined licensing enforcement policy. Instead, the Runtime Library provides the protected application with accurate licensing information while allowing the developer to determine how that information should be used.

Many software publishers choose to implement a Feature Restriction Model rather than completely preventing application execution when a licensing condition is encountered. This approach allows prospective customers to continue evaluating the application while encouraging registration or subscription renewal through the selective limitation of functionality.

Common Feature Restriction strategies include:

- Disabling advanced reporting or analysis functions.
- Restricting data export capabilities.
- Preventing printing while allowing on-screen viewing.
- Limiting the number of records that may be created or maintained.
- Disabling administrative or configuration features.
- Providing read-only access after a Trial or Subscription period has expired.

The LicenseTrak Runtime Library procedures may be used to determine the current License Status, Trial limitations, or Subscription expiration date. The protected application may then enable or disable individual menu options, toolbar buttons, or program features based upon the developer's preferred licensing strategy.

For **example**, a Trial version might permit users to browse and search existing records while disabling export or printing functions. Similarly, a Subscription customer whose Licensed Through Date has expired might be allowed to view historical data while being prevented from adding or modifying records until the subscription is renewed.

This approach often provides a more customer-friendly experience than abruptly terminating the application, while still encouraging software registration or subscription renewal.

Because the Runtime Library separates licensing information from enforcement decisions, developers may combine and adapt these examples to create feature-based licensing strategies that best fit the requirements of their own applications and customer communities.

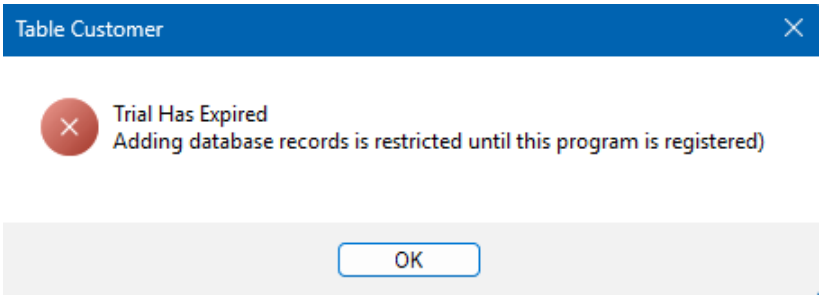
Read-Only Database Mode

One customer-friendly feature restriction strategy is to allow users to continue viewing existing data while preventing database changes when a licensing condition is not satisfied.

This approach is often preferable to immediately terminating the application because it preserves access to existing information while still encouraging registration or subscription renewal.

A Read-Only Database Mode may be implemented when:

- A Trial license has expired.
- A Subscription license has expired.
- A Trial Run limit has been exceeded.
- The developer wishes to allow data review but prevent continued data entry.



And the code used to generate the popup/restriction:

```
Click Btn_New        If Error: by program        When Exception: by program
IF LT_IsTrial("abq87121","Trial") THEN
    Error("Trial Has Expired", "Adding database records is restricted until this program is registered)")

ELSE

    // Create a new record
    Customer.Reset()

    // Open the form of LT_Customer file in creation mode
    IF WIN_Form_LT_Customer.Open() THEN

        // Refresh the table
        TABLE_LT_Customer.Display(taCurrentFirst)

    END
END
```

In the **example** above, the application continues to display existing customer records, but adding new database records is restricted because the Trial license is no longer valid.

Rather than closing the application, the developer has chosen to display an informational message and prevent the requested database update.

Common Read-Only Mode restrictions include:

- Disabling New buttons.
- Disabling Edit or Modify buttons.
- Disabling Delete buttons.
- Preventing direct table updates.
- Allowing Browse and Search operations to continue.

The LicenseTrak Runtime Library provides the licensing information needed to determine whether database updates should be permitted. The application then uses that information to decide whether to allow or block the requested operation.

This strategy provides a balanced customer experience by protecting the developer's licensing interests while still allowing users to access their existing data.

Watermarked Report Output

Many developers choose to permit Trial users to generate reports while applying subtle restrictions that encourage software registration.

Rather than disabling printing entirely, the protected application may use the LicenseTrak Runtime Library to determine the current licensing status and dynamically alter the appearance of printed output when a Trial License.LIC file is detected.

A common approach is to apply a watermark to all printed reports generated by the Trial version of the application.

```
Click BTN_Print    If Error: by program    When Exception: by program
// Declare the watermark variable
IF LT_IsTrial("abq87121","Trial") THEN
    // Add a watermark to the report
    MyWatermark is Watermark

    // Define the text and formatting
    MyWatermark.Text        = "TRIAL - PLEASE REGISTER"
    MyWatermark.Font.Name   = "Arial"
    MyWatermark.Font.Size   = 48
    MyWatermark.Font.Color  = LightGray

    // Set positioning (e.g., centered and in the background)
    MyWatermark.Position    = iCenterH + iCenterV

    // Apply watermark settings for printing and/or duplicates
    iParameterWatermark(iWatermarkPrinting + iWatermarkDuplicate, MyWatermark)
END

// Start printing the report (replace RPT_MyReport with your actual report name)
iPreview(ipvZoomPage,"Customer List")
iPrintReport(RPT_List_LT_Customer)
```

which results in this output:

RPT_List_LT_Customer

06/10/2026

Cust	First Name	Last Name	Address Line1	Address Line2	City	State	Zip Code	Phone Number	Email Address	Cust	Country
1	John	Brown1	101 Brown Street		Albuquerque	NM	10001	555-101-3700			
2	Mary	Miller2	102 Miller Street		Tampa	FL	10002	555-102-7400			
3	Robert	Carter3	103 Carter Street		Denver	CO	10003	555-103-1110			
4	Linda	Adams4	104 Adams Street		Phoenix	AZ	10004	555-104-1480			
5	Michael	Wilson5	105 Wilson Street		Dallas	TX	10005	555-105-1850			
6	Karen	Reed6	106 Reed Street		Orlando	FL	10006	555-106-2220			
7	David	Morgan7	107 Morgan Street		Nashville	TN	10007	555-107-2590			
8	Susan	Bennett8	108 Bennett Street		Seattle	WA	10008	555-108-2960			
9	James	Collins9	109 Collins Street		San Diego	CA	10009	555-109-3330			
10	Patricia	Daughtry10	110 Daughtry Street		Boston	MA	10010	555-110-3700			
11	William	Parker11	111 Parker Street		Austin	TX	10011	555-111-4070			
12	Barbara	Harris12	112 Harris Street		Reno	NV	10012	555-112-4440			
13	Richard	Turner13	113 Turner Street		Atlanta	GA	10013	555-113-4810			
14	Jennifer	Walker14	114 Walker Street		Chicago	IL	10014	555-114-5180			
15	Thomas	Cooper15	115 Cooper Street		Portland	OR	10015	555-115-5550			
16	Angela	Mitchell16	116 Mitchell Street		Tulsa	OK	10016	555-116-5920			
17	Charles	Murphy17	117 Murphy Street		Boise	ID	10017	555-117-6290			
18	Nancy	Bailey18	118 Bailey Street		Raleigh	NC	10018	555-118-6660			
19	Daniel	Foster19	119 Foster Street		Fresno	CA	10019	555-119-7030			
20	Lisa	Hayes20	120 Hayes Street		Omaha	NE	10020	555-120-7400			
21	John	Brown21	121 Brown Street		Albuquerque	NM	10021	555-121-7770			

1/...

In the **example** above, the application examines the current License Status before generating the report. When a Trial License.LIC file is present, a watermark containing the message **"TRIAL - PLEASE REGISTER"** is automatically applied to the report output.

This approach offers several advantages:

- Prospective customers retain the ability to fully evaluate the application's reporting features.
- Printed output clearly identifies the application as operating under a Trial license.
- The need to completely disable printing functionality is eliminated.
- The transition from Trial to Registered operation occurs automatically when a valid Registered License.LIC file is installed.

The LicenseTrak Runtime Library provides the licensing information necessary to support this type of implementation while allowing each developer to determine the most appropriate presentation for their application. Other developers may elect to disable printing entirely, limit the number of printed pages, or apply alternative visual indicators to Trial-generated output.

The **example** shown here illustrates one possible implementation. Because LicenseTrak separates license management from application behavior, developers are free to adapt the concept to match the requirements of their own software products.

Restricting Data Export

Many WinDev applications allow users to export data to external file formats such as CSV, Excel, JSON, or XML.

WinDev provides built-in WLanguage functions for these operations, including `HExportCSV`, `HExportXLS`, `HExportJSON`, and `HExportXML`.

