Processity Data History

Quick Start Guide

A Note on Naming

You might find references to Processity Data History's old name, 'Audicity' in some parts of the application, where renaming would be disruptive to existing customers, or in areas that are impossible to rename due to the constraints of Salesforce managed packaging. If you're in any doubt about an application artefact please don't hesitate to contact us via 'support@processity.ai'.

Overview

Processity Data History enables unlimited field history tracking in Salesforce. With Processity Data History, you can track more fields and retain their history for longer than with standard field tracking.

The field changes are easily accessible by adding a Processity Data History component to your object record pages.

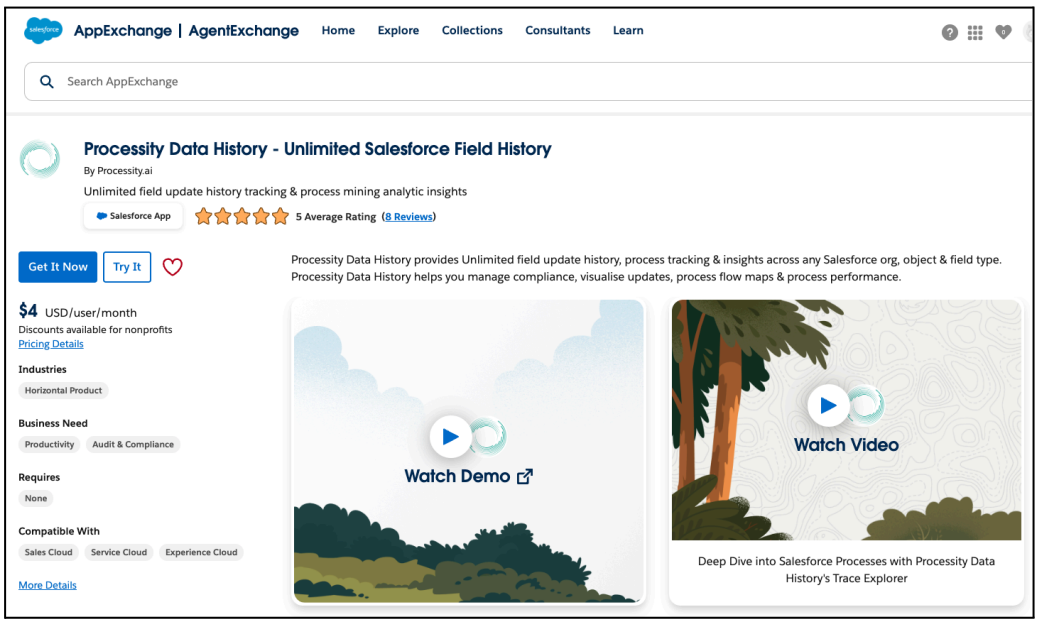
You can gain insights into the context of those changes by looking at the way they are grouped and by using the Processity Data History Trace Explorer. This can help to reveal situations where a change in one object originated elsewhere, but cascaded across records by automation.

This Quick-start guide will take you through the process of installing and configuring Processity Data History into a Salesforce sandbox so that you can start to see the benefits for yourself.

Installation

1. Navigate to the AppExchange Listing: https://appexchange.salesforce.com/appxListingDetail?listingId=8ecf5cc2-cc0d-4292-9d22-ff5a73568828

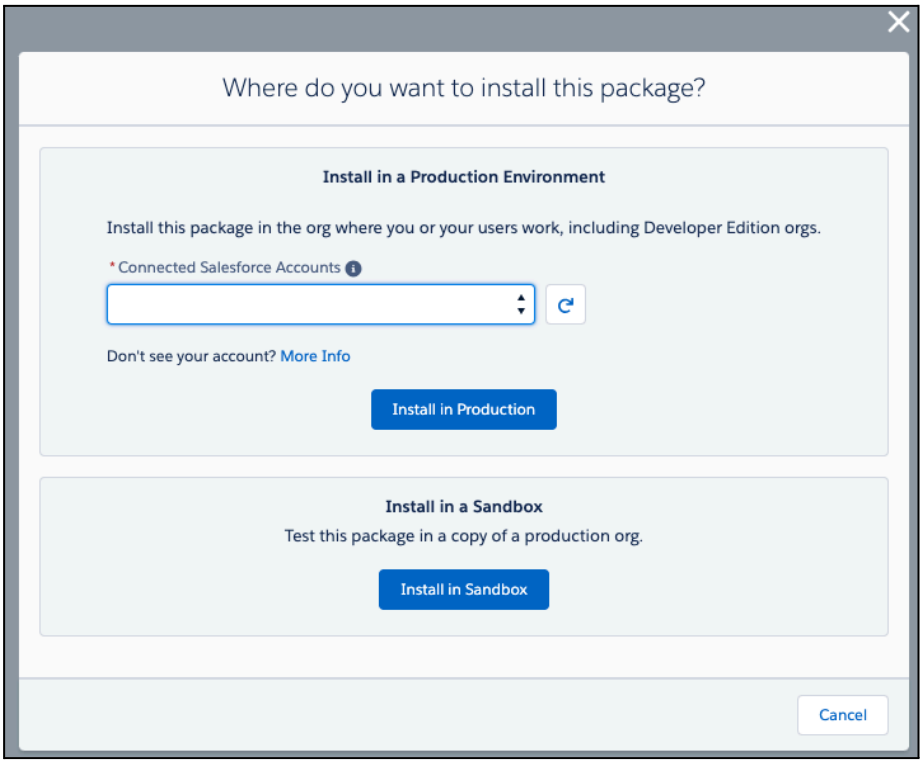
2. Click the “Get It Now” button.



