Common Data and Integration Designer

Overview

The Data and Integration Designer, also called Designer case, uses a case-based approach for extending the data model, automating workflows for adding new entities and properties, adding relationships, and integrating with external sources. The Designer case has the following stages:

1. Create – used to create new entities.
2. Extend the model – Helps in understanding the existing model and to add fields and modify views.
3. Manage relationships – use to create a new relationship between any two entities or extend the existing relationships.
4. Manage integration – Helps with connecting to external systems.
5. Review configuration – Helps in mapping the fields for integrations.
6. Manage sample data – Helps with managing sample data for entities and relationships.
7. Test entity – Test the entity data pages and their profiles.

When you create a new entity using the Designer case, the following rules are automatically created:

CLASS: Entity and all connector related classes are generated

CONNECT REST: Connect rest rule with the name of the entity is generated for the new entity

DATA PAGES: Two Data Pages are generated for the entity: one page mode and one list mode

SECTION: Different types of sections such as the entity header, content, and other display-related sections are generated

HARNESS: A harness for each section is generated

ACTIVITY: Activity for the processing of each type method of connector such as GET, POST, and PUT

DATA TRANSFORM: All the request and response Data transforms for different connector methods are generated. Search-related Data transforms are also generated

PROPERTY: Entity class ID and class name properties are generated

WHEN rule: When rules for different profiles of Data Page conditions are created

SERVICE REST: Service rest rules for the entity are created

LOCALIZATION: rules

VIEW: rules

Using a Designer case, Connect Rest rules can be created for external integration, thereby allowing you to add new connectors to Data Pages and test the entity within the Designer case. You can map these connectors to existing Data Pages and select the appropriate profile for mapping.

Prerequisites for using Common application

Whenever a new implementation application is created, there are certain configurations to be performed before Common app is ready to be used. These configurations are automated in release 25.1 when the configuration called Implementation automation under Common-General Settings is set to **true.** In any other cases, for example if implementation automation is set to false, or if the release is prior to 25.1 i.e., 24.2, these should be manually created.

- Create the DCR toggle. Navigate to Configure-->System-->Release-->Toggles and create a new one with your application extension suffix (for example CDM_UPlus) and enable this toggle.
- Create a new Authentication profile. Records-->Security-->Authentication profile—> Create.
- Set up OAuth 2.0 client registration. Records-->Security-->OAuth 2.0 client registration --> Create.
- DetermineExtension_Ext decision table for extension. Add your application extension suffix in this decision table (Use the same name as the toggle)
- DetermineApplicationTag_Ext for application tag
- Extension data transforms for DCR – ClassSettings, GeneralSettings, ReportSettings, ApiRetrievalSettings, SetCDMAuthenticationProfile.

Installing the Designer case:

The Common Data and Integration Designer case will be shipped as part of the Pega Common Application starting from release 25.

If you are on a version earlier than release 25 or wish to use the latest updates of the Common Data and Integration Designer case, follow these steps:

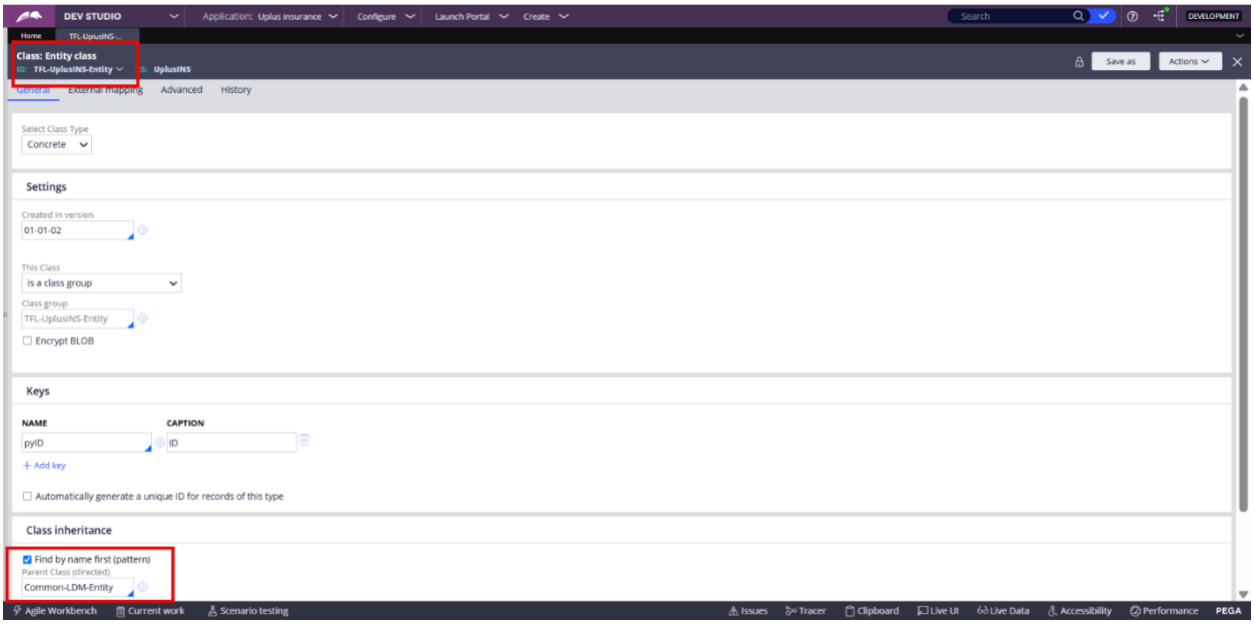
1. In the header of Dev Studio, click **Application > Definition**.
2. On the **Edit Application** page, in the **Enabled componen**ts section, click **Manage components**.
3. On the Pega Marketplace page, click Browse apps & components, and search for **Common Data and Integration Designer**.
4. **Download** the latest version of the component.
5. In Dev Studio, on the Pega Marketplace page, click **Install new**, and select the downloaded file from your Downloads folder.
6. In the Status column, next to the Common Data and Integration Designer component, click **Enabled**, then click **OK**.
7. Click **Close**.
8. Save the application rule to apply the changes.

Prerequisites for using the Designer case:

Before getting started using the Designer case, the following checks and configurations must be performed. Note that this is a one-time configuration and need not be performed every time the designer is run.

1. Create the entity class that the Designer case will automatically create new entities into. By default, the entity class will be shipped as "**Common-LDM-Entity**". If you wish to create the entities within this class, this step can be skipped. If an org-specific class is to be used, then create a new class having the directed inheritance to Common-LDM-Entity. For example, if the entity class is "TFL-UplusINS-Entity", it should be **inherited** from Common-LDM-Entity.

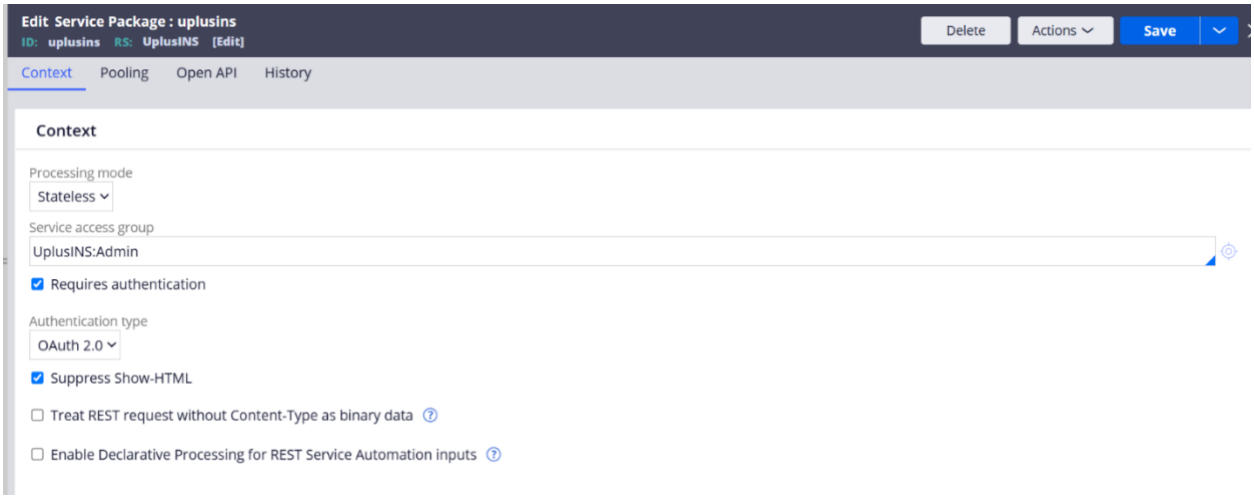
    a. The class type should be concrete.
    b. The class inheritance should be pointed to "Common-LDM-Entity".
    c. Ensure that the entity class is merged into a ruleset. It should not be present in a branch.
    d. When the class is created, a history class will be autogenerated. If the history class was not autogenerated, manually create a new history class and inherit it to "History-Common-LDM-Entity". For example, if the history class is "History- TFL-UplusINS-Entity", it should be inherited from "History-Common-LDM-Entity"



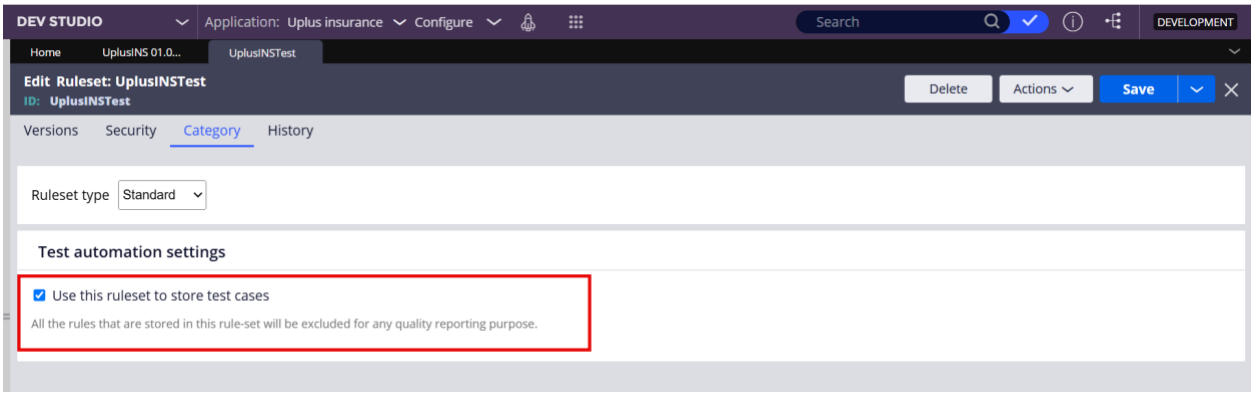
2. A service package is a data instance for a collection of REST services. Specify the appropriate service package that is present in the application. All the REST services

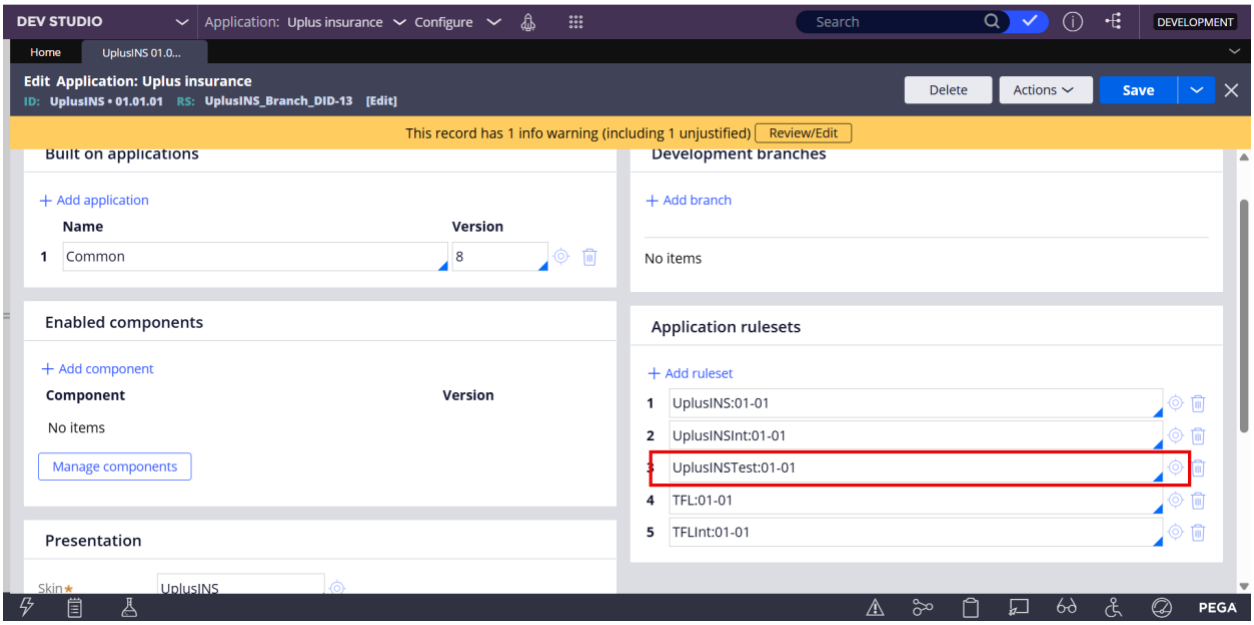
the Designer case generates are stored under this service package. If a service package is not already present in the application, create a new one.

a. Records→ Integration-Resources→ Service package
b. Provide the access group and change the Authentication type to OAuth 2.0 and save the rule.