Because exported data can be used outside the protected application, many developers choose to restrict export capabilities when the application is operating under a Trial license.

A typical implementation places an Export button or menu option on the appropriate window. When the user requests an export, the application first checks the current LicenseTrak licensing status before executing the export operation.

For **example:**

```
IF LT_IsTrial("abq87121", "Trial") THEN
   Error("Data export is not available in the Trial version.")
   RETURN
END

// Example export operation
HExportCSV(Customer, "C:\Exports\Customer.csv", "")
Info("The export file has been created.")
```

In this **example**, the application prevents Trial users from exporting customer data. If the installed License.LIC file is not a Trial license, the export operation continues normally.

Developers may adapt this approach to match their own licensing policies. Possible strategies include:

- Disabling export completely during Trial operation.
- Allowing export only for Registered or active Subscription licenses.
- Limiting the number of records that may be exported.
- Exporting sample data only.
- Applying watermarks or notices to exported PDF output.
- Displaying a registration message when export is attempted.

Restricting data export can be especially useful when the protected application contains valuable business records, customer lists, reports, or other information that may be useful outside the application.

As with other feature restrictions, LicenseTrak provides the licensing information. The protected application determines whether export functionality should be enabled, limited, or disabled.

Chapter 14 - Runtime API Reference

LT_Initialize()

Purpose

The LT_Initialize() procedure initializes the LicenseTrak Runtime Library and prepares the protected application for subsequent licensing operations.

During initialization, the Runtime Library locates the `License.LIC` file, reads the licensing information it contains, validates that the license belongs to the current application, and loads the required values into memory for use by the remaining Runtime Library procedures.

Syntax

LT_Initialize(psDeveloperSeed, psApplicationName)

Parameters

psDeveloperSeed
Developer-defined seed value used during license generation and validation.

psApplicationName
The protected application's name and version string. This value is compared against the Application Name stored within the `License.LIC` file to ensure that the license was generated for the correct application.

Returns

This procedure does not return a value.

If initialization fails, the Runtime Library displays an informational message describing the detected condition and allows the protected application to respond according to the developer's preferred licensing policy.

Example

// Initialize the LicenseTrak Runtime Library
LT_Initialize("abq87121", "LicenseTrak **Example** App 2")

Remarks

The LT_Initialize() procedure should normally be called during application startup, such as within the Main window's initialization code or the project's initialization event.

This procedure should be executed before calling any other LicenseTrak Runtime Library procedures. Successful initialization ensures that the licensing information contained within the `License.LIC` file is available to all subsequent `LT_` functions.

LT_IsTrial()

Purpose

The LT_IsTrial() procedure determines whether the currently installed License.LIC file represents a Trial license.

Syntax

LT_IsTrial(psTrialStatus)

Parameters

psTrialStatus
   The localized text value representing the Trial licensing state.

Returns

Boolean

True
   The installed License.LIC file is a Trial license.

False
   The installed License.LIC file is not a Trial license.

Example

```
IF LT_IsTrial("Trial") THEN
   Info("You are currently running the Trial version.")
END
```

Remarks

The LT_IsTrial() procedure simply reports the current License Status stored within the License.LIC file. It does not determine whether the Trial period remains valid. Use LT_IsTrialDaysValid() or LT_IsTrialRunsValid() to determine whether a Trial has expired.

LT_ValidateLicenseHash()

Purpose:

The LT_ValidateLicenseHash() procedure verifies the integrity of the currently installed License.LIC file by comparing the stored License Hash value against the expected value generated by the LicenseTrak Runtime Library.

This validation process helps detect situations where the License.LIC file has been manually modified or replaced with an unauthorized version.

Syntax:

LT_ValidateLicenseHash(psDeveloperSeed)

Parameters:

psDeveloperSeed

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the License.LIC file was originally created.

Returns:

Boolean

True

: The License Hash value is valid and the License.LIC file appears to be authentic.

False

: The License Hash validation failed, indicating that the License.LIC file may have been modified or does not belong to the protected application.

Example:

```
// Verify the integrity of the installed License.LIC file
IF NOT LT_ValidateLicenseHash("abq87121") THEN
   Error("The License.LIC file failed validation.")
   RETURN
END
```

Remarks:
The LT_ValidateLicenseHash() procedure is normally called internally by LT_Initialize() during application startup. Under typical circumstances, developers will not need to invoke this procedure directly.

However, the procedure is available should the protected application require additional integrity checks during

execution or before performing sensitive licensing operations.

If License Hash validation fails, the developer may elect to display a warning message, increment the application's tamper counter, restrict selected functionality, or terminate program execution according to the desired licensing policy.

LT_IsRegistered()

Purpose:

The LT_IsRegistered() procedure determines whether the currently installed License.LIC file represents a Registered license.

This procedure is commonly used to enable full application functionality for customers who have purchased the software.

Syntax:

```
LT_IsRegistered(psRegisteredStatus)
```

Parameters:

`psRegisteredStatus`

: The localized text value representing the Registered licensing state. For example, an English application might pass `"Registered"` while a French application would pass the appropriate translated value.

Returns:

Boolean

`True`

: The installed `License.LIC` file is a Registered license.

`False`

: The installed `License.LIC` file is not a Registered license.

Example:

```
// Enable full functionality for registered users
IF LT_IsRegistered("Registered") THEN
    MENU_PremiumFeatures..State = Active
END
```

Remarks:

The LT_IsRegistered() procedure reports the current License Status value stored within the License.LIC file.

Registered licenses are typically used for customers who have purchased a perpetual license. Applications that

support renewable licensing models should also evaluate Subscription licenses and Licensed Through Date values as appropriate.

LT_IsSubscription()

Purpose:

The LT_IsSubscription() procedure determines whether the currently installed License.LIC file represents a Subscription license.

This procedure is commonly used by the protected application to determine whether subscription-specific processing or validation should be performed.

Syntax:

`LT_IsSubscription(psSubscriptionStatus)`

Parameters:

`psSubscriptionStatus`

: The localized text value representing the Subscription licensing state. For example, an English application might pass `"Subscription"` while a French application would pass the appropriate translated value.

Returns:

Boolean

`True`

: The installed `License.LIC` file is a Subscription license.

`False`

: The installed `License.LIC` file is not a Subscription license.

Example:

```
// Determine whether the application is operating
// under a Subscription license
IF LT_IsSubscription("Subscription") THEN
    Info("Subscription license detected.")
END
```

Remarks:

The LT_IsSubscription() procedure simply reports the current License Status value stored within the License.LIC file. It does not determine whether the subscription period remains valid.

Applications that support renewable licensing models should use `LT_IsLicensedThroughDateValid()` to determine whether the current subscription has expired.

LT_IsLocked()

Purpose:

The LT_IsLocked() procedure determines whether the currently installed License.LIC file represents a Locked license.

This procedure allows the protected application to determine whether a License.LIC file has been intentionally disabled by the software publisher.

Syntax:

```
LT_IsLocked(psLockedStatus)
```

Parameters:

`psLockedStatus`

: The localized text value representing the Locked licensing state. For example, an English application might pass `"Locked"` while a French application would pass the appropriate translated value.

Returns:

Boolean

`True`

: The installed `License.LIC` file is a Locked license.

`False`

: The installed `License.LIC` file is not a Locked license.

Example:

```
// Determine whether the installed license
// has been marked as Locked
IF LT_IsLocked("Locked") THEN
    Error("This License.LIC file has been disabled.")
    RETURN
END
```

Remarks:

The LT_IsLocked() procedure simply reports the current License Status value stored within the License.LIC file. It does not prescribe how the protected application should respond when a Locked license is encountered.

Depending upon the developer's preferred licensing strategy, a Locked license may result in an informational message being displayed, selected application features being restricted, or application execution being terminated.

LT_IsApplicationRestricted()

Purpose:
The LT_IsApplicationRestricted() procedure determines whether the protected application should operate in a restricted mode based upon the current licensing information stored within the License.LIC file.
This procedure provides developers with a convenient method of implementing feature restrictions without requiring each application module to independently evaluate multiple licensing conditions.

Syntax:

LT_IsApplicationRestricted()

Parameters:

None.

Returns:

Boolean

`True`

: The protected application should operate in a restricted mode. The developer may use this condition to disable selected features, prevent database updates, restrict printing, or block data exports.

`False`

: The current licensing information does not require the application to operate in a restricted mode.