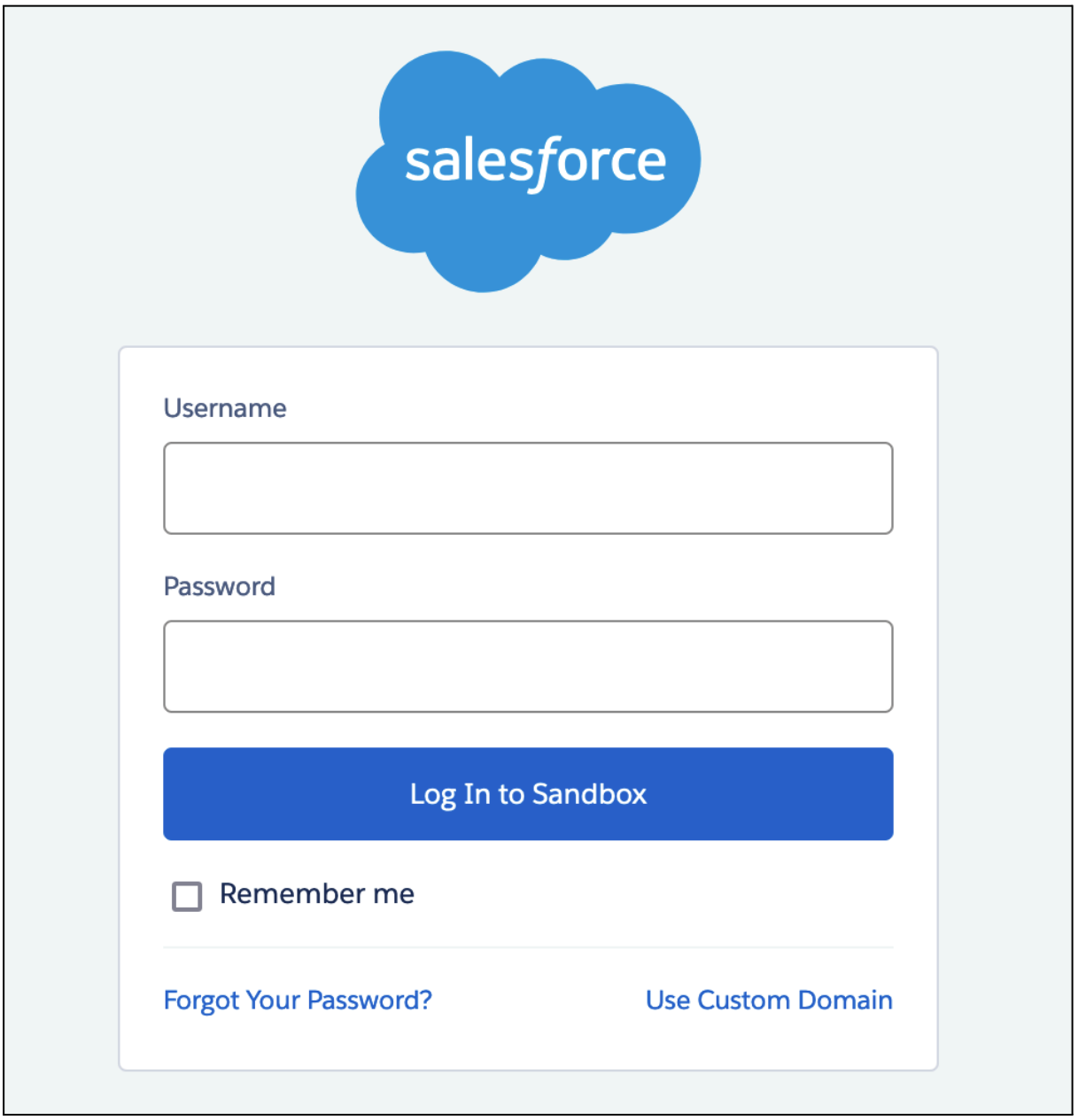
Note that the “Try It” button lets you access a ‘test drive’ org to see how Processity Data History looks in a read-only org that we created for that purpose. By all means, check that out before installing in your sandbox. Then come back and use the “Get It Now” button.

3. Log in to your Trailblazer.me profile.

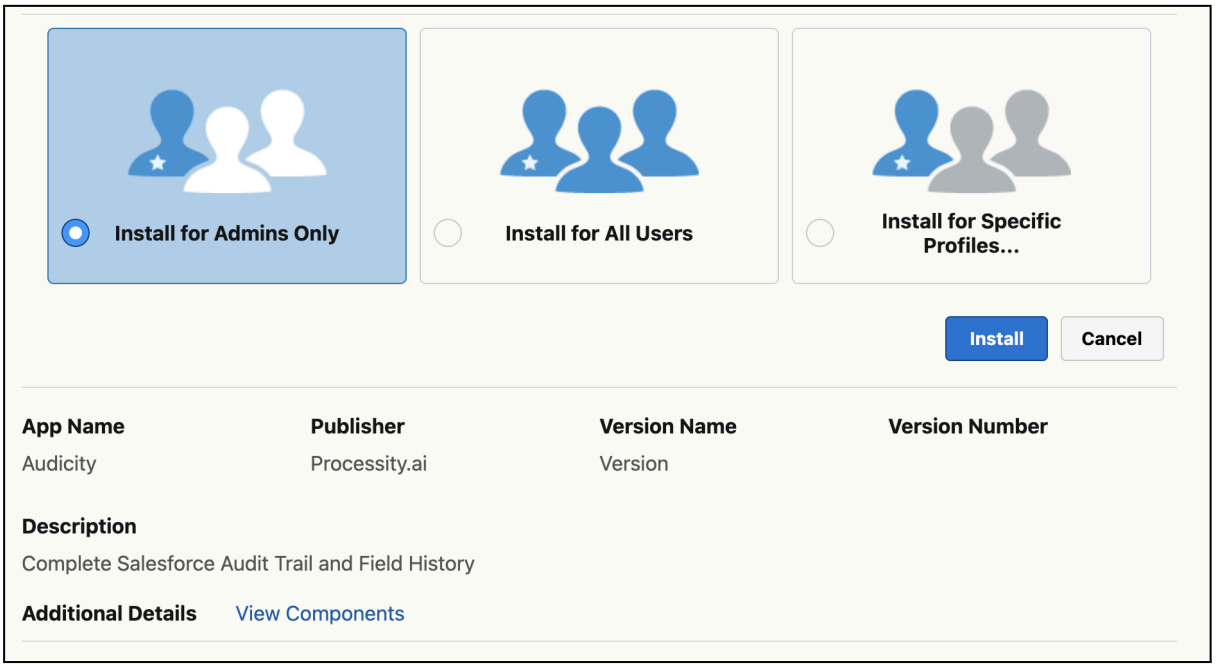
4. Choose the “Install in a Sandbox” option:



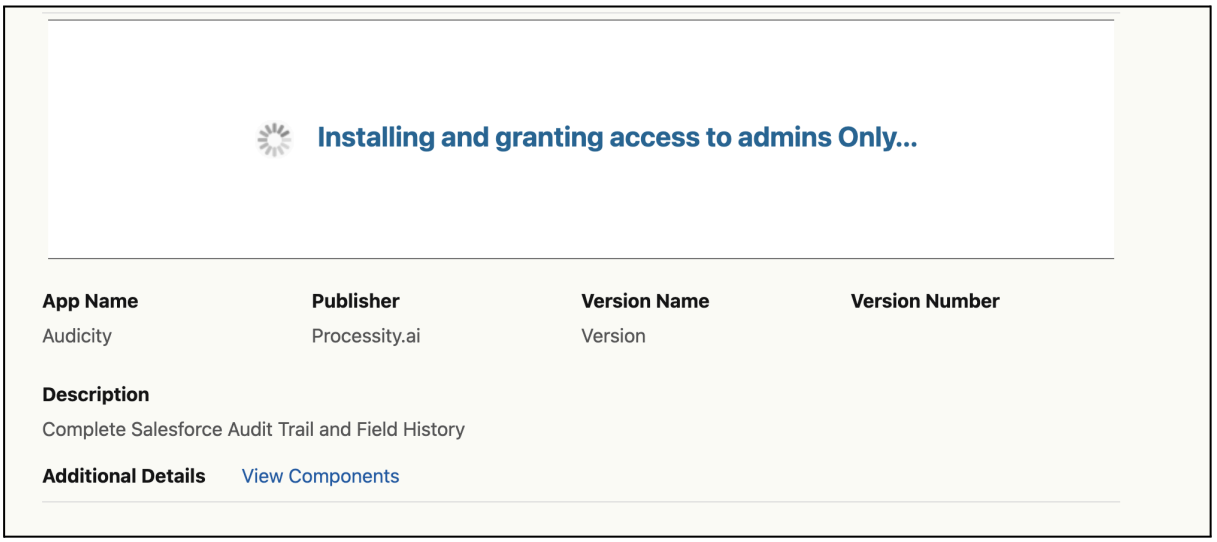
5. You will be taken to the sandbox login screen - this is how you choose which sandbox to install into:



6. Choose the option to “Install for Admins Only”:



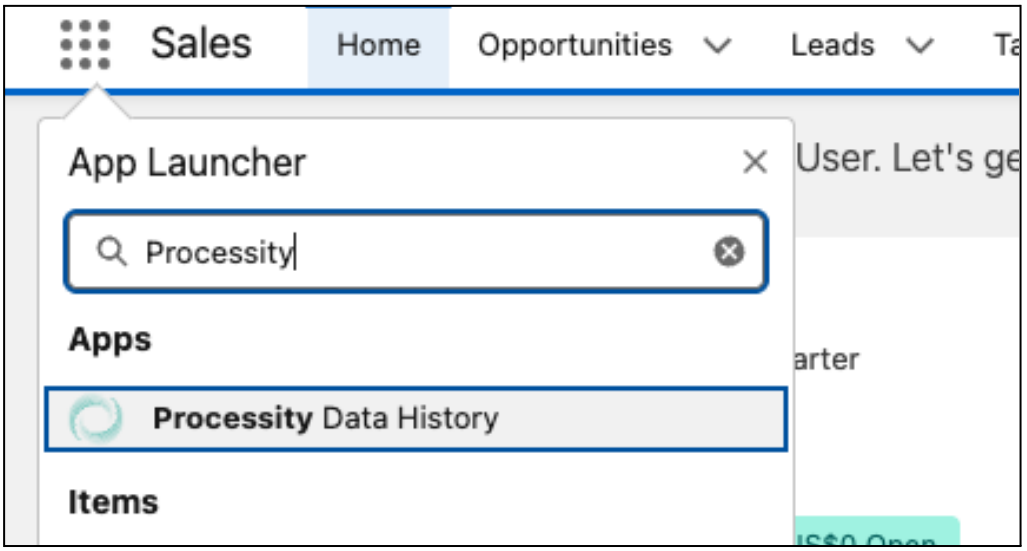
7. Initiate the installation by clicking “Install”.