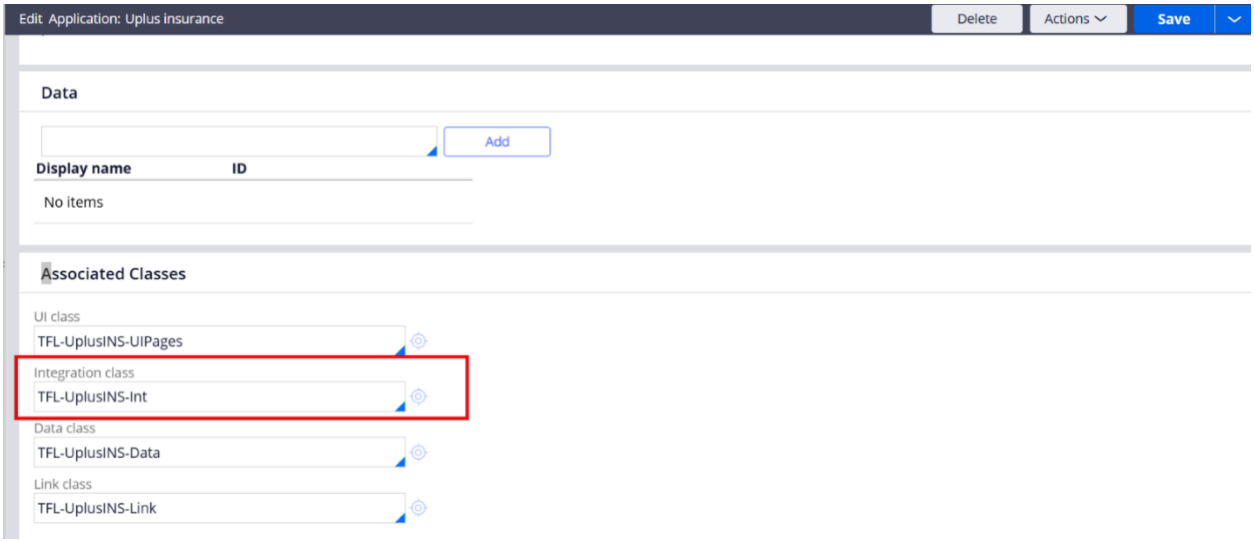


3. Optional: Create a test ruleset if it doesn't already exist. Having a test ruleset ensures that the rules needed to enable sample data generation are autogenerated using the Designer case. If a test ruleset is not provided, the rules required to create sample data should be manually created by the user. Sample data rules are currently only supported if the entity class is "**Common-LDM-Entity**".
    a. Create a new test ruleset "UplusINSTest" and ensure that the check box to store test cases is enabled and saved (verify this under category tab of the ruleset). Lock the ruleset.
    b. Add the ruleset to the application ruleset stack and save the app rule.





4. Ensure that all the rulesets are locked in the application.
5. Ensure that the integration class in application rule is not empty, this class will be used to generate all the required integration assets. Click the **Cases & data** tab on the application rule to check this.



6. Add configuration values for the Designer case. In the navigation pane of App Studio, click **Settings → Configurations →Configuration set: Common – Designer Settings**.
   NOTE – The configuration values for this application will be saved as an instance under Data-Configuration-Setting

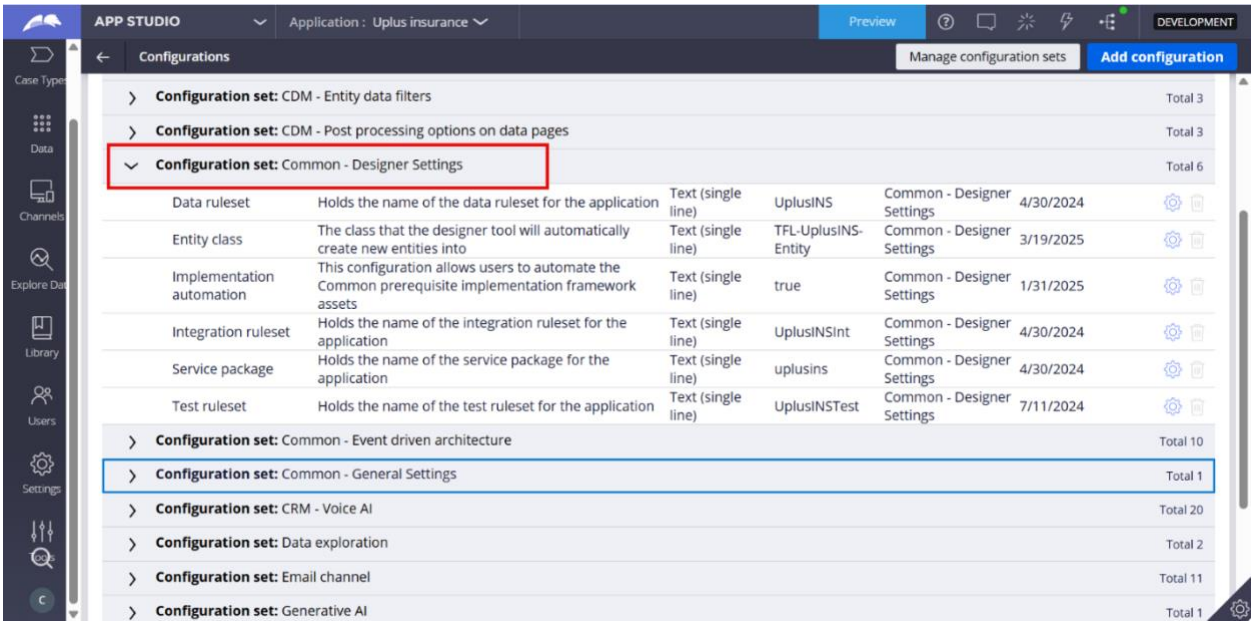
    a. Data ruleset – Specify the primary Ruleset of the application. All the core rules that are generated through the Designer case are stored in this primary Ruleset.

    b. Integration ruleset – Specify the integration ruleset of the application, All the integration- layer rules and integration classes rules that are generated through the Designer case (which are used for External SOR/Local SOR integrations) are stored in the integration ruleset.

    c. Service package – A service package is a data instance for a collection of REST services. Specify the appropriate service package that is present in the application. All the REST services generated through the Designer case are stored under this service package.

    d. Entity class – Provide the name of the entity class that the Designer case will automatically create new entities into.

    e. (Optional) Test ruleset – Provide the name of the test ruleset where all the rules for sample data for the entity and relationship will be created. This step is optional. Sample data rules are currently only supported if the entity class is "**Common-LDM-Entity**".

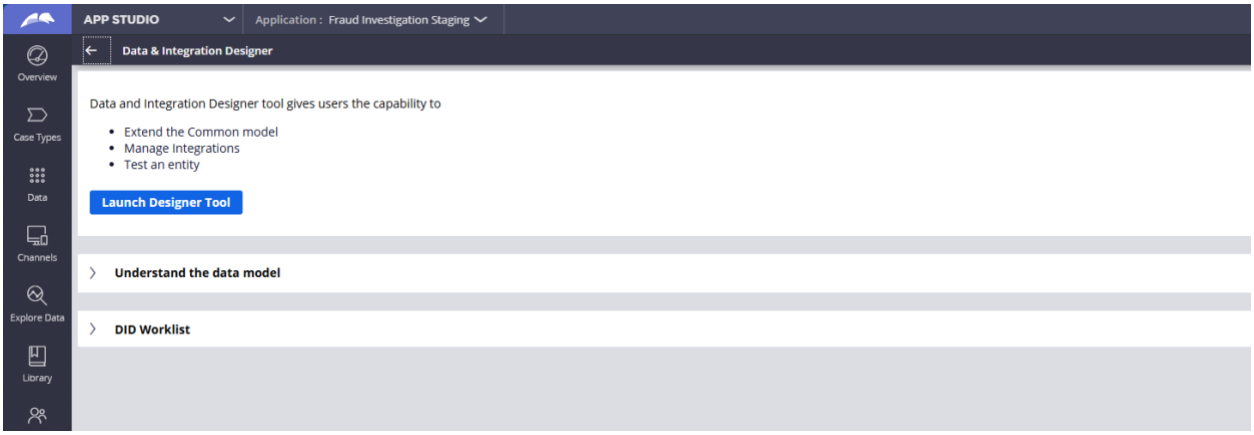


8. Log off and log in to the application to pick up the configuration changes.
9. Enable branch development, click on the toggle for enabling branch development and create a new branch.

   Note: It is recommended to use an empty branch for easier maintenance and to avoid any other issues.

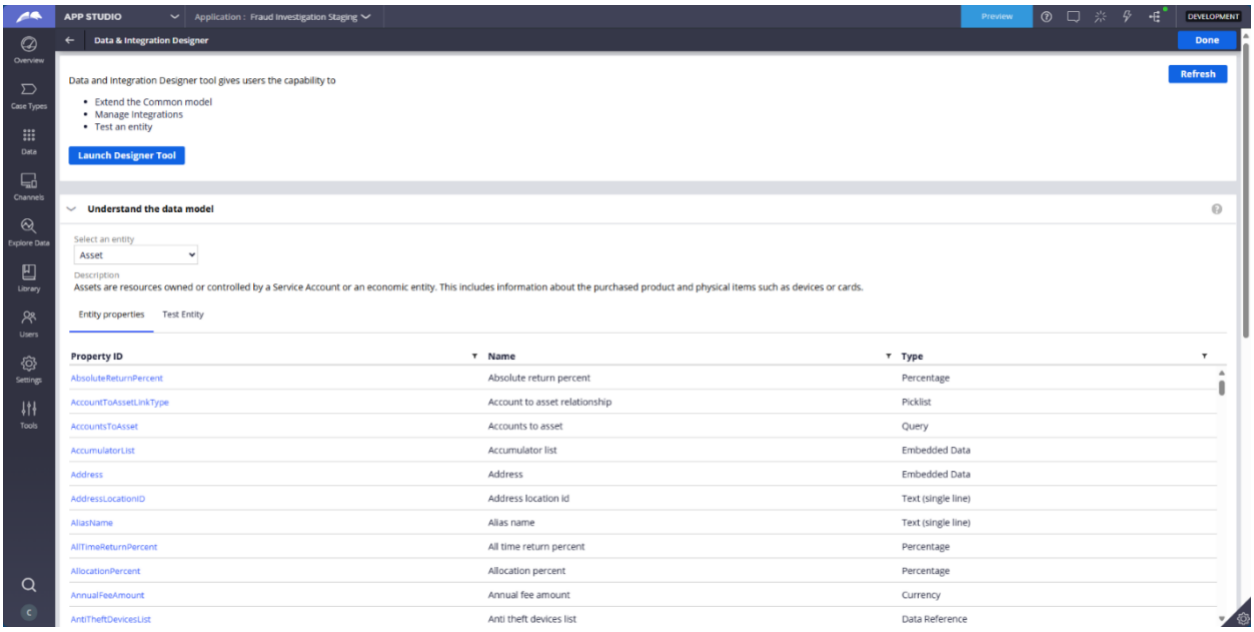
Designer landing page

Navigate to the Designer landing page through **App Studio → Tools → Data and Integration Designer**, where you will find an option to launch the Designer case. Additionally, you can explore the data model and view the list of Designer cases in the DID Worklist.