Example:

```
// Determine whether the application
// should operate with restricted features
IF LT_IsApplicationRestricted() THEN
   BTN_Export..State = Grayed
   BTN_AddRecord..State = Grayed
END
```

Remarks:

The LT_IsApplicationRestricted() procedure is intended to simplify the implementation of feature-based licensing strategies. Rather than repeatedly evaluating individual licensing conditions throughout the application, developers may centralize their licensing policy within this procedure and respond to a single Boolean result.

Typical conditions that may cause an application to operate in a restricted mode include an expired Trial license, an expired Subscription license, or the presence of a Locked license. The specific conditions that define a "restricted" application state are determined entirely by the developer.

LT_IsEnglish()

Purpose:
The LT_IsEnglish() procedure determines whether the currently installed License.LIC file specifies English as the application's active language.
This procedure allows protected applications to enable English-specific text, messages, reports, or other language-dependent processing.
Syntax:
LT_IsEnglish(psDeveloperSeed)
Parameters:
psDeveloperSeed
: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the License.LIC file was originally created.
Returns:
Returns **True** if the license language is English; otherwise returns **False**.
Example:
```
IF LT_IsEnglish("abq87121") THEN
  Info("Welcome")
END
```
Remarks:
The LT_IsEnglish() procedure examines the language value stored within the installed License.LIC file.
This procedure is intended to simplify multilingual application development by eliminating the need to compare language strings throughout an application. Instead, developers can determine the active language using a simple Boolean test.

LT_IsFrench()

Purpose:
The LT_IsFrench() procedure determines whether the currently installed License.LIC file specifies French as the application's active language.
This procedure allows protected applications to display French text, messages, reports, and other localized resources.
Syntax:
LT_IsFrench(psDeveloperSeed)
Parameters:
psDeveloperSeed
: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the License.LIC file was originally created.
Returns:
Returns **True** if the license language is French; otherwise returns **False**.
Example:
```
IF LT_IsFrench("abq87121") THEN
  Info("Bienvenue")
END
```
Remarks:
The LT_IsFrench() procedure provides a convenient method of determining whether the application should operate using French resources.
Developers may use this procedure to select localized text, reports, menus, dialogs, or other language-specific

functionality without manually examining the language field contained within the license.

LT_IsSpanish()

Purpose:
The LT_IsSpanish() procedure determines whether the currently installed License.LIC file specifies Spanish as the application's active language.
This procedure allows protected applications to display Spanish text, reports, menus, and other localized resources.
Syntax:
LT_IsSpanish(psDeveloperSeed)
Parameters:
psDeveloperSeed
: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the License.LIC file was originally created.
Returns:
Returns **True** if the license language is Spanish; otherwise returns **False**.
Example:

```
IF LT_IsSpanish("abq87121") THEN
  Info("Bienvenido")
END
```

Remarks:
The LT_IsSpanish() procedure provides a simple Boolean test for determining whether the application should operate using Spanish resources.
Although LicenseTrak currently includes English and French language support within the License Manager, this procedure allows developers to build multilingual applications that include additional languages such as Spanish.

LT_IsTrialDaysValid()

Purpose:

The `LT_IsTrialDaysValid()` procedure determines whether the number of elapsed Trial Days remains within the limit defined by the currently installed `License.LIC` file.

This procedure is commonly used by the protected application to determine whether a Trial license should continue to permit normal application operation.

Syntax:

```
LT_IsTrialDaysValid()
```

Parameters:

None.

Returns:

Boolean

`True`

: The Trial Day limit has not been exceeded and the Trial license remains valid.

`False`

: The permitted number of Trial Days has elapsed and the Trial license should be considered expired.

Example:

```
// Verify that the Trial period remains valid
IF NOT LT_IsTrialDaysValid() THEN
    Error("The Trial period has expired.")
    RETURN
END
```

Remarks:

The `LT_IsTrialDaysValid()` procedure compares the current system date against the Trial Day information stored within the `License.LIC` file. This procedure should normally be used in conjunction with `LT_IsTrial()` to determine whether Trial-specific restrictions should be applied.

Developers may respond to an expired Trial period in any manner appropriate for their application. Common responses include displaying a registration reminder, enabling a read-only operating mode, restricting selected application features, or terminating application execution.

LT_IsTrialRunsValid()

Purpose:

The `LT_IsTrialRunsValid()` procedure determines whether the number of permitted Trial Runs defined within the currently installed `License.LIC` file has been exceeded.

This procedure is commonly used by the protected application to determine whether a Trial license based upon application executions should continue to permit normal operation.

Syntax:

`LT_IsTrialRunsValid()`

Parameters:

None.

Returns:

Boolean

`True`

: The Trial Run limit has not been exceeded and the Trial license remains valid.

`False`

: The permitted number of Trial Runs has been reached or exceeded, and the Trial license should be considered expired.

Example:

```
// Verify that the Trial Run limit has not been exceeded
IF NOT LT_IsTrialRunsValid() THEN
    Error("The Trial Run limit has been exceeded.")
    RETURN
END
```

Remarks:

The `LT_IsTrialRunsValid()` procedure compares the current Execution Count maintained by the LicenseTrak Runtime Library against the Trial Run limit stored within the `License.LIC` file.

This procedure is typically used in conjunction with `LT_IsTrial()` to implement execution-based or Hybrid Trial licensing models.

Developers may respond to an expired Trial Run limit in any manner appropriate for their application. Common responses include displaying a registration reminder, enabling a read-only operating mode, restricting selected application features, or preventing further application execution.

LT_IsLicensedThroughDateValid()

Purpose:

The `LT_IsLicensedThroughDateValid()` procedure determines whether the current system date is within the Licensed Through Date stored in the currently installed `License.LIC` file.

This procedure is commonly used by applications that implement Subscription licensing or other time-limited licensing models.

Syntax:

```
LT_IsLicensedThroughDateValid()
```

Parameters:

None.

Returns:

Boolean

`True`

: The current system date is on or before the Licensed Through Date stored within the `License.LIC` file.

`False`

: The current system date is after the Licensed Through Date, indicating that the license should be considered expired.

Example:

```
// Verify that the subscription period remains valid
IF NOT LT_IsLicensedThroughDateValid() THEN
    Error("This subscription has expired.")
    RETURN
END
```

Remarks:

The `LT_IsLicensedThroughDateValid()` procedure compares the current system date to the Licensed Through Date stored within the `License.LIC` file.

This procedure is typically used in conjunction with `LT_IsSubscription()` when implementing Subscription licensing models.

Developers may respond to an expired Licensed Through Date in any manner appropriate for their application. Common responses include displaying a renewal reminder, enabling read-only operation, restricting selected features, or preventing further application execution.

LT_DebugDisplayLicense()

Purpose:

The `LT_DebugDisplayLicense()` procedure displays the contents of the currently installed `License.LIC` file in a simple popup window.

This procedure is intended primarily as a development and diagnostic aid, allowing programmers to quickly verify that the Runtime Library has correctly loaded and interpreted the licensing information.

Syntax:

```
LT_DebugDisplayLicense(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

This procedure does not return a value.

Example:

```
// Display the contents of the installed
// License.LIC file for diagnostic purposes
```

```
LT_DebugDisplayLicense("abq87121")
```

Remarks:

The `LT_DebugDisplayLicense()` procedure is intended for development, testing, and troubleshooting. It provides a convenient method of verifying the contents of the installed `License.LIC` file without requiring the programmer to examine the underlying HFSQL data file directly.

The popup window displays key licensing information, including the Application Name, Registered User, License Status, Trial limits, Subscription expiration date, and other values maintained by the Runtime Library.

Many of the examples presented throughout this User Manual make use of `LT_DebugDisplayLicense()` to illustrate how different licensing strategies are represented within the `License.LIC` file.

LT_GetRegisteredTo()

Purpose:

The `LT_GetRegisteredTo()` procedure retrieves the Registered To value stored within the currently installed `License.LIC` file.

This procedure is commonly used to display customer registration information within the protected application, such as in an About dialog, status bar, startup message, or printed report.

Syntax:

```
LT_GetRegisteredTo(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

String

: Returns the Registered To value stored within the `License.LIC` file. If the license information cannot be retrieved, an empty string may be returned.

Example:

```
// Display the name associated with the
// installed License.LIC file
Info("Registered to: " + _
     LT_GetRegisteredTo("abq87121"))
```

Remarks:

The `LT_GetRegisteredTo()` procedure simply retrieves the Registered To field from the installed `License.LIC` file. It does not perform any license validation or determine whether the current license remains valid.

This procedure is frequently used in conjunction with application About dialogs, status bars, startup notifications, and report headers to display ownership information to the user. Several examples of these integration techniques are demonstrated in Chapter 13.

