

AddressTools V10 - Security and Permissions Changes

Date: March 2026

Product: AddressTools Premium (Salesforce AppExchange Managed Package)

Audience: Salesforce Administrators, IT Teams

Classification: May be shared with customers

Summary

AddressTools V10 adopts modern Apex security best practices to deliver improved security that aligns with current AppExchange standards. Code within the managed package now executes with appropriate access levels, enforcing the running user's object-level and field-level security permissions at runtime.

While these changes strengthen overall security, they do introduce behavioural changes that need to be considered, especially for organisations that do not license all of their users.

This document explains what has changed, why the changes were made, the impact on your Salesforce Organization, and the options available to you.

Background

In earlier versions of AddressTools, the managed package executed its code in system context (elevated permissions). In practice, this meant Apex runs with substantially elevated permissions over the user running the code. For licensed users, this meant they'd be able to run AddressTools without the need for ensuring permissions were allocated, and for unlicensed users, it would silently bypass the logic.

This created a potential **data integrity risk**: Salesforce administrators had no way to know whether address validation was actually being performed for a given user. If unlicensed users were editing address records, their changes would bypass validation and standardisation silently, potentially introducing inconsistent or invalid data.

Note: During implementation, Salesforce administrators should have used standard Salesforce methods - such as profiles, permission sets, and page layouts - to ensure that unlicensed users were prevented from editing address fields. We understand that due to changes in staffing, team restructuring, or evolving business processes, these controls may have been subsequently overlooked or not maintained.

What Has Changed in V10

1. User-Context Execution (Enhanced Security)

AddressTools code now executes with a defined access level, aligning with current AppExchange security standards. All Salesforce object-level and field-level permissions are respected at runtime where appropriate.

Because the code now runs in the user's context, logic results in a failure may occur if the user is unlicensed or has insufficient permissions. Item 2 and 3 intervene to handle this and prompt guidance on how to resolve.

2. Packaged Triggers - Silent Return for Unlicensed Users

Packaged AddressTools triggers (those included in the managed package for standard objects) will **silently return** for unlicensed users. No error is raised - but crucially, **no validation or standardisation takes place**. Address data entered by unlicensed users will be saved without any AddressTools processing.

This is a deliberate design decision. Standard Salesforce behaviour is that objects, pages, and components are simply invisible or inaccessible to users who lack the relevant license or permission. A silent return aligns with this convention - the majority of Salesforce administrators would expect a managed package trigger to do nothing for an unlicensed user, rather than raise an error.

Important: Validation and standardisation can only be guaranteed for licensed users.

3. Custom Trigger Entry Points - Explicit Errors for Unlicensed Users

If your Salesforce Organization uses custom Apex triggers that explicitly call AddressTools validation classes (e.g. for custom address objects), these calls will now **raise a NoAccessException** for unlicensed users. The enhanced security checks mean the code cannot access the managed package resources it needs to run, and this exception will be thrown.

To prevent this, your custom trigger code should check whether the running user holds a valid AddressTools license before calling any AddressTools class. See *Your Options* below for details on how to perform this check.

4. License Checks Added to Additional Code Paths

As part of this work, license checks have been added to code paths that previously lacked them - including entry points used by custom triggers calling the AddressTools codebase. This ensures consistent behaviour: if a user is unlicensed, all code paths will either silently return (packaged triggers) or raise an explicit error (custom invocations).

Impact on Your Salesforce Organization

If All Users Are Licensed

No action is required. All functionality will continue to work as expected.

If Some Users Are Unlicensed

The impact depends on how AddressTools is invoked:

- **Packaged triggers (standard objects)** - The trigger will silently return for unlicensed users. No error is raised, but **address validation and standardisation will not occur**. Data entered by these users will be saved unvalidated.
 - **Custom triggers calling AddressTools validation classes** - The call will raise a `NoAccessException` for unlicensed users. If your trigger does not handle this, it may prevent the record from being saved.
 - **New-Record Override component** - All users who access a page using this override must hold a valid AddressTools license. There is no mechanism within the override component itself to distinguish between licensed and unlicensed users prior to invoking managed package code.
-

New-Record Override Component

The AddressTools New-Record Override component replaces the standard Salesforce new-record page with a version that includes address validation. Because this component executes managed package code as soon as the page loads, **every user who accesses the overridden page must hold a valid AddressTools license**.

If your Salesforce Organization needs to allow some users to create records without address validation:

- **Remove the override** for those record types or profiles where validation is not required, and rely on the standard Salesforce record creation page instead.
 - **Create a custom Lightning Web Component (LWC)** that acts as a wrapper. The custom component can check whether the current user holds an AddressTools license (or meets other business criteria) before deciding whether to invoke the AddressTools override or redirect to the standard page.
-

Your Options

Option 1: License All Users Who May Invoke AddressTools (Recommended)

The simplest and most reliable approach is to ensure that every user who could potentially trigger AddressTools logic holds a valid license. This includes users who:

- Directly edit address fields on standard or custom objects
- Create records via an overridden new-record page
- Run batch processes, integrations, or flows that update address fields

This eliminates any need for custom code changes on your side and guarantees consistent address validation across your Salesforce Organization.

To discuss adjusting your license allocation, please contact the ProvenWorks renewals team.

Contact: renewals@provenworks.com

Option 2: Ensure Only Licensed Users Invoke AddressTools

If your Salesforce Organization does not require address validation for all users, you can implement business logic within your own Apex triggers to control when AddressTools is called. Before calling any AddressTools class or method, check whether the running user holds a valid license:

UserInfo.isCurrentUserLicensed()

The simplest and most direct method. This returns true if the current user has an active license for the specified managed package namespace. It does not consume a SOQL query.

```
if (UserInfo.isCurrentUserLicensed('pw_ccpro')) {
    // Safe to call AddressTools classes
}
```

See: [UserInfo.isCurrentUserLicensed\(\)](#)

Regardless of the approach chosen:

1. Only invoke AddressTools logic when the check passes.
2. For the New-Record Override component, use a custom wrapper component as described above.

This option is appropriate when you have a clear understanding of which users need address validation and can confidently ensure that other users' processes do not require it.

Important: If unlicensed users are editing address fields and your business expects those addresses to be validated, those records will contain unvalidated data. Ensure this is an acceptable outcome before proceeding with this option.

Version Support Lifecycle

ProvenWorks maintains a rolling support window for all Salesforce AppExchange products. Only versions released within the prior 24 months from the current date are eligible for bug fixes, security patches, and compatibility support.

Customers running versions outside this window should plan to upgrade to a supported release.

For full details, refer to the **ProvenWorks Product Version Support Policy** (POL-VERS-001).

For upgrade guidance including bridge packages, refer to the AddressTools Upgrade Guide.

Frequently Asked Questions

Can we continue using our current version?

You are not required to upgrade immediately. However, if your current version is outside the 24-month support window, it will no longer receive bug fixes, security patches, or compatibility assurance. When you do upgrade, V10 (or later) will include the security and permissions changes described in this document.

Can we test this in a sandbox?

In Salesforce sandbox environments, all users automatically receive a site license for installed managed packages - this is standard Salesforce platform behaviour and is outside the control of AppExchange publishers. As a result, you cannot reliably test license-related behaviour in a sandbox because all users will appear to be licensed. We recommend using sandbox environments for functional testing of address validation, and reviewing your license assignments and trigger logic in production.

Document History

Version	Date	Description
1.0	March 2026	Initial publication - AddressTools V10 security and permissions changes