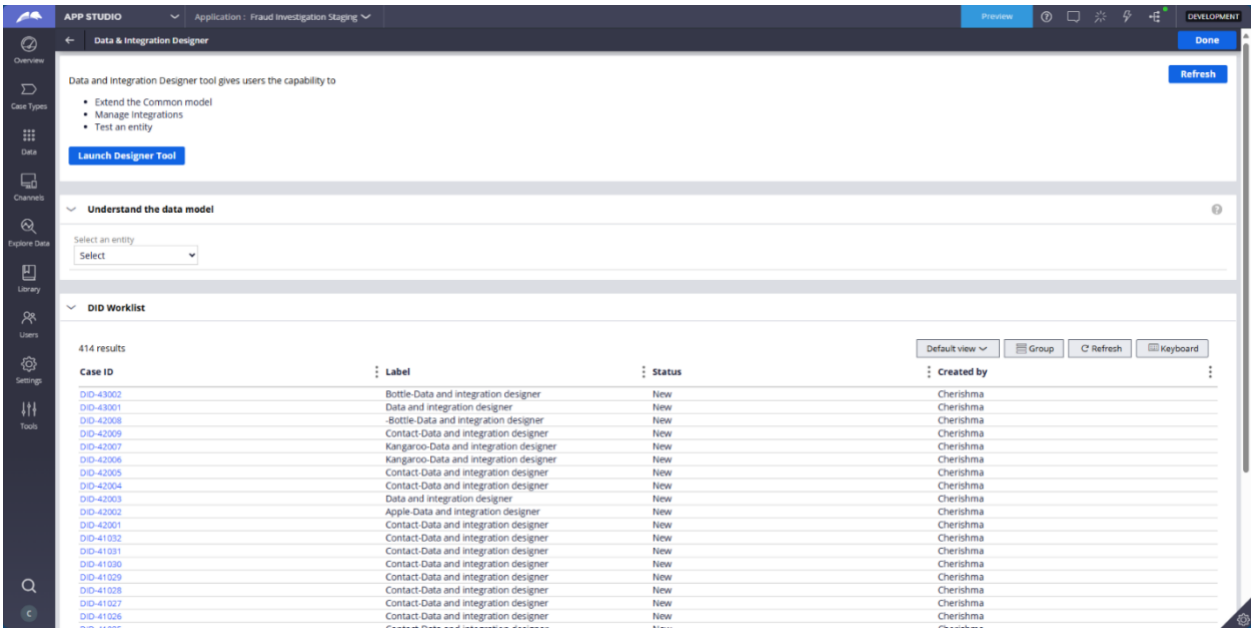
Understand the data model

This feature allows users to explore existing entities by viewing their properties and related information. Users can also test an entity's data pages and integrations.



Designer worklist (DID worklist)

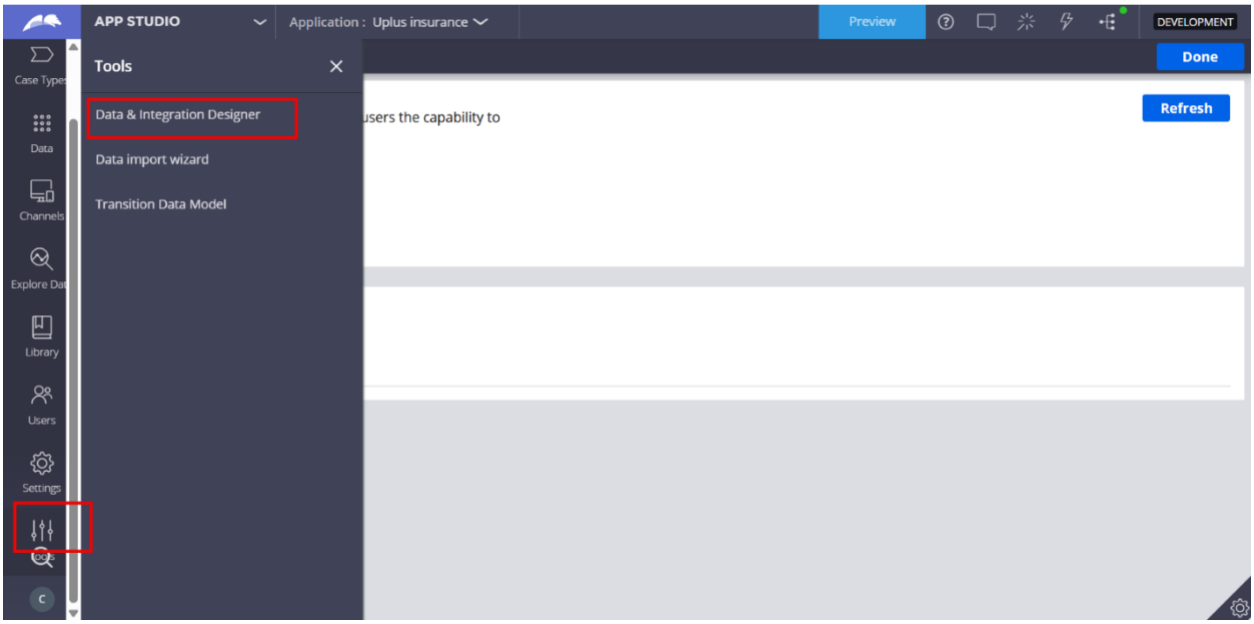
This is a list of all Designer cases, which you can open and resume from where you previously left off.

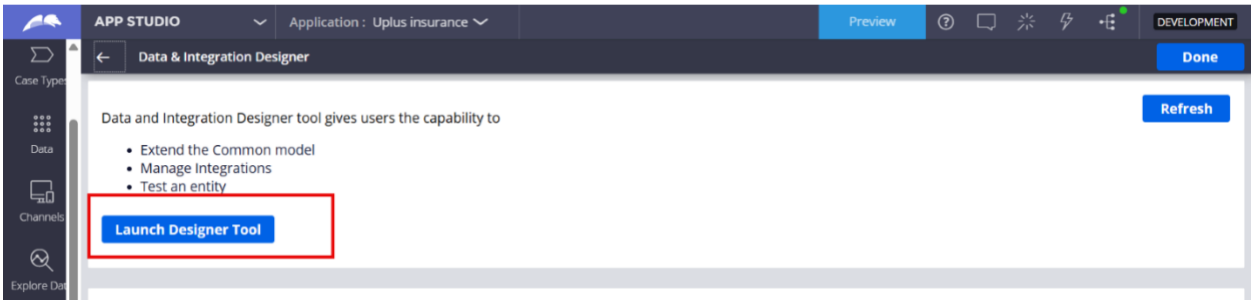


Using the Designer case:

Note – before beginning, ensure that branch development is enabled.

Navigate to App studio→ Tools→ Data & Integration Designer → Launch Designer Tool.

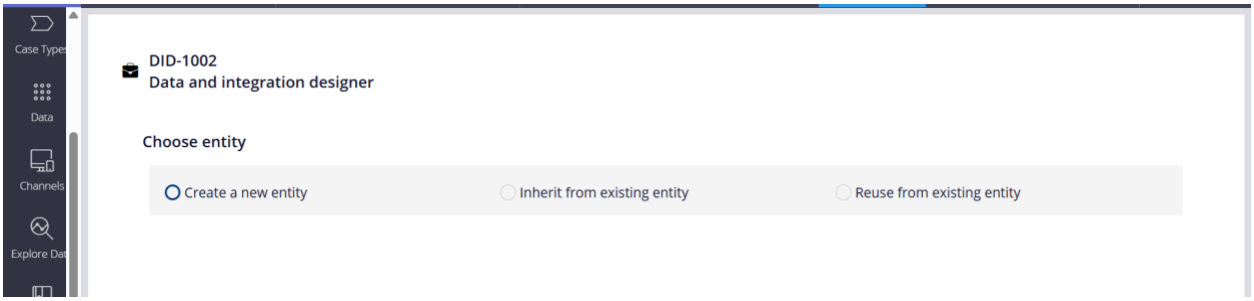




Create stage

After launching the Designer case, choose from one of the 3 options:

1. Create a new entity – used for creating new entities in the application
2. Inherit from existing entity - used for creating entities that inherit from available entity classes in the application.
3. Reuse from existing entity – used for selecting existing entities and performing other actions like adding fields, managing integrations, creating relationships, etc.



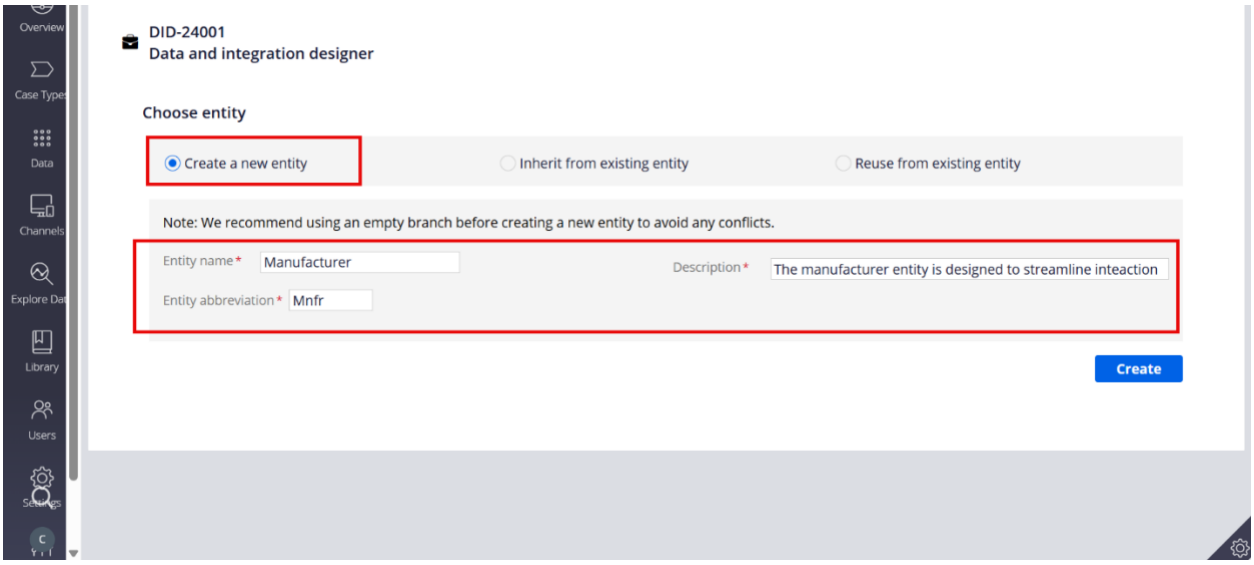
Create a new entity

When a new entity is created using the Designer case, all the essential rules will be automatically created. The class rule, List and savable data pages, connector rules, service REST rules, data transforms, default views, database table, etc.