LT_GetLicenseStatus()

Purpose:

The `LT_GetLicenseStatus()` procedure retrieves the current License Status value stored within the installed `License.LIC` file.

This procedure allows the protected application to determine and display the current licensing state, such as Trial, Registered, Subscription, or Locked.

Syntax:

```
LT_GetLicenseStatus(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

String

: Returns the current License Status value stored within the `License.LIC` file. Typical values include `"Trial"`, `"Registered"`, `"Subscription"`, and `"Locked"` (or their localized equivalents).

Example:

```
// Display the current licensing state
Info("License Status: " + _
     LT_GetLicenseStatus("abq87121"))
```

Remarks:

The `LT_GetLicenseStatus()` procedure simply retrieves the License Status field from the installed `License.LIC` file. It does not determine whether a Trial or Subscription license remains valid.

Developers should use the appropriate Runtime Library validation procedures, such as `LT_IsTrialDaysValid()`, `LT_IsTrialRunsValid()`, or `LT_IsLicensedThroughDateValid()`, when determining whether a time-limited license should continue to permit normal application operation.

The LT_GetLicenseStatus() procedure is frequently used to populate About dialogs, status bars, startup notifications, and diagnostic displays. Examples of these integration techniques are presented in Chapters 12 and 13.

LT_GetTrialDays()

Purpose:

The LT_GetTrialDays() procedure retrieves the Trial Day limit stored within the currently installed License.LIC file.

This procedure allows the protected application to determine the number of calendar days allocated for software evaluation and may be used to display Trial information to the user or implement developer-defined licensing policies.

Syntax:

```
LT_GetTrialDays(psDeveloperSeed)
```

Parameters:

psDeveloperSeed

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the License.LIC file was originally created.

Returns:

Integer

: Returns the Trial Day limit stored within the installed License.LIC file. A value of zero typically indicates that a Trial Day limit has not been defined.

Example:

```
// Display the configured Trial Day limit
Info("Trial Days: " + _
     LT_GetTrialDays("abq87121"))
```

Remarks:

The LT_GetTrialDays() procedure simply retrieves the Trial Day value stored within the License.LIC file. It does not determine whether the Trial period remains valid or calculate the number of Trial Days remaining.

To determine whether a Trial license has expired, use the LT_IsTrialDaysValid() procedure. Developers wishing to display the number of remaining Trial Days may calculate that value using the Trial Day limit in conjunction with the application's recorded First Run Date.

The LT_GetTrialDays() procedure is commonly used within startup messages, About dialogs, status bars, and Trial reminder notifications.

LT_GetTrialRuns()

Purpose:

The `LT_GetTrialRuns()` procedure retrieves the Trial Run limit stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine the number of application executions allocated for software evaluation.

Syntax:

```
LT_GetTrialRuns(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Integer

: Returns the Trial Run limit stored within the installed `License.LIC` file. A value of zero typically indicates that a Trial Run limit has not been defined.

Example:

```
// Display the configured Trial Run limit
Info("Trial Runs: " + _
     LT_GetTrialRuns("abq87121"))
```

Remarks:

The `LT_GetTrialRuns()` procedure simply retrieves the Trial Run value stored within the `License.LIC` file. It does not determine whether the Trial Run limit has been exceeded.

To determine whether a Trial Run license remains valid, use the `LT_IsTrialRunsValid()` procedure.

Applications that implement execution-based Trial licensing typically use `LT_GetTrialRuns()` together with `LT_GetExecutionCount()` to display Trial usage information to the user.

LT_GetExecutionCount()

Purpose:

The `LT_GetExecutionCount()` procedure retrieves the current Execution Count value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine how many times the application has been successfully executed and is commonly used when implementing Trial Run or Hybrid Trial licensing models.

Syntax:

```
LT_GetExecutionCount(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Integer

: Returns the current Execution Count value stored within the installed `License.LIC` file. This value is automatically maintained by the LicenseTrak Runtime Library as the application is executed.

Example:

```
// Display the current application
// Execution Count
Info("Execution Count: " + _
     LT_GetExecutionCount("abq87121"))
```

Remarks:

The `LT_GetExecutionCount()` procedure simply retrieves the current Execution Count value stored within the `License.LIC` file. It does not determine whether the permitted Trial Run limit has been exceeded.

Applications that implement Trial Run or Hybrid Trial licensing models typically use `LT_GetExecutionCount()` together with `LT_GetTrialRuns()` to display Trial usage information or to implement developer-defined licensing policies.

For example, an application may choose to display a message similar to:

```
You have used 12 of your 25 Trial Runs.
```

To determine whether the Trial Run limit remains valid, use the `LT_IsTrialRunsValid()` procedure.

LT_GetFirstRunDate()

Purpose:

The `LT_GetFirstRunDate()` procedure retrieves the First Run Date value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine when the application was first executed and is commonly used when implementing Trial Day or Hybrid Trial licensing models.

Syntax:

```
LT_GetFirstRunDate(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Date

: Returns the First Run Date value stored within the installed `License.LIC` file.

Example:

```
// Display the application's First Run Date
Info("First Run Date: " + _
     DateToString(LT_GetFirstRunDate("abq87121")))
```

Remarks:

The `LT_GetFirstRunDate()` procedure retrieves the date on which the protected application was first executed.

Applications that implement Trial Day or Hybrid Trial licensing models may use this value together with `LT_GetTrialDays()` to calculate and display Trial usage information.

To determine whether the Trial Day limit remains valid, use the `LT_IsTrialDaysValid()` procedure.

LT_GetTamperCount()

Purpose:

The `LT_GetTamperCount()` procedure retrieves the current Tamper Count value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine how many validation failures, license mismatches, or developer-defined tamper events have been recorded.

Syntax:

```
LT_GetTamperCount(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Integer

: Returns the current Tamper Count value stored within the installed `License.LIC` file.

Example:

```
// Display the current Tamper Count
Info("Tamper Count: " + _
     LT_GetTamperCount("abq87121"))
```

Remarks:

The `LT_GetTamperCount()` procedure retrieves the current Tamper Count value only. It does not increment the counter.

Applications may use this value to determine whether repeated validation failures or suspicious licensing events have occurred.

The developer determines how the application should respond when the Tamper Count reaches a particular threshold. Possible responses include displaying a warning, restricting selected features, requiring customer support contact, or preventing further application execution.

LT_GetLicensedThroughDate()

Purpose:

The `LT_GetLicensedThroughDate()` procedure retrieves the Licensed Through Date value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine the expiration date associated with a Subscription license or other time-limited licensing model.

Syntax:

```
LT_GetLicensedThroughDate(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Date

: Returns the Licensed Through Date value stored within the installed `License.LIC` file.

Example:

```
// Display the subscription expiration date
Info("Licensed Through: " + _
     DateToString( _
     LT_GetLicensedThroughDate("abq87121")))
```

Remarks:

The `LT_GetLicensedThroughDate()` procedure simply retrieves the Licensed Through Date value stored within the `License.LIC` file. It does not determine whether the subscription period remains valid.

Applications that implement Subscription licensing models may use this value to display renewal reminders, populate About dialogs or status bars, or present subscription information to the user.

To determine whether the current subscription remains valid, use the `LT_IsLicensedThroughDateValid()` procedure.

LT_GetLicenseHash()

Purpose:

The `LT_GetLicenseHash()` procedure retrieves the License Hash value stored within the currently installed `License.LIC` file.

The License Hash is generated during license creation and serves as an integrity verification mechanism, helping to detect unauthorized modification or substitution of the `License.LIC` file.

Syntax:

```
LT_GetLicenseHash(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

String

: Returns the License Hash value stored within the installed `License.LIC` file.

Example:

```
// Display the current License Hash
Info("License Hash: " + _
     LT_GetLicenseHash("abq87121"))
```

Remarks:

The `LT_GetLicenseHash()` procedure simply retrieves the License Hash value stored within the `License.LIC` file. It does not perform any validation or determine whether the License Hash is authentic.

Applications that require integrity verification should use the `LT_ValidateLicenseHash()` procedure to compare the stored License Hash against the expected value generated by the Runtime Library.

The License Hash is primarily intended for internal Runtime Library operations and diagnostic purposes. Most protected applications will not need to directly display or manipulate this value, although it may be useful during development, troubleshooting, or advanced licensing scenarios.

LT_IncrementExecutionCount()

Purpose:

The `LT_IncrementExecutionCount()` procedure increments the Execution Count value stored within the currently installed `License.LIC` file.

This procedure is typically called during application startup to record a successful application execution and to support Trial Run or Hybrid Trial licensing models.

Syntax:

```
LT_IncrementExecutionCount(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

This procedure does not return a value.