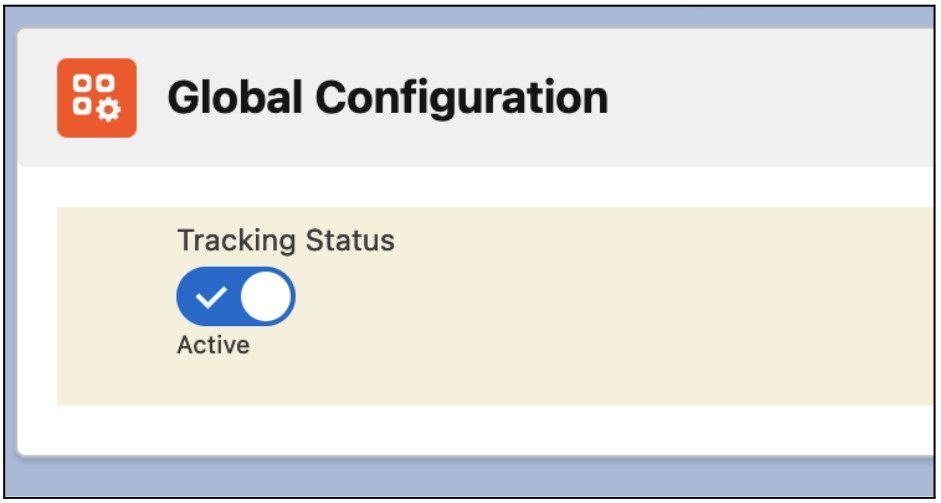
Configuration

In these configuration steps, we will enable Processity Data History and configure it to track all fields on the Account object.

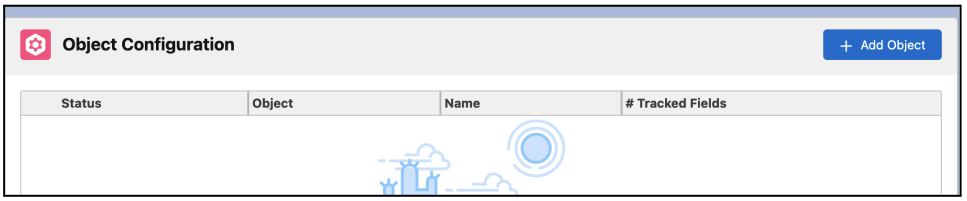
1. Using the App Launcher, navigate to the Processity Data History app:



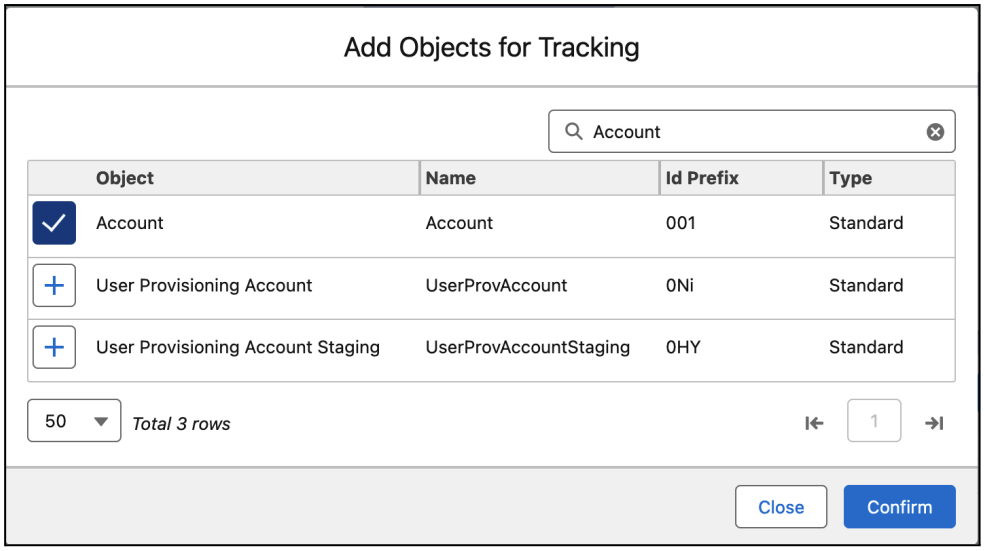
2. On the Processity Data History Overview screen:
   a. Set Tracking Status to “Active”:



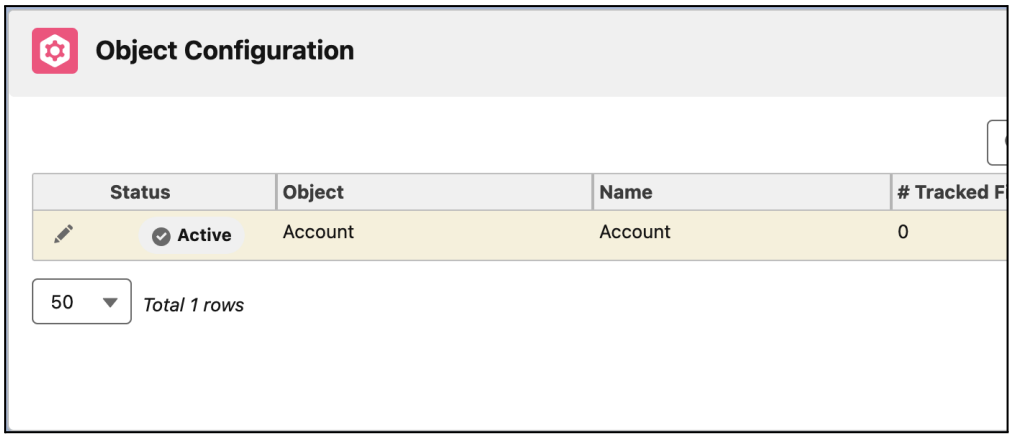
   b. Use the “Add Object” button to open a new dialog:.