As an example, if Manufacturer is a new entity being created in the UPlusINS application and the entity class provided was TFL-UplusINS-Entity, then the new entity created would be "***TFL-UplusINS-Entity-Manufacturer***".

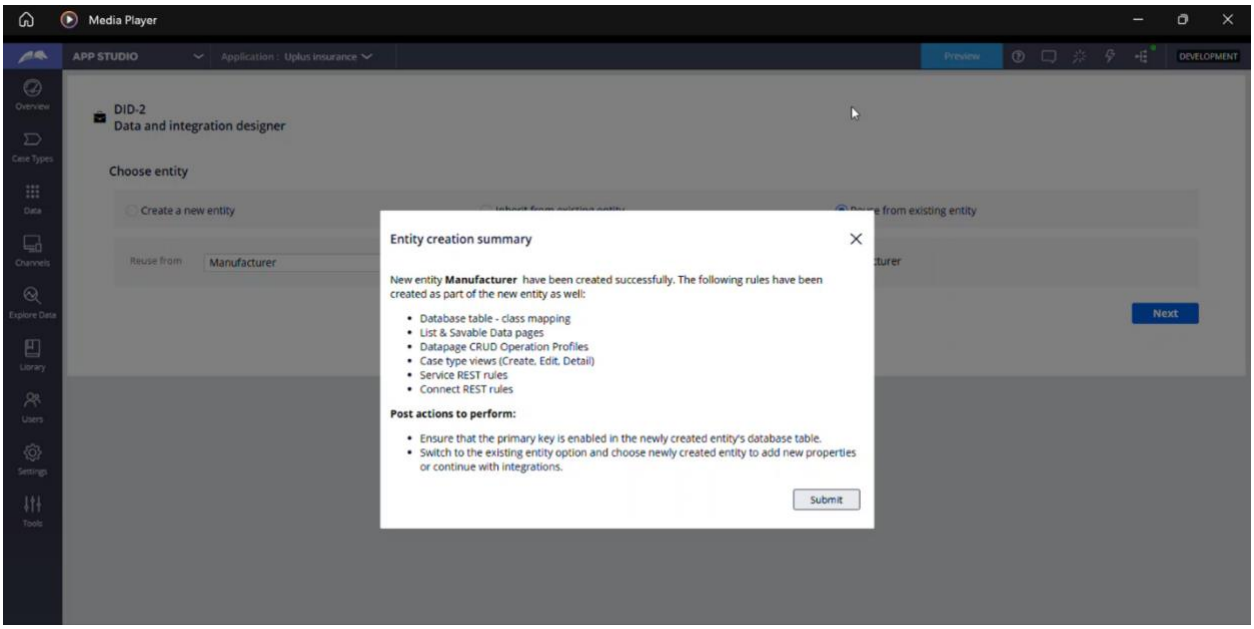
Steps:

1. From the 3 options, select **Create a new entity**
2. Provide the new **Entity name**, a **Description** and an **Entity abbreviation** to be used when creating relations, data transforms, etc.



3. Then click **Create**.

Once all the rules have been generated, it displays a success modal listing the types of rules generated.



4. Click on **Submit** to proceed to the next steps which are creating the properties, editing views, managing integrations and relations, etc.

To verify the rules generated, open the branch in Dev Studio and check for the details.

Inherit from existing entity

Inheriting from an existing entity inherits the capabilities of the parent entity. When inheriting, you can choose to either use the same name as the parent entity or use a different name.

Inherit using the same name

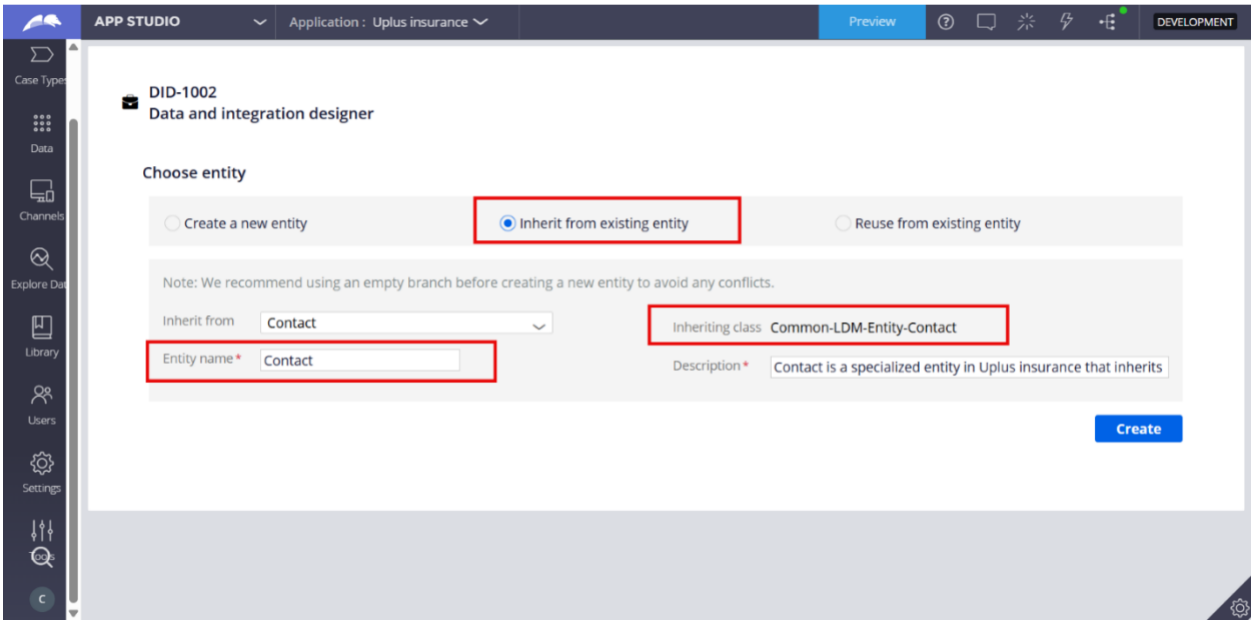
When an entity is inherited from an existing entity using the same name, all the capabilities of the parent entity can be leveraged.

A new entity class is created and the data pages and other essential rules from the parent class are saved into the new entity class. However, the integrations and services will **not** be newly created, rather the parent entity's rules are reused.

Steps:

1. From the 3 options, select **Inherit from existing entity**

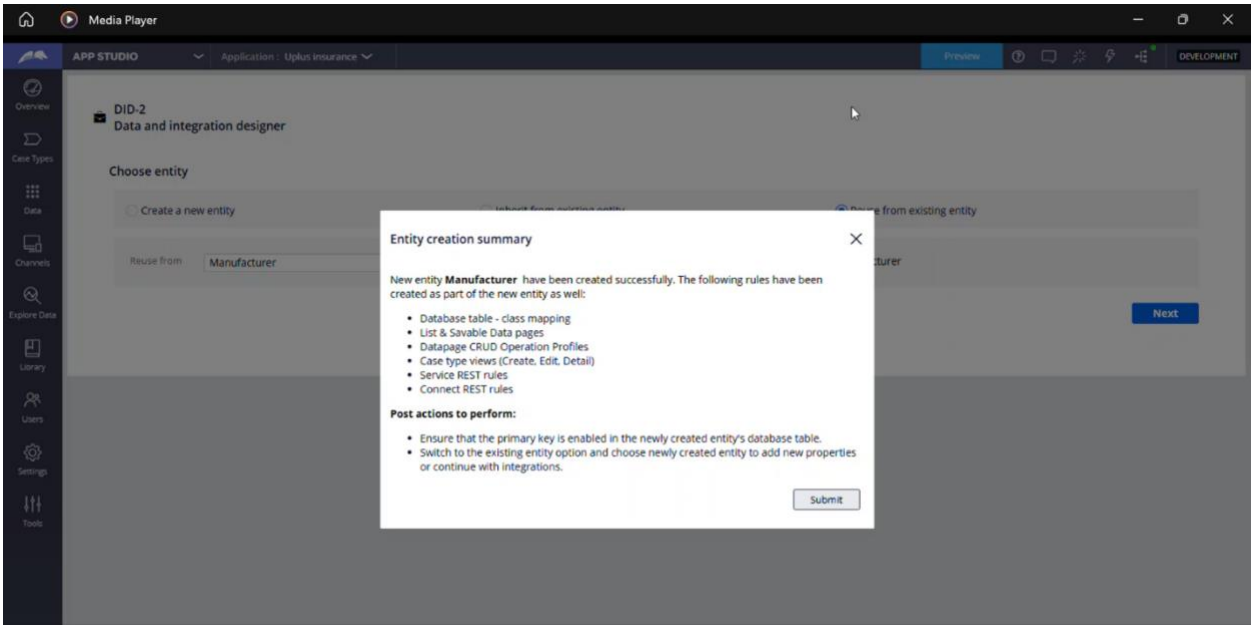
2. In the dropdown to **Inherit from**, choose the entity to inherit. Note that the entity name will be auto populated and should not be changed.
3. Provide the new **Description** for this entity



4. Then click **Create**.

Once all the rules have been generated, it displays a success modal listing the types of rules generated.

Note: All the properties from the parent class will be included in the new entity and they can be reused.



5. Click on **Submit** to proceed to the next steps which are creating the properties, editing views, managing integrations and relations, etc.

To verify the rules generated, open the branch in Dev Studio and check for the details.

For example, if creating a new Contact entity that inherits from Common's Contact entity using the same name results in: *TFL-UplusINS-Entity-Vehicle inherits Common-LDM-Entity-Asset*

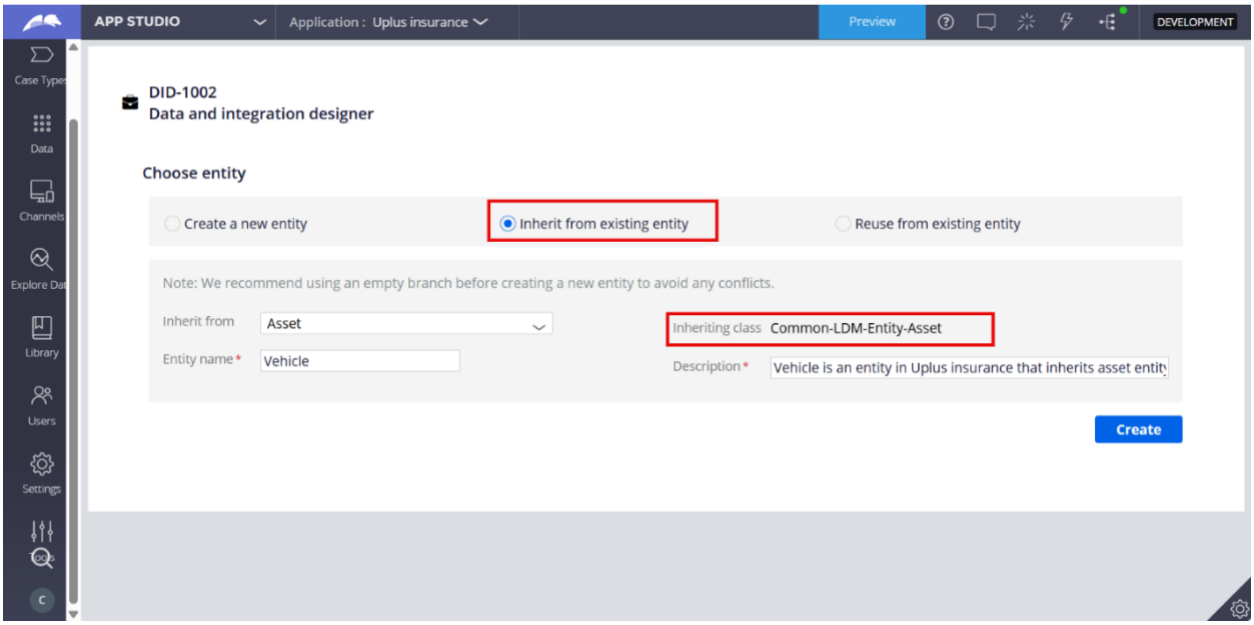
Inherit using a different name

When an entity is inherited from an existing entity using a different name, the capabilities of the parent entity can be leveraged.