Example:

```
// Record a successful application execution
LT_IncrementExecutionCount("abq87121")
```

Remarks:

The `LT_IncrementExecutionCount()` procedure updates the Execution Count field stored within the installed `License.LIC` file by incrementing its current value by one.

This procedure is normally called internally by the LicenseTrak Runtime Library during application initialization. Under typical circumstances, developers will not need to invoke this procedure directly.

Applications that implement Trial Run or Hybrid Trial licensing models may use the updated Execution Count value together with `LT_GetExecutionCount()` and `LT_IsTrialRunsValid()` to display Trial usage information or determine whether Trial Run limits have been exceeded.

LT_IncrementTamperCount()

Purpose:

The `LT_IncrementTamperCount()` procedure increments the Tamper Count value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to record validation failures or other developer-defined events that may indicate unauthorized modification or misuse of the application's licensing system.

Syntax:

```
LT_IncrementTamperCount(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

This procedure does not return a value.

Example:

```
// Record a license validation failure
LT_IncrementTamperCount("abq87121")
```

Remarks:

The `LT_IncrementTamperCount()` procedure updates the Tamper Count field stored within the installed `License.LIC` file by incrementing its current value by one.

The Runtime Library may invoke this procedure automatically when certain license validation checks fail. Developers may also choose to call this procedure directly when application-specific events or conditions warrant recording a potential tamper incident.

The Tamper Count value may be retrieved using the `LT_GetTamperCount()` procedure and evaluated against developer-defined thresholds to determine the appropriate application response. Possible responses include displaying an informational message, restricting selected application features, requiring user intervention, or preventing further application execution.

LT_GetDateCreated()

Purpose:

The `LT_GetDateCreated()` procedure retrieves the Date Created value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine when the License.LIC file was originally generated by the LicenseTrak license management application.

Syntax:

```
LT_GetDateCreated(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Date

: Returns the Date Created value stored within the installed `License.LIC` file.

Example:

```
// Display the date the License.LIC
// file was originally created
Info("License Created: " + _
     DateToString( _
     LT_GetDateCreated("abq87121")))
```

Remarks:

The `LT_GetDateCreated()` procedure simply retrieves the Date Created field stored within the `License.LIC` file. It does not determine whether the license remains valid or whether the application is currently operating within any Trial or Subscription limitations.

The Date Created value may be useful for diagnostic purposes, customer support activities, or displaying license information within an About dialog or administrative utility. Developers may also use this value to assist with license auditing or troubleshooting when working with end users.

LT_GetApplicationName()

Purpose:

The `LT_GetApplicationName()` procedure retrieves the Application Name value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to verify or display the name and version information associated with the installed license.

Syntax:

```
LT_GetApplicationName(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

String

: Returns the Application Name value stored within the installed `License.LIC` file.

Example:

```
// Display the licensed application name
Info("Licensed Application: " + _
     LT_GetApplicationName("abq87121"))
```

Remarks:

The `LT_GetApplicationName()` procedure simply retrieves the Application Name field stored within the `License.LIC` file. It does not validate that the license belongs to the currently executing application.

Application validation is normally performed by the `LT_Initialize()` procedure during application startup. Under typical circumstances, developers will use `LT_GetApplicationName()` for informational or diagnostic purposes, such as displaying license details within an About dialog, administrative utility, or support screen.

The Application Name field typically contains both the application name and version information, allowing a single `License.LIC` file to be associated with a specific protected application release.

LT_GetApplicationID()

Purpose:

The `LT_GetApplicationID()` procedure retrieves the Application ID value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to retrieve the unique identifier associated with the installed license and may be used for informational, diagnostic, or developer-defined licensing operations.

Syntax:

```
LT_GetApplicationID(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

String

: Returns the Application ID value stored within the installed `License.LIC` file.

Example:

```
// Display the licensed Application ID
Info("Application ID: " + _
     LT_GetApplicationID("abq87121"))
```

Remarks:

The `LT_GetApplicationID()` procedure simply retrieves the Application ID field stored within the `License.LIC` file. It does not validate that the Application ID belongs to the currently executing application.

The Application ID may be useful for development, troubleshooting, or customer support purposes where a concise identifier is preferable to the full Application Name and Version string.

In most implementations, application validation is performed by the `LT_Initialize()` procedure during startup. The `LT_GetApplicationID()` procedure is provided primarily as a convenience function for applications or utilities that wish to display or otherwise utilize the stored Application ID value.

LT_GetDateCreated()

Purpose:

The `LT_GetDateCreated()` procedure retrieves the Date Created value stored within the currently installed `License.LIC` file.

This procedure allows the protected application to determine when the `License.LIC` file was originally generated by the LicenseTrak license management application.

Syntax:

```
LT_GetDateCreated(psDeveloperSeed)
```

Parameters:

`psDeveloperSeed`

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created.

Returns:

Date

: Returns the Date Created value stored within the installed `License.LIC` file.

Example:

```
// Display the date the License.LIC
// file was created
Info("License Created: " + _
     DateToString( _
     LT_GetDateCreated("abq87121")))
```

Remarks:

The `LT_GetDateCreated()` procedure simply retrieves the Date Created field stored within the `License.LIC` file. It does not determine whether the license remains valid or whether any Trial or Subscription limitations have been exceeded.

The Date Created value is primarily intended for informational, diagnostic, and customer support purposes. Developers may choose to display this value within an About dialog, administrative utility, or license information window to assist with troubleshooting and license management activities.

LT_SetFirstRunDate()

Purpose:

The LT_SetFirstRunDate() procedure initializes the FirstRunDate field within the License.LIC file. This procedure should be called during application startup before evaluating trial-day restrictions. If the FirstRunDate field is empty, the current system date is written to the License.LIC file. If a date already exists, no changes are made.

This procedure is typically used in conjunction with LT_IsTrialDaysValid() to begin tracking the customer's evaluation period from the first successful execution of the application.

Syntax:

```
LT_SetFirstRunDate(DeveloperSeed)
```

Parameters:

```
psDeveloperSeed
```

: Developer-defined seed value used during license generation and validation. This value must match the Developer Seed used when the `License.LIC` file was originally created. **Return Value**

Returns:

This procedure does not return a value.

Example

```
// Initialize the FirstRunDate field if this is the
// first time the application has been executed.


LT_SetFirstRunDate(gsDeveloperSeed)
```

Typical Usage

```
// Verify the License.LIC file belongs to this application.

IF NOT LT_Initialize(gsDeveloperSeed, gsAppNameVersion) THEN
   RETURN
END

// Record the first execution date if necessary.
LT_SetFirstRunDate(gsDeveloperSeed)

// Verify the trial period has not expired.
IF NOT LT_IsTrialDaysValid(gsDeveloperSeed) THEN
   Info("The trial period has expired.")
   RETURN
END
```

Notes

- The FirstRunDate field is written only once.
- Subsequent executions of the application will not modify the existing date.
- This procedure is intended for use with trial-day licensing strategies.
- The procedure should normally be called before LT_IsTrialDaysValid().

If the License.LIC file already contains a FirstRunDate value, the procedure exits without making changes.

LT_IIF()

Purpose:

The LT_IIF() procedure evaluates a Boolean expression and returns one of two values.

This procedure provides functionality similar to the traditional Immediate IF (IIF) function found in many programming languages and can simplify assignment statements within protected applications.

Syntax:

LT_IIF(plExpression, psTrueValue, psFalseValue)

Parameters:

plExpression

: Boolean expression to evaluate.

psTrueValue

: Value returned when the expression evaluates to **True**.

psFalseValue

: Value returned when the expression evaluates to **False**.

Returns:

Returns either psTrueValue or psFalseValue, depending upon the result of the Boolean expression.

Example:

sStatus = LT_IIF(LT_IsRegistered("abq87121"), "Registered", "Trial")

Remarks:

The LT_IIF() procedure provides a concise alternative to multiple IF...ELSE statements when assigning values based upon a simple condition.

Although intended primarily for use within the LicenseTrak Runtime Library, the procedure may also be used as a general-purpose utility routine throughout the protected application.

Chapter 15 - Best Practices

Protecting Your Developer Seed

The Developer Seed is one of the most important elements of the LicenseTrak licensing system. It is used during license generation and runtime validation to help ensure that a `License.LIC` file belongs to the intended software publisher and protected application.

Although the Developer Seed is not intended to be a military-grade secret, it should be treated as an internal implementation detail and should not be unnecessarily exposed to end users or third-party applications.