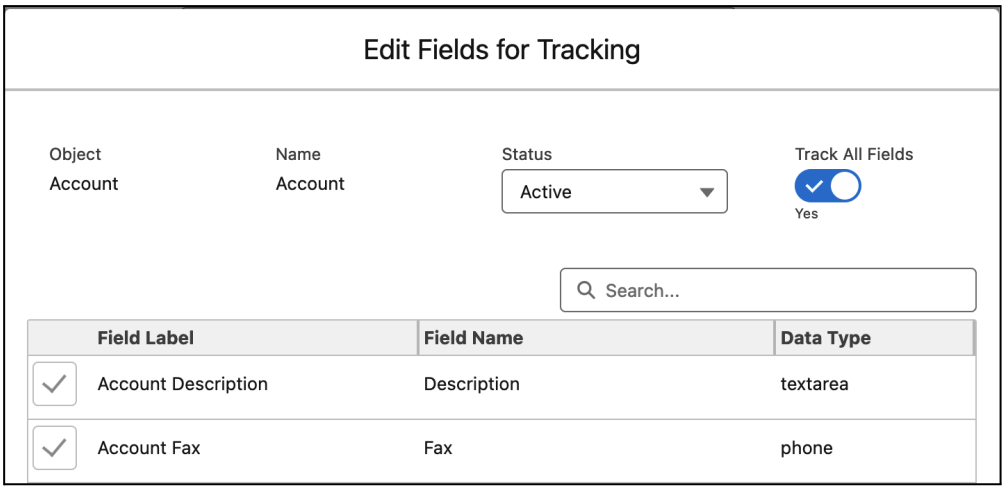
   In the dialog, search for Account and then use the + button to add the Account object. Close the dialog by clicking “Confirm”:



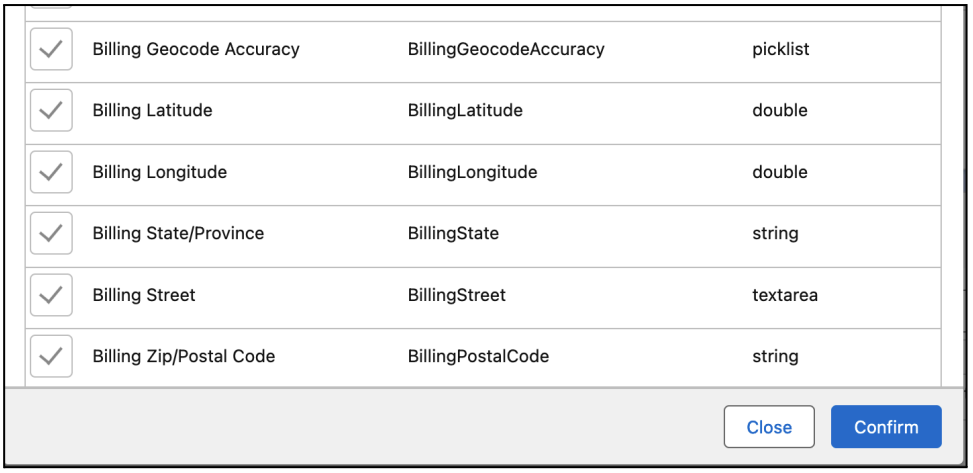
c. Click on the pencil icon next to Account in the Object Configuration section:



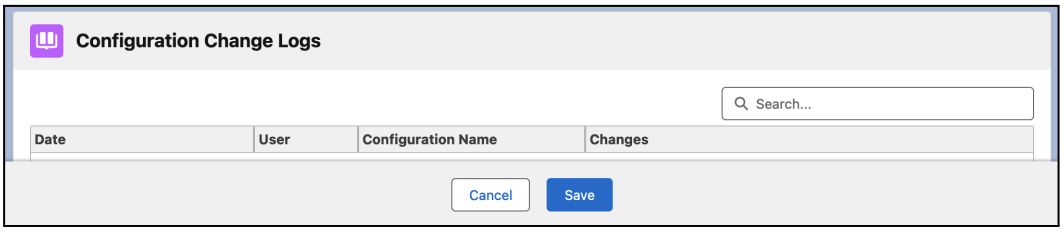
Set Track All Fields to “Yes”:



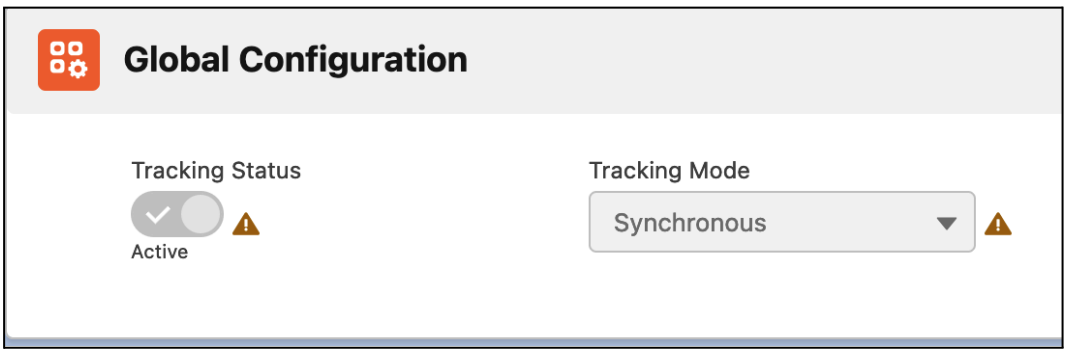
And then click “Confirm”:



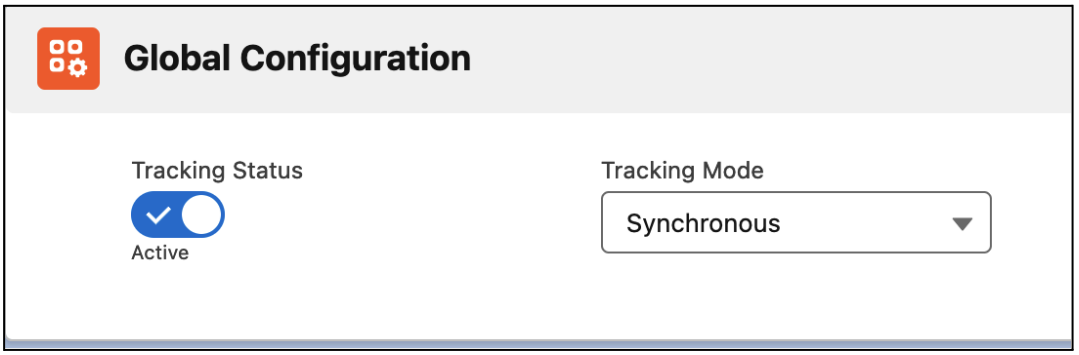
3. Finally, click “Save” at the bottom of the main Processity Data History screen. **None of the changes are saved until you click “Save” button.**



4. It may take a few minutes to save the changes. While they are saving, you will see a warning icon next to the items that you changed:



And when the changes are saved, the warning will disappear:



Congratulations, you have configured Processity Data History on the Account object!