A new entity class is created, and new rules are created for that entity like data pages, integration rules, views, etc. that inherit from the parent entity class.

Steps:

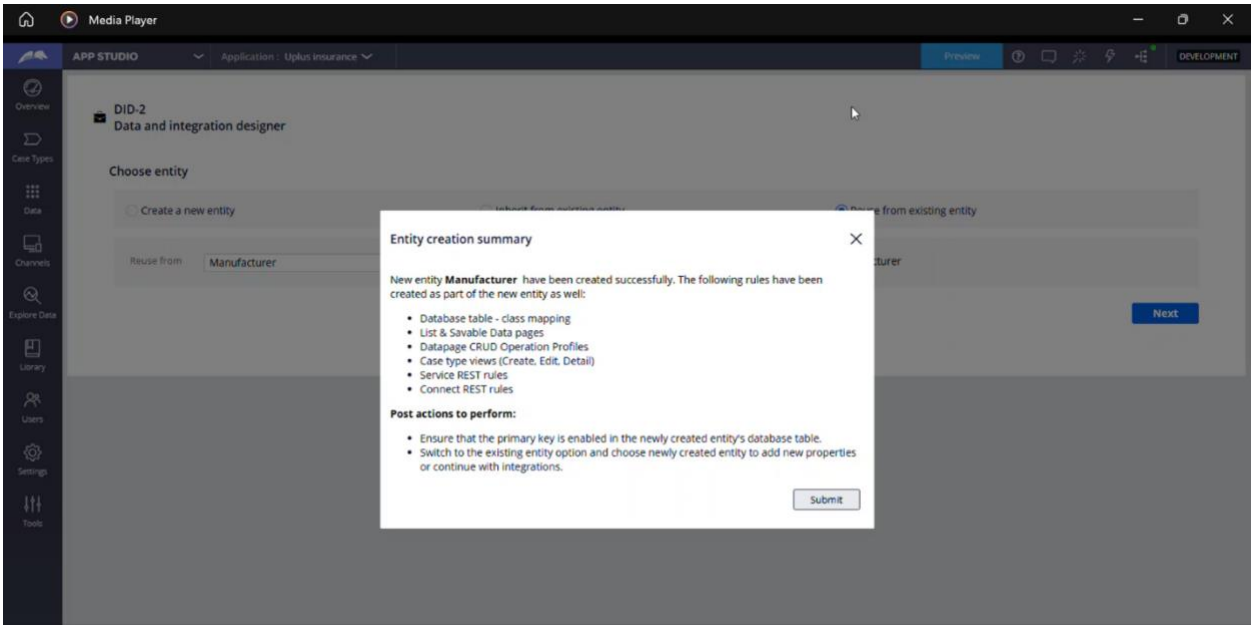
1. From the 3 options, select **Inherit from existing entity**
2. In the dropdown to **Inherit from**, choose the entity to inherit. Note that the entity name will be auto populated and should not be changed.
3. Provide the new **Name** and **Description** for this entity



4. Then click **Create**.

Once all the rules have been generated, it displays a success modal listing the types of rules generated.

Note: All the properties from the parent class will be included in the new entity and they can be reused.



5. Click on **Submit** to proceed to the next steps which are creating the properties, editing views, managing integrations and relations, etc.

To verify the rules generated, open the branch in Dev Studio and check for the details.

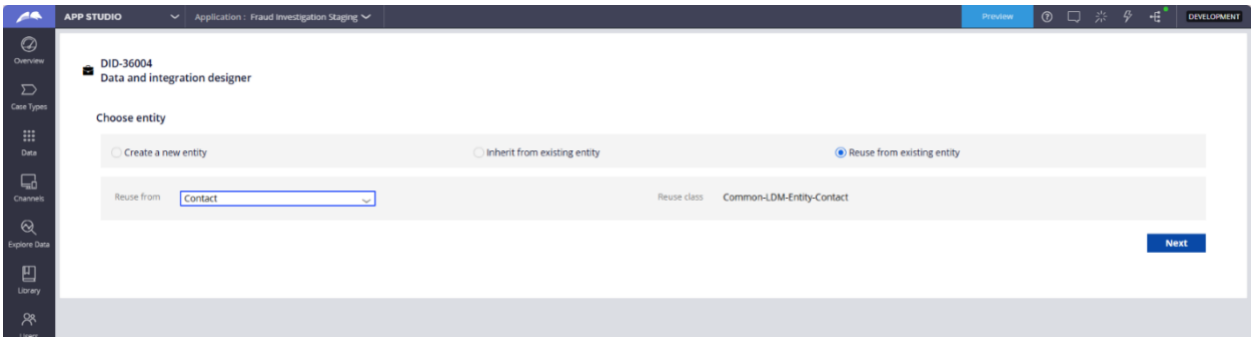
For example, if creating a new Vehicle entity that inherits from Common's Asset entity using a different name results in: *TFL-UplusINS-Entity-Vehicle inherits Common-LDM-Entity-Asset*

Reuse from existing entity

This option allows users to manage an existing entity by adding properties, configuring views, managing its relationships or integrations. Additionally, users can test the entity and manage its sample data template.

Steps:

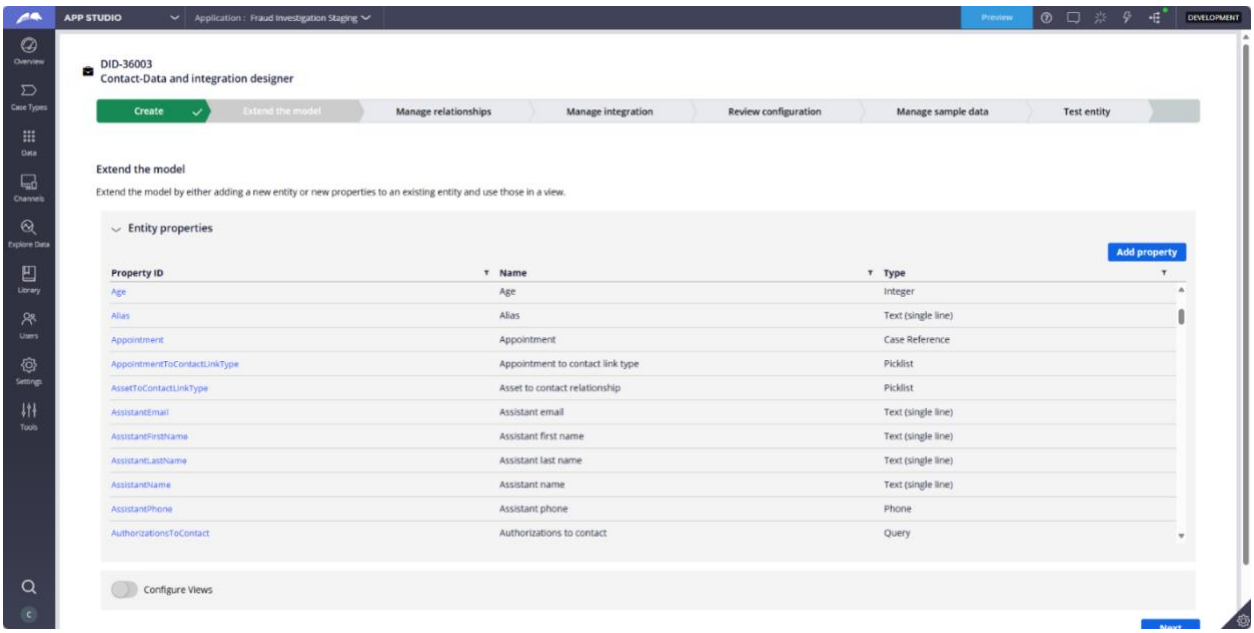
1. From the 3 options, select **Reuse from existing entity**
2. In the **Reuse from** field, choose an existing entity
3. Click **Next** to proceed.



Extend the model stage

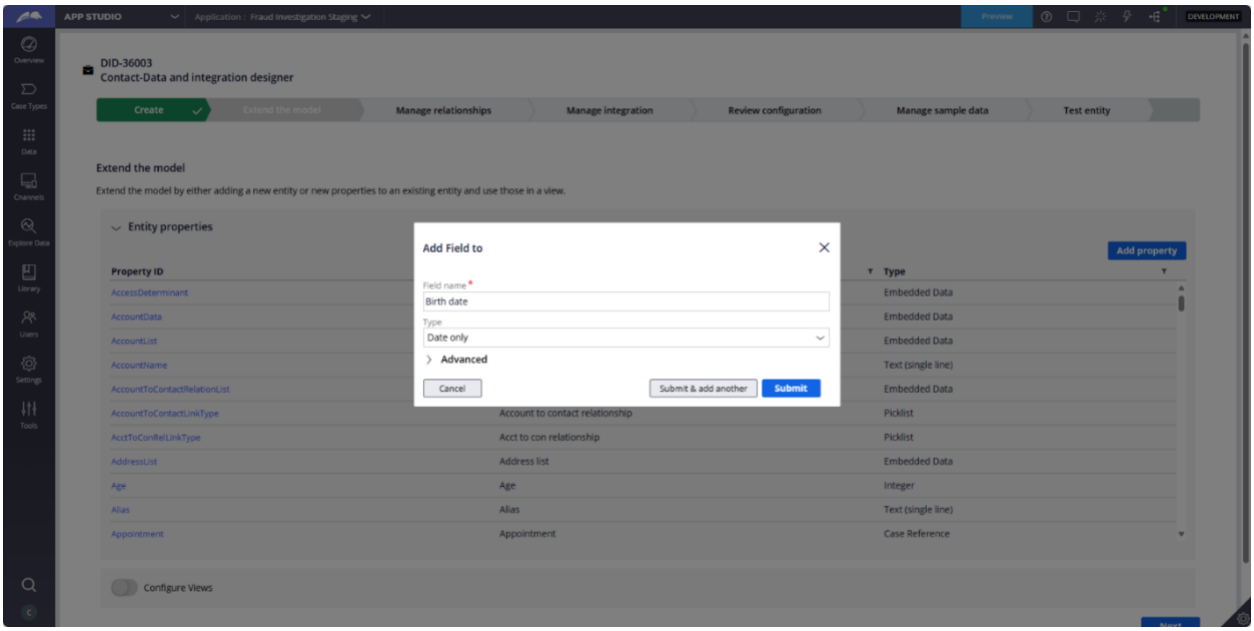
You can quickly add new properties to an entity that can be of any Pega-supported format.

The **Extend the model** portion of the page lists the entity's properties.