The following recommendations will help maximize the effectiveness of the Developer Seed within your LicenseTrak implementation:

- Create a Developer Seed that is unique to your organization or software product line.
- Avoid using obvious or easily guessed values such as company names, product names, or simple sequential numbers.
- Do not display the Developer Seed in application dialogs, About boxes, or user-facing reports.
- Avoid storing the Developer Seed in plaintext configuration files that are distributed with the application whenever practical.
- Do not include the Developer Seed in log files, debug messages, or technical support screenshots provided to customers.
- Maintain a secure backup of your Developer Seed. If the value is lost, recreating or regenerating existing customer licenses may become significantly more difficult.

Remember that the Developer Seed is only one component of the overall LicenseTrak protection model. The strongest implementations combine proper Developer Seed management with application identity validation, License Hash verification, and appropriate runtime licensing checks.

Finally, resist the temptation to overcomplicate the design. The goal of the Developer Seed is not to create an unbreakable protection mechanism—no desktop licensing system can make that claim. Instead, the objective is to provide a practical and effective layer of protection that discourages casual misuse while remaining easy to administer and support.

Once a LicenseTrak implementation has been deployed and customer licenses have been generated, changing the Developer Seed should be undertaken only after careful planning. Existing licenses are tied to the original Developer Seed, and changing it may require regenerating and redistributing replacement `License.LIC` files to existing customers.

Validating At Startup

One of the most effective ways to integrate LicenseTrak into a protected application is to perform license validation as early as possible during application startup. By verifying the integrity and ownership of the `License.LIC` file before the application begins normal operation, developers can ensure that licensing issues are detected and handled in a predictable and user-friendly manner.

A typical LicenseTrak implementation calls the `LT_Initialize()` procedure from the application's main initialization event. This allows the Runtime Library to locate the `License.LIC` file, validate the Application Name and License Hash values, and load the licensing information into memory before any protected functionality is accessed.

Performing license validation at startup provides several advantages:

- Invalid or missing `License.LIC` files are detected before the user begins working.
- Application identity mismatches are identified immediately.
- Runtime licensing information is available to all subsequent application modules.
- Feature restrictions and Trial limitations can be applied consistently throughout the application.

Developers should avoid delaying the initial licensing check until a user attempts to perform a restricted operation. Discovering a licensing problem after a user has already entered data or completed several tasks may lead to confusion and a poor user experience.

A recommended startup sequence is:

1. Call `LT_Initialize()` during the application's initialization process.
2. Verify that the installed `License.LIC` file belongs to the protected application.
3. Evaluate the current licensing state (Trial, Registered, Subscription, or Locked).
4. Apply any developer-defined restrictions or notifications.
5. Continue normal application execution.

Applications should always fail gracefully when a licensing issue is encountered. Clear, informative messages explaining the condition and the corrective action required are generally more effective than abrupt program termination or cryptic error codes. A customer who understands *why* a licensing issue occurred is far more likely to resolve the problem than one who is simply presented with a generic error message.

Finally, remember that startup validation is intended to establish a trusted baseline for the application session. Additional checks may still be appropriate before performing critical operations, particularly when implementing Trial limitations or feature-restricted licensing models.

Validating Before Critical Operations

While validating the `License.LIC` file during application startup establishes a trusted baseline, many applications benefit from performing additional license checks before executing critical operations. These secondary checks provide an additional layer of protection and allow the application to respond appropriately when a licensing condition changes.

Critical operations are typically those that provide significant value to the end user or that expose application functionality beyond what is intended for a Trial or restricted license. Examples include:

- Adding or modifying database records.
- Exporting application data to external formats.
- Printing reports or generating documents.
- Accessing premium or advanced application features.
- Performing administrative or maintenance functions.

Rather than assuming that a single startup validation is sufficient for the lifetime of the application session, developers should consider validating the current licensing state immediately before these operations are performed. This approach helps ensure that Trial limitations, Subscription expirations, and feature restrictions are consistently enforced.

In many cases, the implementation can be as simple as placing a LicenseTrak function call within the button or menu option that initiates the operation:

```
// Prevent data export while operating
// under a Trial license
IF LT_IsTrial("Trial") THEN
    Error("Data export is unavailable in the Trial version.")
    RETURN
END
```

This strategy also provides a better user experience. Instead of preventing the application from launching entirely, the developer can allow the user to continue working while selectively restricting premium functionality until the software is registered or the subscription is renewed.

As a general design principle, LicenseTrak encourages developers to think in terms of **graduated restrictions** rather than an all-or-nothing approach. A user who can continue viewing existing data, printing a reminder message, or exploring the application's interface is more likely to become a satisfied customer than one who is abruptly locked out of the program.

Finally, remember that not every operation requires a separate licensing check. The goal is to protect the features that are most valuable to your application while avoiding unnecessary complexity or excessive validation logic. A well-designed licensing strategy should be effective, maintainable, and largely invisible to legitimate users.

Handling Expired Licenses

Sooner or later, every protected application will encounter an expired Trial period or a Subscription license that has reached its Licensed Through Date. How the application responds to these situations can have a significant impact on the customer's overall experience.

Whenever practical, applications should handle expired licenses gracefully. Rather than abruptly terminating the program or presenting cryptic error messages, developers should provide clear explanations of the licensing condition and guide the user toward the appropriate corrective action.

For example, an expired Trial version might display a message explaining that the evaluation period has ended and that a registered license is required to continue using premium functionality. Similarly, an expired Subscription license might encourage the customer to renew their subscription while still permitting access to existing data.

Many developers find that a graduated response model provides the best balance between protecting the application and maintaining customer goodwill. Examples include:

- Allowing users to continue viewing existing data while preventing the creation of new records.
- Permitting report preview while restricting printing or export operations.
- Operating in a read-only mode until a valid license is installed.
- Displaying periodic reminders that a Trial or Subscription period has expired.

Where possible, avoid placing customer data at risk. A user who can still access and review their information is far more likely to purchase or renew a license than one who believes their data has been lost or held hostage.

LicenseTrak was intentionally designed to give developers flexibility in this area. The Runtime Library provides the information needed to determine the current licensing state, but the application developer remains in control of the enforcement policy. This allows each software publisher to implement a licensing strategy that best fits the needs of their application and customer base.

Finally, remember that legitimate customers occasionally encounter licensing issues due to hardware failures, file corruption, or accidental deletion of the `License.LIC` file. Designing the application to respond with informative messages and reasonable recovery options can significantly reduce support calls while fostering a positive relationship with the customer.

Managing Trial Versions

A well-designed Trial version allows potential customers to experience the value of an application before making a purchasing decision. The objective of a Trial license is not simply to restrict access, but to provide a meaningful evaluation experience while encouraging users to become registered customers.

LicenseTrak supports several common Trial licensing strategies, including Trial Day limits, Trial Run limits, and Hybrid models that combine both approaches. The most appropriate strategy depends upon the nature of the protected application and the developer's business model.

When designing a Trial version, developers should strive to balance software protection with customer goodwill. A Trial that is too restrictive may prevent users from adequately evaluating the application, while a Trial with few limitations may provide insufficient incentive to purchase a registered license.

The following recommendations have proven effective for many desktop applications:

- Allow users to explore the application's primary features.
- Clearly communicate the Trial limitations and remaining evaluation period.
- Display friendly registration reminders rather than intrusive or repetitive warning messages.
- Avoid placing customer data at risk when the Trial period expires.
- Provide a straightforward process for upgrading from a Trial to a Registered or Subscription license.

Many developers find that a Hybrid licensing model offers an effective balance between flexibility and protection. By combining both Trial Day and Trial Run limitations, the evaluation period remains fair for users who launch the application frequently as well as those who use it only occasionally.

As discussed in earlier chapters, Trial limitations do not need to result in complete loss of application functionality. Restricting selected features, operating in a read-only mode, or disabling data export while still permitting users to access their existing information often provides a more positive customer experience than abruptly terminating the application.

Finally, remember that the Trial version serves as the first impression many customers will have of your software. A professional, well-designed evaluation experience demonstrates confidence in your product and helps build trust with prospective customers. In many cases, the quality of the Trial experience itself can play a significant role in converting an evaluator into a long-term customer.

Deploying Updates

As a protected application evolves, developers will inevitably release updates that add features, correct defects, or improve the overall user experience. A well-designed update process should allow customers to install these improvements without unnecessarily disrupting their existing licensing information.

One of the primary goals of LicenseTrak is to separate application updates from license management whenever practical. In most cases, a customer should be able to install a newer version of the application while continuing to use their existing `License.LIC` file without modification.