But wait, there are two more things to do before you can see Processity Data History audit trails in your sandbox…

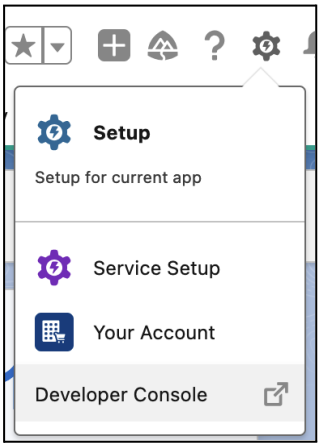
Add Apex Triggers

To capture the changes in your system, Processity Data History's Apex code must be called at the beginning and end of each transaction where a record is saved.

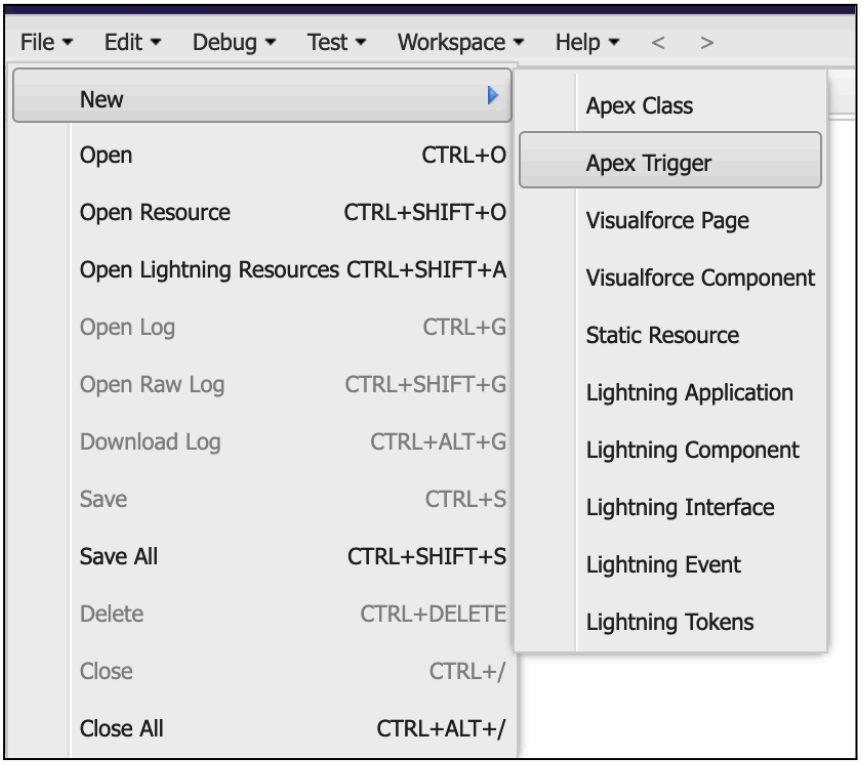
For best results, Processity Data History should be used with a trigger handler framework (wondering what a trigger handler framework is? Then skip to the FAQ section at the end of this document). Such a framework forces Apex triggers to run in a predictable order, and allows you to make sure that Processity Data History runs as the first thing and the last thing in each transaction.

In this Quick Start Guide, we will add Processity Data History as if there is no trigger framework. See the User Guide (available on the Processity Data History AppExchange listing) for more details on using Processity Data History with a trigger framework, and work with your development team to set that up.

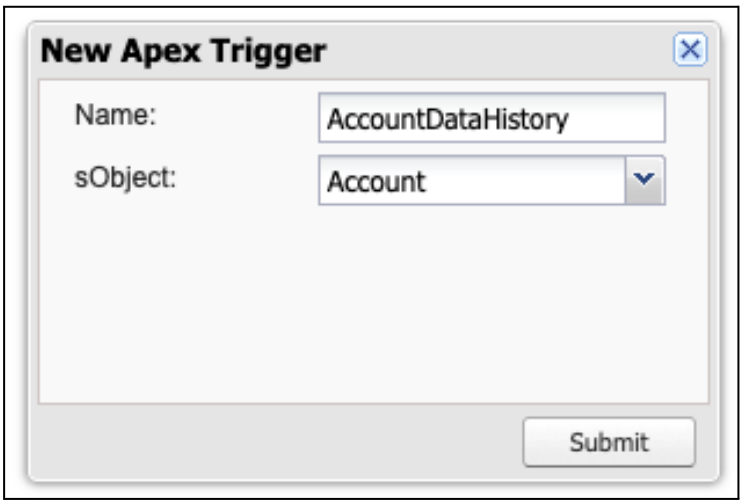
1. From the standard Setup menu, select "Developer Console":



2. In Developer Console, select File → New → Apex Trigger:



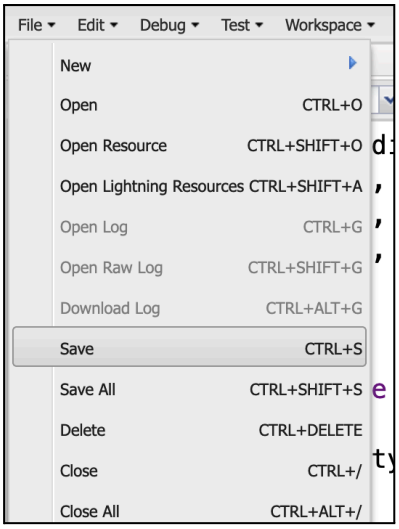
3. Call the trigger “AccountDataHistory” and choose “Account” as the sObject. Then press “Submit” to create the trigger:



4. Copy and paste the following code into the trigger:

```
None
trigger AccountDataHistory on Account (
       before insert,
       before update,
       before delete,
       after insert,
       after update,
       after delete,
       after undelete
) {
       mantra.AudicityApex.track();
}
```

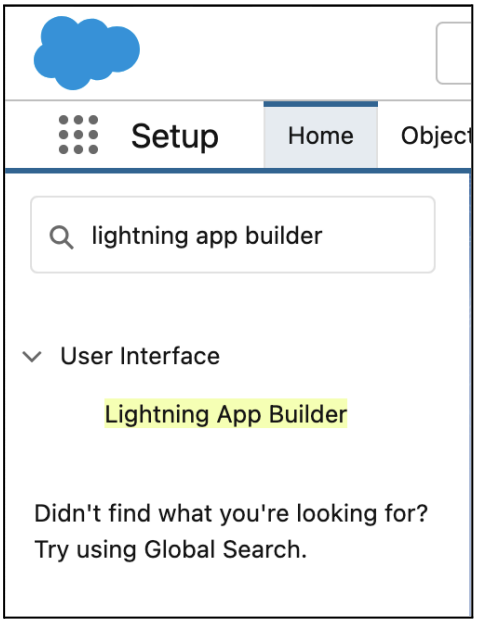
5. Save this trigger using File → Save:



Now you have fully configured Processity Data History to collect data about changes to the Account record. All we need to do now is make the Processity Data History data visible to you when viewing Accounts…

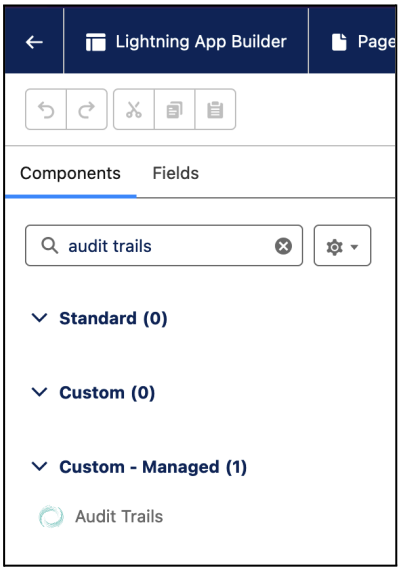
Add Audit Trails Component to Record Pages

1. In Setup, open “Lightning App Builder”:

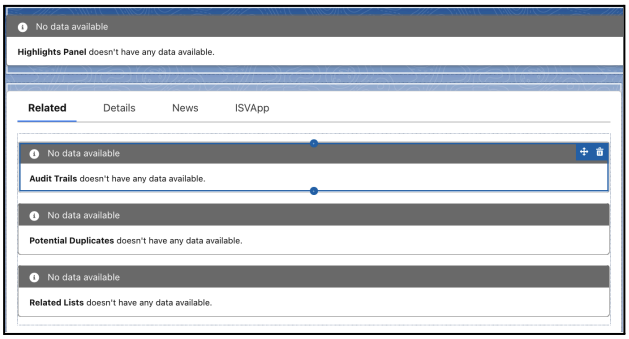


2. Find and edit your Account Record Page. The details of this can vary greatly and will depend on your particular Salesforce org. There may be an existing page, or you may need to create a new one.

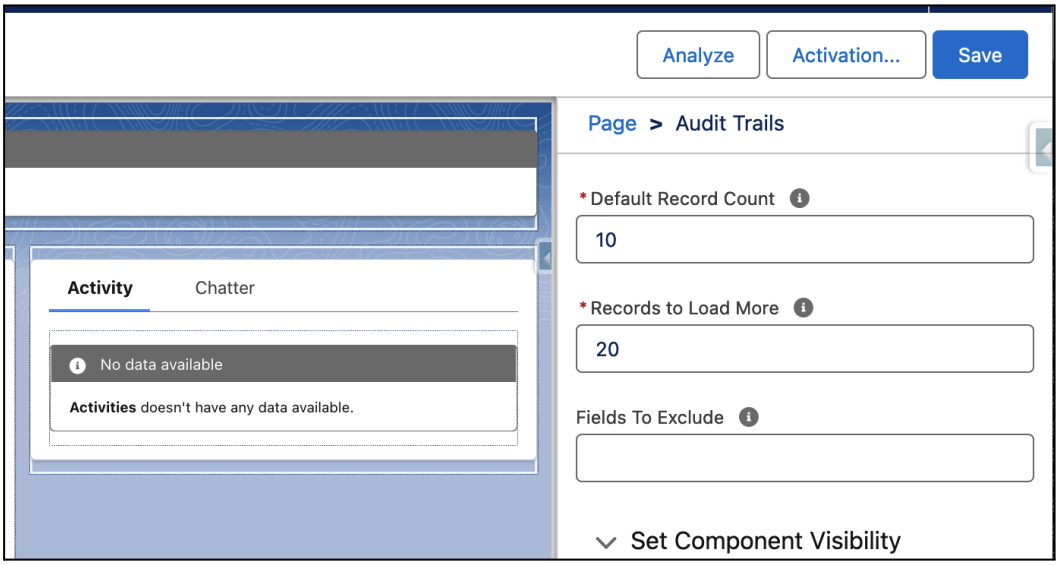
3. In Lightning App Builder for your Account Record Page, search the component list of “Audit Trails”:



4. Drag that component onto the page, next to your other related lists:



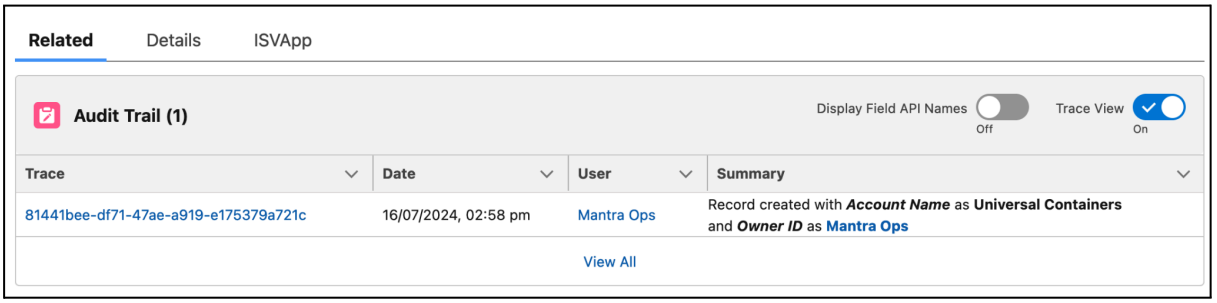
5. Save the Record Page (if you created a new page, then Salesforce will prompt you to activate it - you will need to do this to make sure that the new page is used when you view records):



Make a change and see it tracked!

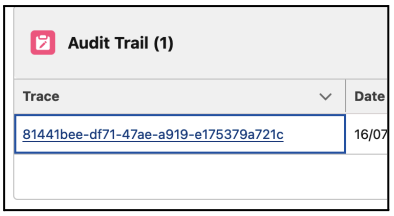
Now you're ready to create a new record.

1. Create an Account in the way you normally would
2. After creation, navigate to the place on the Record Page where you added the Audit Trails component:

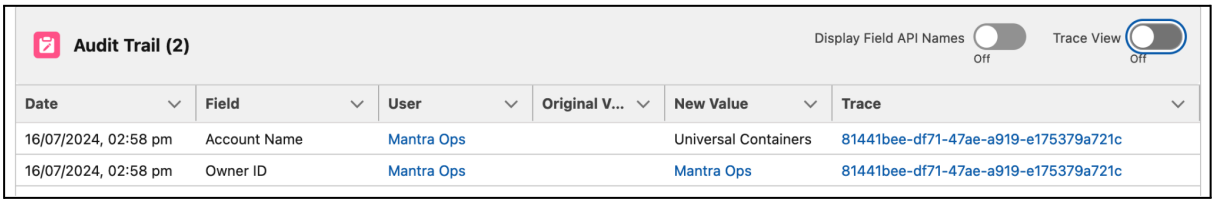


You will see all the fields that you set, and anything set by automation, listed together in the Audit Trails component.

3. Click on the link in the “Trace” (what is a Trace? See the FAQ section for details) column to see an expanded view of everything that happened within the trace of a particular change:



4. Click on the “Trace View” button to ungroup the changes and get a view that is more like standard field history tracking with one change per line:



Next Steps

Some things that you might consider doing next with Processity Data History:

Track More Objects

By revisiting the configuration page, you can enable tracking on more objects. Don’t forget that with each extra object, you will also need to add an Apex trigger.

Add Permissions For Other Users

As the administrator who installed the package, you have access to Processity Data History and your changes will be tracked. Processity Data History supports a number of permissions and personas to enable tracking, viewing of tracking data, and access to configuration.

Write and/or run Apex tests

The Apex trigger that you wrote above should be tested against your existing customisations. You can do this by running all of your existing Apex tests. And, if necessary, writing a test specifically for that trigger.

Enable Tracking Across Asynchronous Transactions

Processity Data History can track more than just what happens in a single transaction. If you update your Apex code to notify Processity Data History when it enqueues Queueable Apex, it is able to connect up those separate transactions into a single trace.

Read The User Guide

Processity Data History's User Guide contains more details about how to take the Next Steps listed above and about some of the choices you can take during the installation, configuration, and use of Processity Data History.

Troubleshooting

If you have any questions or difficulties, then please contact us at support@processity.ai

FAQ

What is an Apex trigger handler framework?

A Salesforce object can have more than one Apex trigger attached to it. The default behaviour is that there is no predetermined order to those triggers. Nor is it guaranteed that the trigger order will stay the same over time.

So, instead of leaving things to chance, Salesforce recommends using a trigger handler framework.

Many trigger handler frameworks exist, but they all share a common theme: you create a single trigger on each object, and then the framework manages the order of executing all of your code after that. Some frameworks do everything in Apex, some use Custom Metadata Types to manage which code to run.

While many frameworks offer other features (like being able to selectively disable triggers), they all allow you to pick the order in which code runs.

So, making use of a trigger handler framework, in combination with Processity Data History, is ideal because you can call Processity Data History once as the first event in the save cycle and once again at the end. This will give you the most detailed and accurate information about the context around the changes on your records.

What is a Trace, and how is it different from a transaction?

Trace is not a term commonly used in Salesforce, but it is used more widely when people are interested in observing what is happening in systems.

A Trace is a record of all the things that happened to fulfill a user action. It can span multiple systems (Processity Data History traces are currently confined to Salesforce), and it can span multiple transactions (Processity Data History does trace across multiple transactions).

For example, when an Opportunity is set to Closed Won by a user, many things may happen as consequences of that.

There is the initial transaction where the Opportunity Stage changes along with some other fields, and perhaps some other automation. And then there are usually other consequences. e.g. notifying a warehousing system outside of Salesforce.

In Salesforce, an API call to the warehouse system cannot happen within the same transaction, so it happens in a later asynchronous transaction. This means that from one user action, there are two transactions (the one where the Opportunity became Closed Won, and the one where the API call is made to the warehouse system) but both of these are part of the same trace because they are both caused by the same user action.