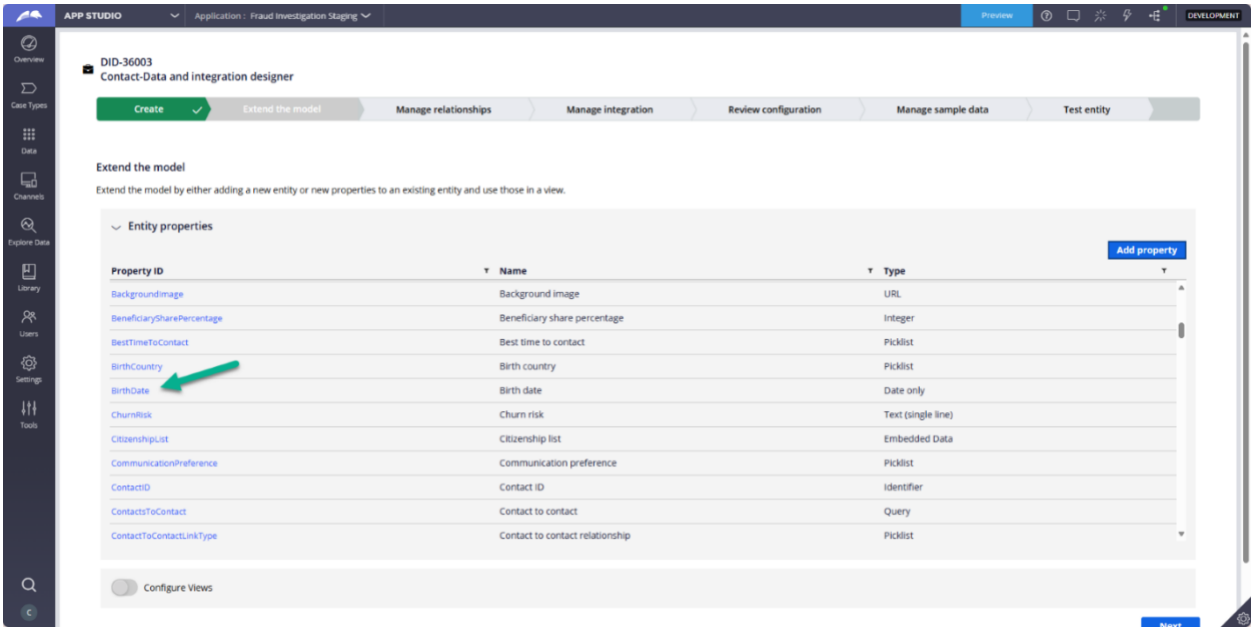


Adding a new property

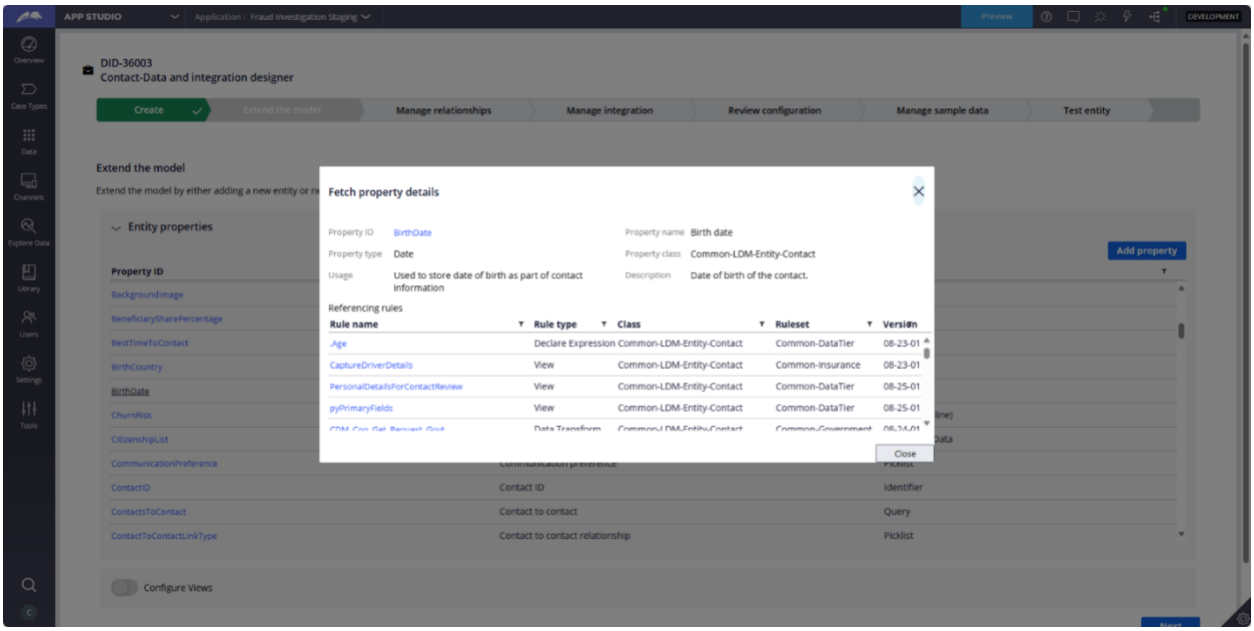
1. Click **Add property**, fill in the required details in the dialog box, and click **Submit**.
   For more information, refer to Fields and Field Types | Pega Academy.



2. Verify that the new property appears in the **Entity properties** table.



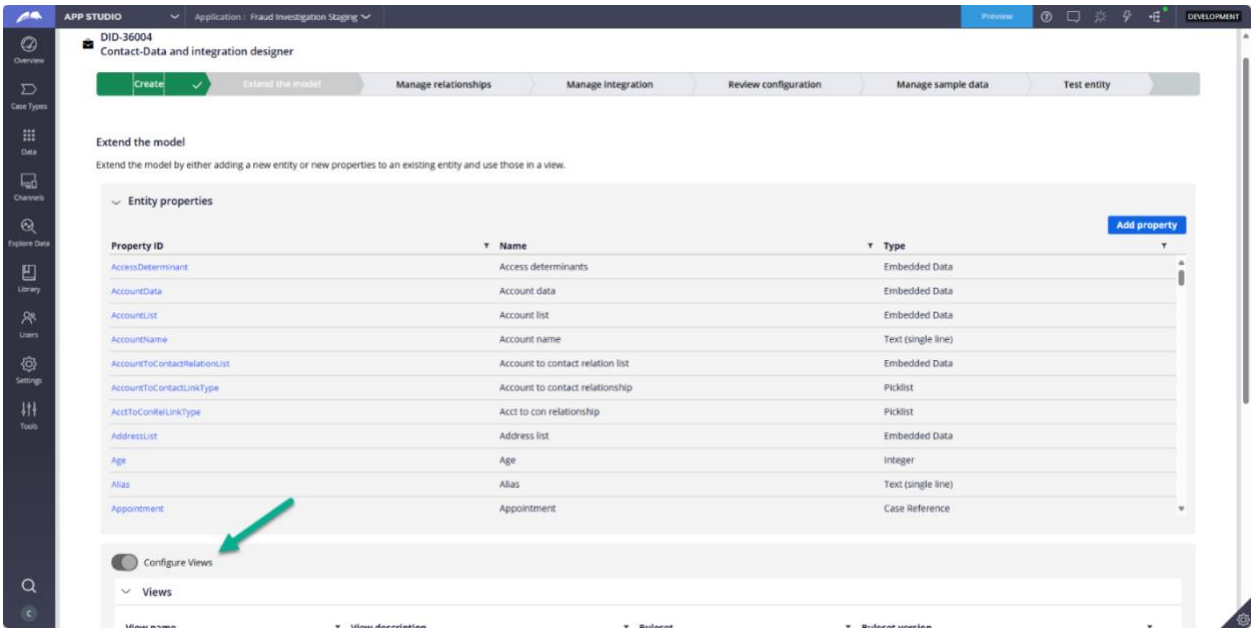
3. Click the Property ID to view the property's details.



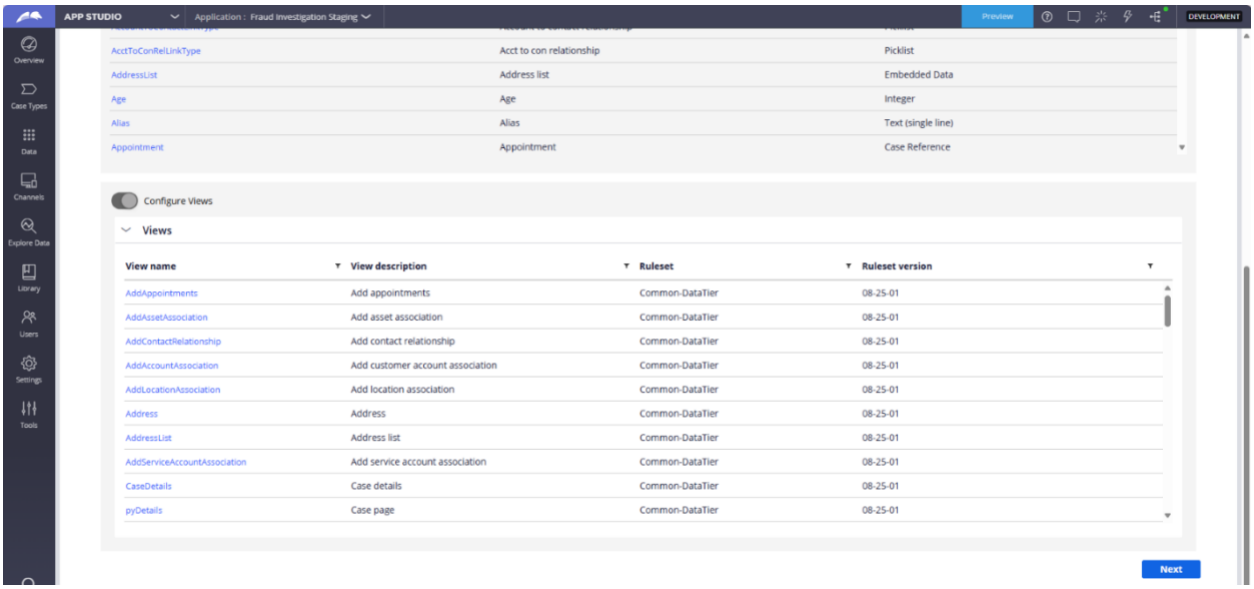
Configuring a view

You can configure views within an entity, deciding which template to use and what kind of content to display in your View.

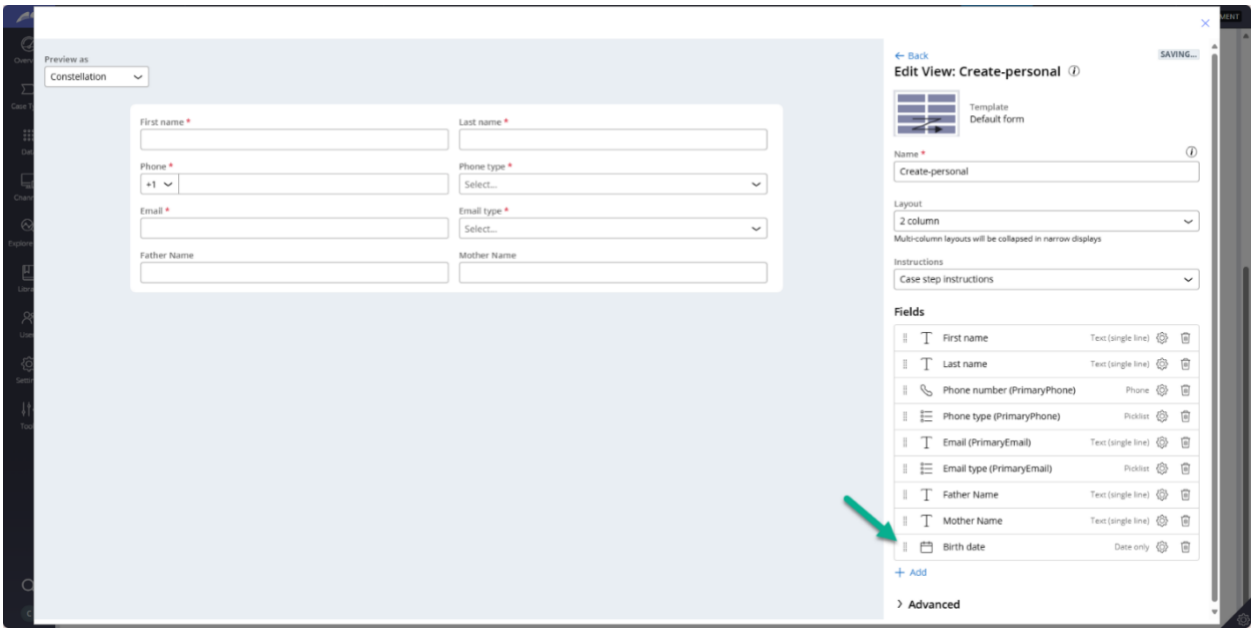
1. To configure views, toggle the Configure Views to On



2. The entity's Views are listed with relevant information.



3. Click the **View name** link to open the view configuration window.
4. You can either Add an existing field or view, or create a new field to configure within the view. For more details, refer to **Configuring Views**.