When planning an update, developers should consider the impact that application changes may have on the licensing system. Changes to user interface elements, reports, database structures, or internal business logic typically do not require customer licenses to be regenerated. However, significant changes to the application's identity or licensing architecture may require additional planning.

Before distributing a software update, consider the following recommendations:

- Verify that existing `License.LIC` files continue to function correctly with the updated application.
- Test the update using Trial, Registered, Subscription, and Locked license scenarios.
- Confirm that Trial limitations and Subscription expiration checks continue to operate as expected.
- Verify that feature restrictions, report watermarks, and export limitations remain properly enforced.
- Maintain backups of the LicenseTrak management database before performing major application or licensing changes.

Whenever possible, avoid making changes that require customers to request replacement license files. Minimizing administrative overhead benefits both the software publisher and the customer, while reducing the number of support requests associated with routine software maintenance.

If a future application update requires changes to the licensing architecture, communicate those changes clearly and provide straightforward instructions for obtaining updated `License.LIC` files. A simple, predictable update process helps build customer confidence and reinforces the professionalism of the protected application.

Finally, developers should maintain a small collection of representative test licenses for use during update validation. Maintaining Trial, Registered, Subscription, and Locked sample `License.LIC` files makes it easy to verify that application updates have not unintentionally altered expected licensing behavior.

Backing Up LicenseTrak

A reliable backup strategy is one of the simplest and most effective ways to protect both the LicenseTrak management environment and the applications that depend upon it. While considerable attention is often given to software licensing and security, the ability to recover from accidental data loss or hardware failure is equally important.

The LicenseTrak management application contains valuable information, including customer records, application definitions, issued licenses, and support history. Recreating this information after a disk failure or accidental deletion can be time-consuming and, in some cases, impossible.

Developers should establish a regular backup schedule for the LicenseTrak data files and include them as part of their

normal system maintenance procedures. Frequent backups reduce the risk of data loss and provide a reliable recovery point should an unexpected problem occur.

A good backup strategy should include:

- Regular backups of the LicenseTrak application data files.
- Backups of generated `License.LIC` files and customer license archives.
- Secure storage of the Developer Seed and other licensing-related configuration information.
- Maintaining multiple generations of backups rather than relying upon a single copy.
- Periodically verifying that backup files can be successfully restored.

Whenever significant changes are made to the LicenseTrak database or licensing configuration, consider creating an additional manual backup before proceeding. This simple precaution provides a recovery point should an unexpected error occur during maintenance or software updates.

Where practical, backup copies should be stored on media that is physically separate from the primary development system. External drives, network-attached storage devices, or other offline storage solutions can help protect against hardware failures, accidental deletion, or ransomware incidents.

Finally, remember that a backup strategy is only as good as its restore process. Periodically performing a test restoration of the LicenseTrak environment helps verify that backup procedures are functioning correctly and provides confidence that the licensing database can be recovered if the need ever arises.

A backup that has never been tested is only a theory.

Chapter 16 - FAQ

Q: What is LicenseTrak?

A: LicenseTrak is a lightweight software licensing system designed for desktop applications developed with WINDEV. It combines a simple license management application with a reusable Runtime Library that developers can integrate into their own applications to implement Trial, Registered, Subscription, and feature-restricted licensing models.

Q: Does LicenseTrak require an Internet connection?

A: No. LicenseTrak was designed to operate entirely offline. The LicenseTrak management application generates a `License.LIC` file that is distributed to the customer and validated locally by the protected application. No online activation servers or cloud services are required.

Q: What types of licensing models does LicenseTrak support?

A: LicenseTrak supports several common licensing strategies, including:

- Trial Day licensing.
- Trial Run licensing.
- Hybrid Trial licensing (combining Trial Days and Trial Runs).
- Perpetual Registered licensing.

- Subscription licensing.
- Developer-defined feature restriction models.

These models may be used individually or combined to meet the requirements of a particular application.

Q: Is LicenseTrak limited to WINDEV applications?

A: The LicenseTrak management application and Runtime Library are written in WINDEV and are intended primarily for WINDEV developers. However, because the `License.LIC` file is a standard HFSQL data file, developers using other programming languages may create compatible Runtime Libraries if desired.

Q: Does LicenseTrak use online activation or hardware fingerprinting?

A: No. LicenseTrak intentionally avoids mandatory online activation and hardware fingerprinting. The system was designed to provide a straightforward, low-maintenance licensing solution that respects user privacy while minimizing administrative overhead for the software publisher.

Q: What versions of WINDEV are supported?

A: LicenseTrak was developed and tested using modern WINDEV versions. The LicenseTrak Runtime Library uses the `Encrypt()` function, which requires WINDEV 24 or later. Developers using earlier WINDEV versions may need to adapt the Runtime Library to use the older `Crypt()` function and should perform a test import and compilation before deployment.

Chapter 17 - The LicenseTrak Example Application

The LicenseTrak distribution package includes a complete example application designed to demonstrate the integration of the LicenseTrak Runtime Library into a typical WINDEV desktop project. Rather than presenting isolated code fragments, the example application provides a working reference implementation that developers may study, modify, and adapt for their own projects.

The example application was intentionally designed to remain simple while showcasing the most common licensing scenarios encountered by desktop software developers. It demonstrates how the Runtime Library can be initialized during application startup, how licensing information may be displayed to the user, and how Trial or Subscription restrictions can be incorporated into day-to-day application workflows.

Throughout this manual, many of the screenshots and code examples have been taken directly from the example application. The purpose of this chapter is to bring those individual demonstrations together and show how the various LicenseTrak components operate as a cohesive whole.

Developers are encouraged to explore the example project, experiment with different `License.LIC` files, and modify the sample code to better understand the behavior of the Runtime Library. The example application is not intended to dictate a single licensing strategy, but rather to provide a practical foundation upon which custom

licensing solutions may be built.

Appendix A - Deployment Checklist

The following checklist summarizes the recommended steps for integrating LicenseTrak into a new WINDEV application. Completing these tasks in order will help ensure a smooth installation and provide a solid foundation for implementing your desired licensing strategy.

Initial Setup

☐ Install the LicenseTrak management application.

☐ Review the LicenseTrak User Manual and familiarize yourself with the available licensing models.

☐ Select and record a unique **Developer Seed** value for your organization or application.

☒ **Important:** Store the Developer Seed in a secure location. Existing customer licenses are tied to this value.

Project Preparation

☐ Import the `License.FIC` hyperfile into your WINDEV Analysis.

☐ Modify the imported file properties to use the `.LIC` file extension rather than the default HFSQL extension.

☒ **Recommended:** Configure the `License.LIC` file to reside within the application folder for simplified deployment.

☐ Import the `LicenseTrak.WL` Runtime Library into your project as a Global Procedure set.

☐ Verify that the imported Runtime Library compiles successfully.

Configure the LicenseTrak Management Application

☐ Create an entry for your protected application.

☐ Enter the appropriate Application Name and Version information.

☐ Configure and verify your Developer Seed value.

☒ **Recommended:** Create a special customer record (for example, "Trial User") that will be used to generate evaluation licenses distributed with your application.

Generate Initial License Files

☐ Generate a Trial `License.LIC` file for development and testing.

☒ **Recommended:** Use a 30–45 day Trial period during development to simplify testing.

☐ Copy the generated `License.LIC` file into your application's execution folder.

☒ **Optional:** Create additional Registered, Subscription, and Locked sample licenses for future testing.

Integrate the Runtime Library

☐ Call `LT_Initialize()` during application startup.

☒ **Recommended:** Perform initialization before the user accesses any application functionality.

☐ Validate that the application starts correctly with a valid Trial license.

☒ **Recommended:** Test startup behavior with missing, invalid, and tampered license files.

Implement Your Licensing Strategy

☐ Decide which licensing model best fits your application:

- Trial Day
- Trial Run
- Hybrid Trial
- Registered
- Subscription
- Feature Restriction

☐ Add LicenseTrak validation checks before critical operations.

☐ Determine which application features should remain available after Trial expiration.

☐ Design user-friendly messages for Trial expiration, Subscription renewal, and invalid license conditions.

Test and Validate

☐ Verify application behavior using Trial, Registered, Subscription, and Locked license files.

☐ Test report printing and watermark behavior.

☐ Test feature restriction and read-only modes.

☐ Test data export restrictions.

☒ **Recommended:** Maintain a small library of sample `License.LIC` files for future regression testing.

Final Deployment

☐ Include the initial Trial `License.LIC` file with your application installer.

☒ **Recommended:** Use a generic customer such as "Trial User" for evaluation copies.

☒ **Recommended:** Maintain backups of your LicenseTrak database and generated customer license files.

☒ **Recommended:** Test the complete installation process on a clean computer before releasing your application.

Post-Deployment

☐ Generate and distribute Registered or Subscription `License.LIC` files as customers purchase your software.

☒ **Recommended:** Archive all generated license files for future support and recovery purposes.

☒ **Recommended:** Keep the LicenseTrak management database and Runtime Library under version control or regular backup.

☒ **Recommended:** Before releasing application updates, verify operation using representative Trial, Registered, Subscription, and Locked test licenses.

Appendix B - Troubleshooting

This appendix describes common issues that may occur while integrating or using LicenseTrak and provides suggested corrective actions.

The application reports that License.LIC is missing

Verify that the `License.LIC` file exists in the application's execution folder.

Confirm that the file was generated by the LicenseTrak management application.

If the application was installed using a setup program, verify that the installer includes the Trial `License.LIC` file and places it in the correct destination folder.

The application reports that the license was not generated for this application

This message indicates that the Application Name stored inside `License.LIC` does not match the application name passed to `LT_Initialize()`.

Verify that the Application Name and version string in LicenseTrak exactly match the value used by the protected application.

Generate a new `License.LIC` file for the correct application and replace the existing file.

The Runtime Library does not compile

Verify that the `License.FIC` definition has been imported into the application's Analysis.

Confirm that the imported file is named `License` and uses the `.LIC` extension.

Verify that the shipping `LicenseTrak_Runtime.WL` file was imported completely and that development-only procedures have not been included.

LicenseTrak requires WINDEV 24 or later because the Runtime Library uses the `Encrypt()` function.

The application always runs as Trial

Verify that the correct `License.LIC` file has been copied into the application folder.

Use `LT_DebugDisplayLicense()` to confirm the current `LicenseStatus` value.

If the customer has purchased the application, generate a Registered or Subscription license and replace the Trial license file.

Trial Days or Trial Runs are not behaving as expected

Use `LT_DebugDisplayLicense()` to inspect the `TrialDays`, `TrialRuns`, `ExecutionCount`, and `FirstRunDate` values.

Confirm that the application calls the appropriate validation functions, such as `LT_IsTrialDaysValid()` or `LT_IsTrialRunsValid()`.

If using Trial Run limits, confirm that execution counting is being updated as expected.

Subscription licenses appear expired

Verify the `LicensedThroughDate` value using `LT_DebugDisplayLicense()`.

Confirm that the system date on the customer's computer is correct.

Generate and deploy a replacement `License.LIC` file with an updated Licensed Through Date when the customer renews.

Feature restrictions are not applied

Confirm that the application is checking the correct LicenseTrak function before the restricted operation.

For example, export, print, add, edit, and delete operations should each perform their own license check if they are intended to be restricted.

Do not rely only on startup validation if the application must restrict critical operations during normal use.

The wrong customer name appears

Verify that the correct `License.LIC` file was copied into the application folder.

Generate a new license for the intended customer and replace the existing file.

Use `LT_DebugDisplayLicense()` to confirm the `RegisteredTo` value.

The license hash fails validation

The `License.LIC` file may have been modified, corrupted, or generated with a different Developer Seed.

Confirm that the Developer Seed used by LicenseTrak matches the value used by the protected application.

Generate a replacement license file and test again.

After updating the application, existing licenses no longer work

Verify that the Application Name/version string passed to `LT_Initialize()` still matches the value stored in existing customer licenses.

If the application identity has intentionally changed, replacement `License.LIC` files may need to be generated.

Before deploying updates, test with representative Trial, Registered, Subscription, and Locked license files.

Appendix C - Sample License Files

The LicenseTrak distribution package includes a collection of sample `License.LIC` files that demonstrate common licensing scenarios and provide a convenient testing framework for developers integrating the Runtime Library into their own applications. Each sample license is stored within a dedicated subfolder named according to its intended purpose. Developers are encouraged to copy these files into the example application's execution folder to observe the behavior of the various LicenseTrak validation and restriction functions under real-world conditions:

Incorrect Application Version
Incorrect Developer Seed
Locked
Registered
Subscription Expired
Subscription Valid
Trial – 10 Runs
Trial – 30 Days
Trial – 30 Days (Tampered License Hash)
Trial (Exceeded 30 Day Trial)

Appendix D - Runtime Function Cross Reference

The following cross reference groups the LicenseTrak Runtime Library functions by common development task. Use this appendix when you know what you want to accomplish but are not sure which Runtime Library function applies.

Application Startup and Initialization

`LT_Initialize()`
Initializes the LicenseTrak Runtime Library, loads the installed `License.LIC` file, validates application identity, and prepares licensing information for runtime use.

`LT_ValidateLicenseHash()`
Validates the integrity of the installed `License.LIC` file.

Determining License Status

`LT_IsTrial()`
Determines whether the installed license is a Trial license.

`LT_IsRegistered()`
Determines whether the installed license is a Registered license.

`LT_IsSubscription()`
Determines whether the installed license is a Subscription license.

`LT_IsLocked()`
Determines whether the installed license is a Locked license.

`LT_GetLicenseStatus()`
Retrieves the current License Status value.

Language Detection

LT_IsEnglish()
Determines whether the active application language is English.

LT_IsFrench()
Determines whether the active application language is French.

LT_IsSpanish()
Determines whether the active application language is Spanish.

Trial License Management

`LT_GetTrialDays()`
Retrieves the configured Trial Day limit.

`LT_IsTrialDaysValid()`
Determines whether the Trial Day limit remains valid.

`LT_GetTrialRuns()`
Retrieves the configured Trial Run limit.

`LT_GetExecutionCount()`
Retrieves the current Execution Count.

`LT_IncrementExecutionCount()`
Increments the Execution Count.

`LT_IsTrialRunsValid()`
Determines whether the Trial Run limit remains valid.

`LT_GetFirstRunDate()`
Retrieves the First Run Date.

Subscription License Management

`LT_GetLicensedThroughDate()`
Retrieves the Licensed Through Date.

`LT_IsLicensedThroughDateValid()`
Determines whether the current date remains within the Licensed Through Date.
Feature Restriction and Enforcement

`LT_IsApplicationRestricted()`
Determines whether the application should operate in a restricted mode based upon developer-defined licensing policy.

`LT_GetTamperCount()`
Retrieves the current Tamper Count.

`LT_IncrementTamperCount()`
Increments the Tamper Count.
Display and User Interface Integration

`LT_GetRegisteredTo()`
Retrieves the Registered To value for display in About boxes, status bars, reports, or startup messages.

`LT_GetApplicationName()`
Retrieves the Application Name stored within the license.

`LT_GetApplicationID()`
Retrieves the Application ID stored within the license.

`LT_GetDateCreated()`
Retrieves the date the `License.LIC` file was generated.

`LT_DebugDisplayLicense()`
Displays the contents of the installed license file for development and troubleshooting.
License Integrity and Diagnostics

`LT_GetLicenseHash()`
Retrieves the License Hash value.

`LT_ValidateLicenseHash()`
Validates the License Hash against the expected value.

`LT_DebugDisplayLicense()`
Displays all major license fields for diagnostic review.

Utility Functions

LT_IIF()
Returns one of two values based upon the evaluation of a Boolean expression.

Appendix E - Support and Registration

If you experience a problem while installing or using LicenseTrak, please contact us using the support email address provided on the Software by Daughtry web site. When requesting assistance, include a detailed description of the issue, the steps necessary to reproduce the problem, and any error messages or screenshots that may help us diagnose the situation. We will make every effort to respond as quickly as possible.

Before performing any troubleshooting or applying software updates, it is strongly recommended that you create a complete backup of your application folder and all associated data files. This simple precaution can help prevent accidental data loss and provide a reliable recovery point should an unexpected problem occur.

LicenseTrak is distributed as a Trial version. Upon receipt of payment, a personalized `License.LIC` file will be generated and delivered electronically. Simply replace the existing Trial `License.LIC` file with the registered version to unlock the application and remove the Trial restrictions.

Your personalized license file may be branded with your name or company name, providing a simple means of identifying the licensed owner. All data created or entered while using the Trial version remains fully accessible after registration; no data conversion or reinstallation is required.

Current pricing information, ordering instructions, and product updates are available on the Software by Daughtry web site. We sincerely appreciate your interest in LicenseTrak and hope it proves to be a valuable addition to your software development toolkit.