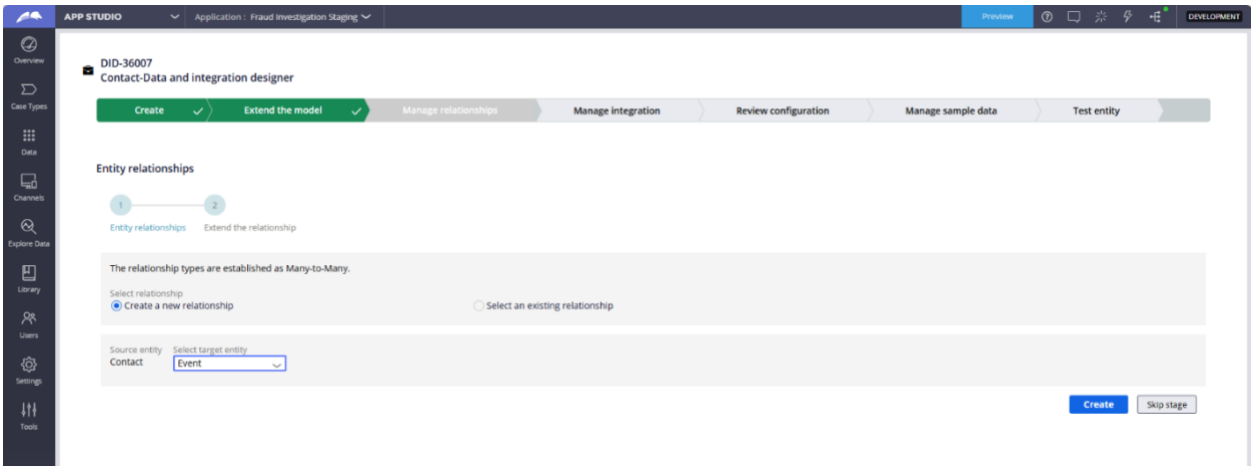
Manage relationships stage

You can manage relationships between entities, either creating a new relationship or managing an existing relationship by adding properties to it.

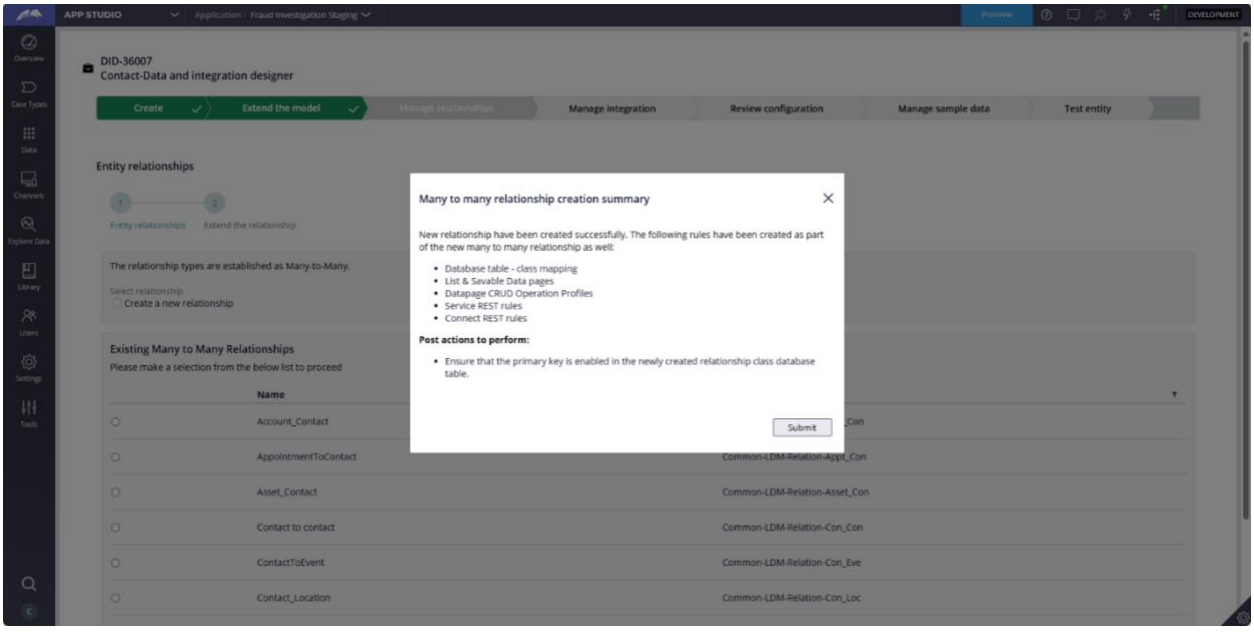
Note: The Designer case currently supports only many-to-many relationships between entities.

Creating a new relationship

1. Select **Create a new relationship**. By default, the source entity will be the selected entity (e.g., Contact).
2. Select the target entity to establish the relationship with.



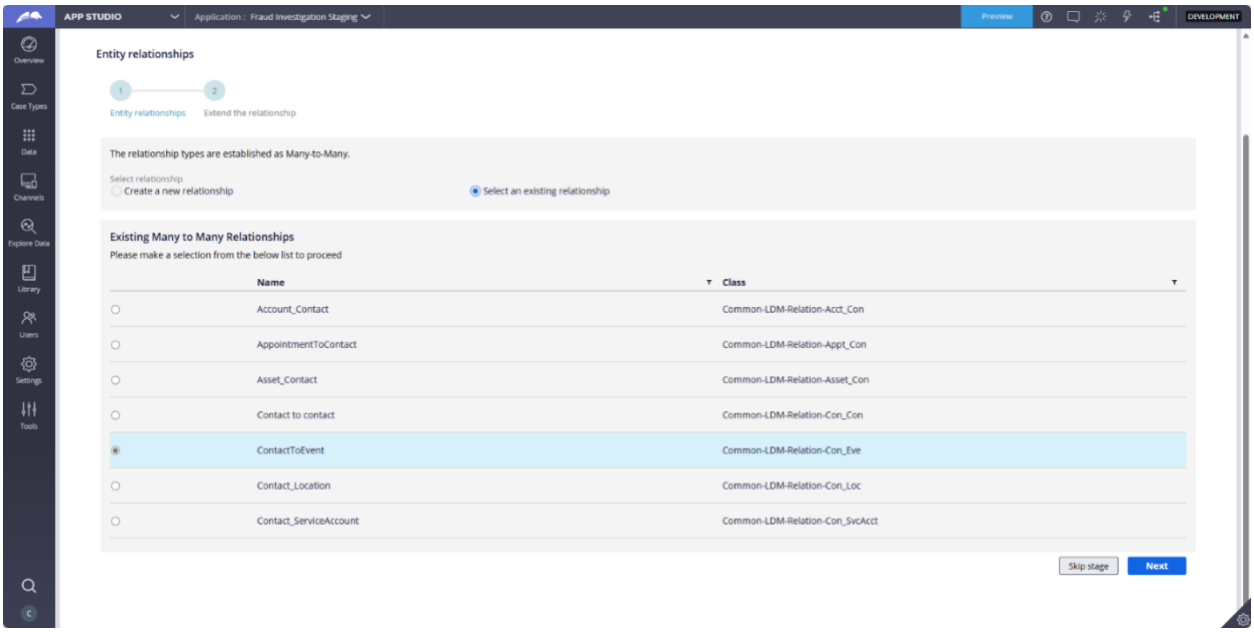
3. Click **Create** to establish the relationship between the source and target entities.
4. Upon successful creation, a relationship creation summary will be displayed with relevant details.



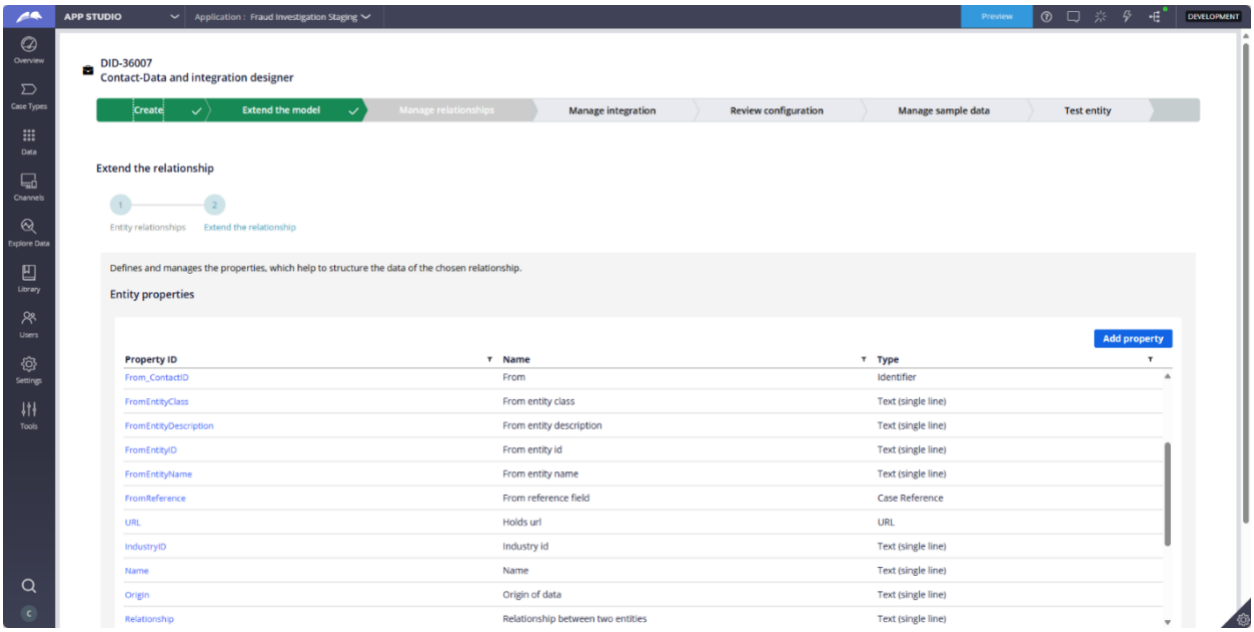
5. Click **Submit** to proceed.

Updating an existing relationship

1. Choose the **Select an existing relationship** option, which lists all relationships for the selected entity.

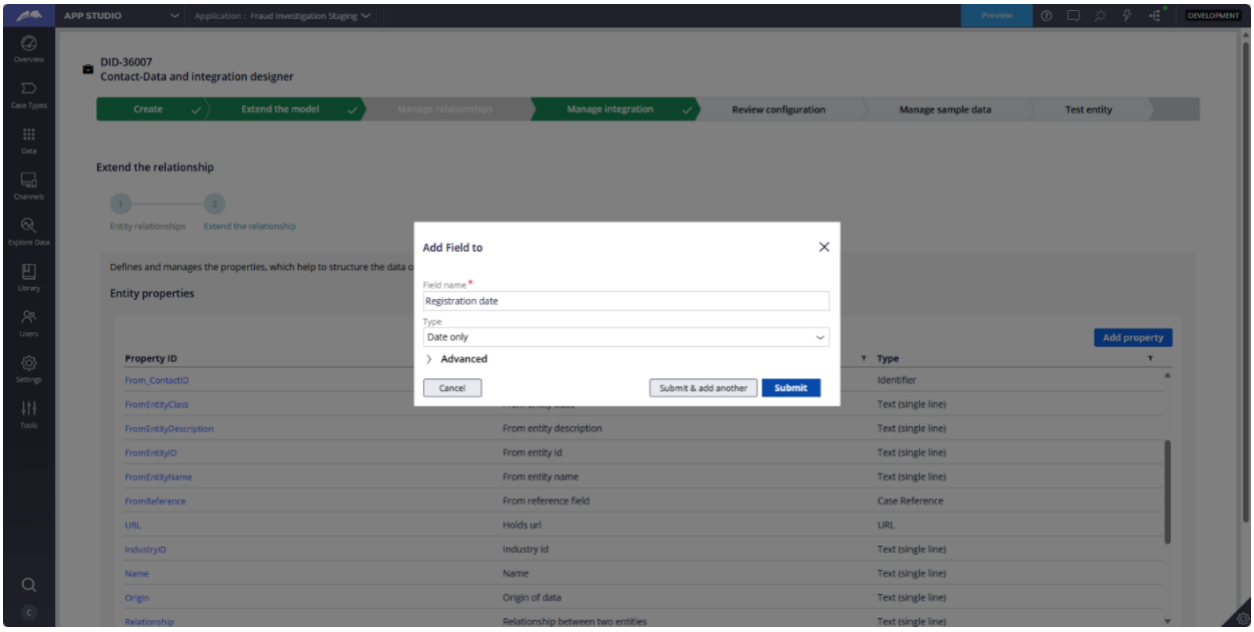


2. Choose the relationship you want to update and click **Next**.
3. On the **Extend the relationship** screen, you can view all properties available for the selected relationship.



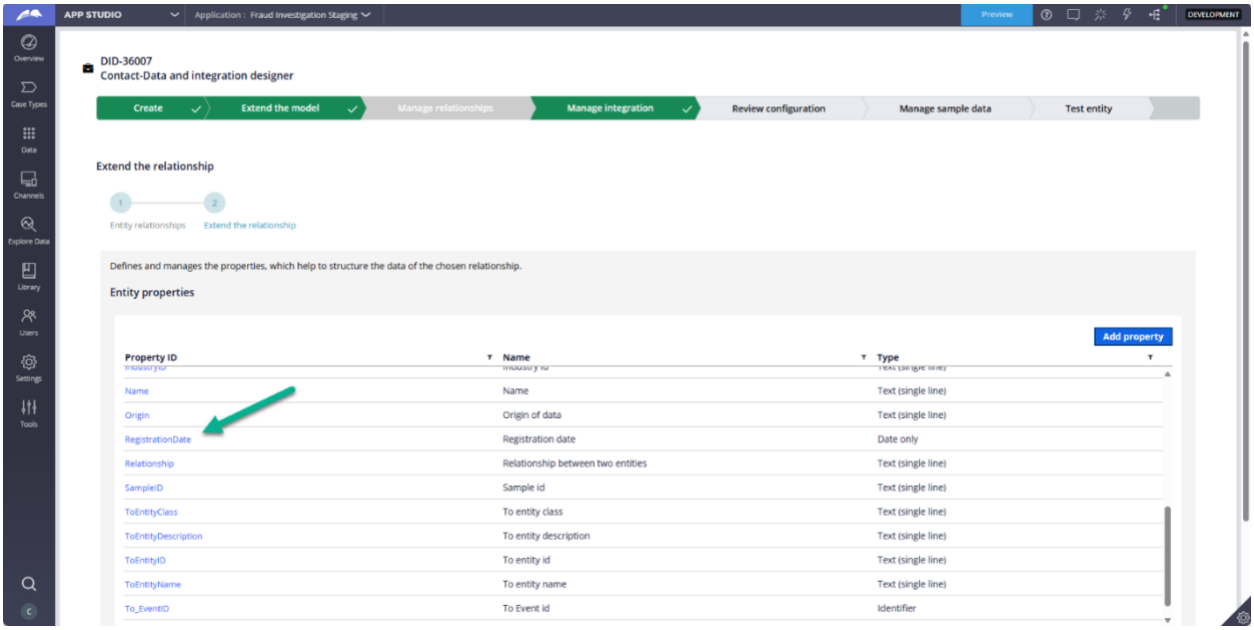
4. To add a new property, click **Add property**

5. Complete the required details in the dialog box

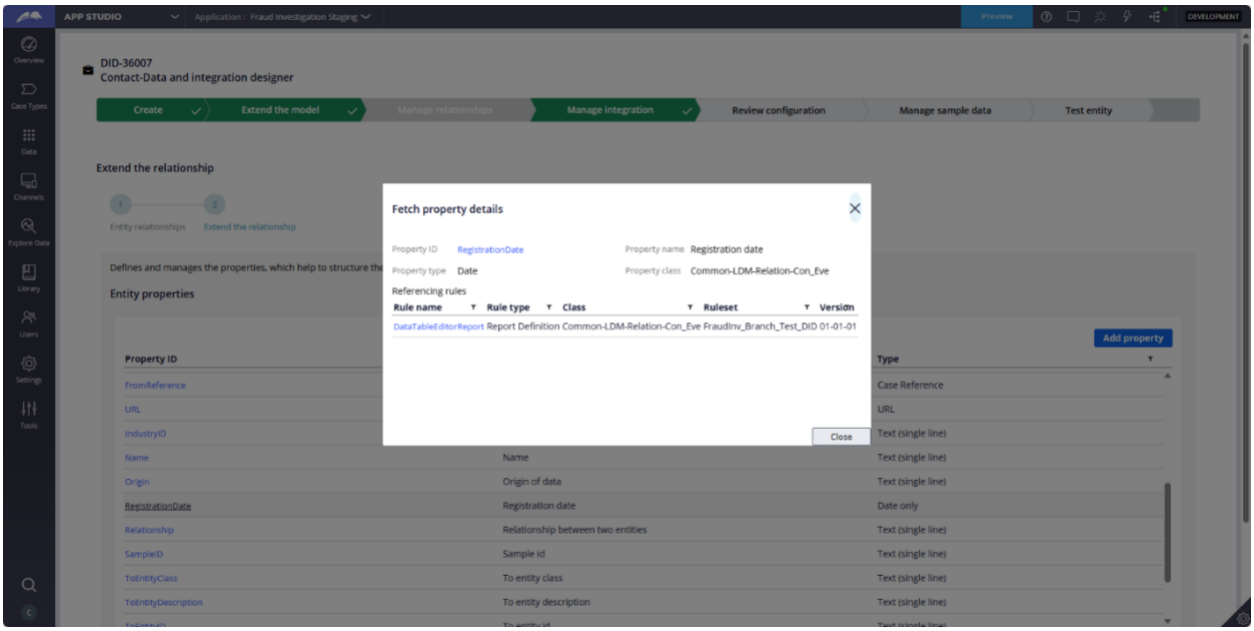


6. Click **Submit**.
   For more information, refer to Fields and Field Types | Pega Academy.

7. Verify that the new property appears in the **Entity properties** table.



8. Click the **Property ID** to view its details.



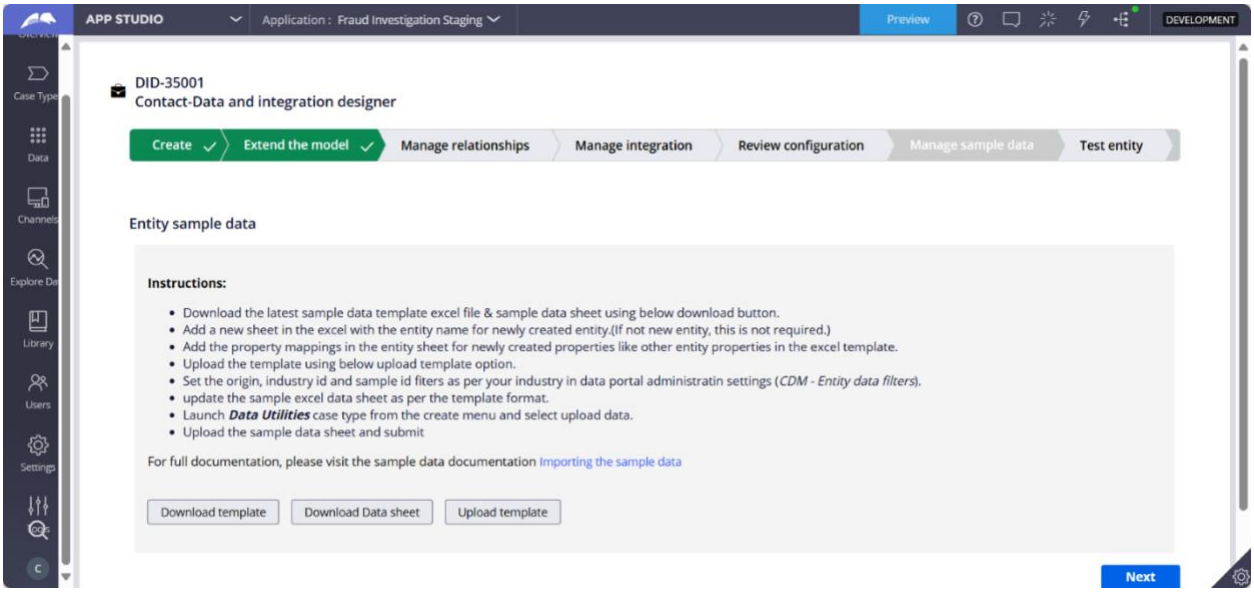
Manage sample data stage

In the 'Manage sample data' stage, you can:

- Update the **sample data template** which tells the sample data upload logic the properties to map from the sample data spreadsheet to the entity's properties
- Download the **sample data spreadsheet** which contains each entity and relationship's sample data.

For more information, see Sample Data Overview.

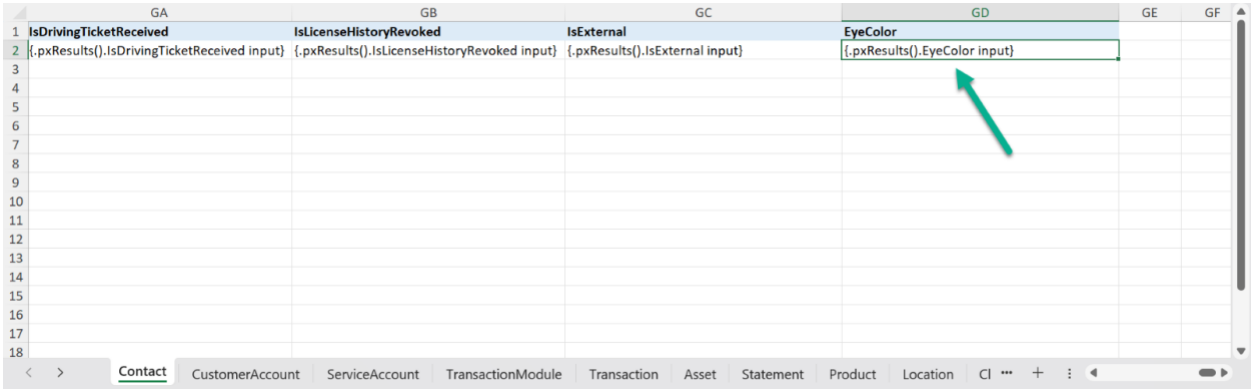
Note: If a test ruleset is not provided in the prerequisites or if the entity class is not "Common-LDM-Entity", the rules required to create sample data should be manually created by the user. Sample data rules are currently only supported if the entity class is "**Common-LDM-Entity**".



Adding new fields to the Sample Data Template

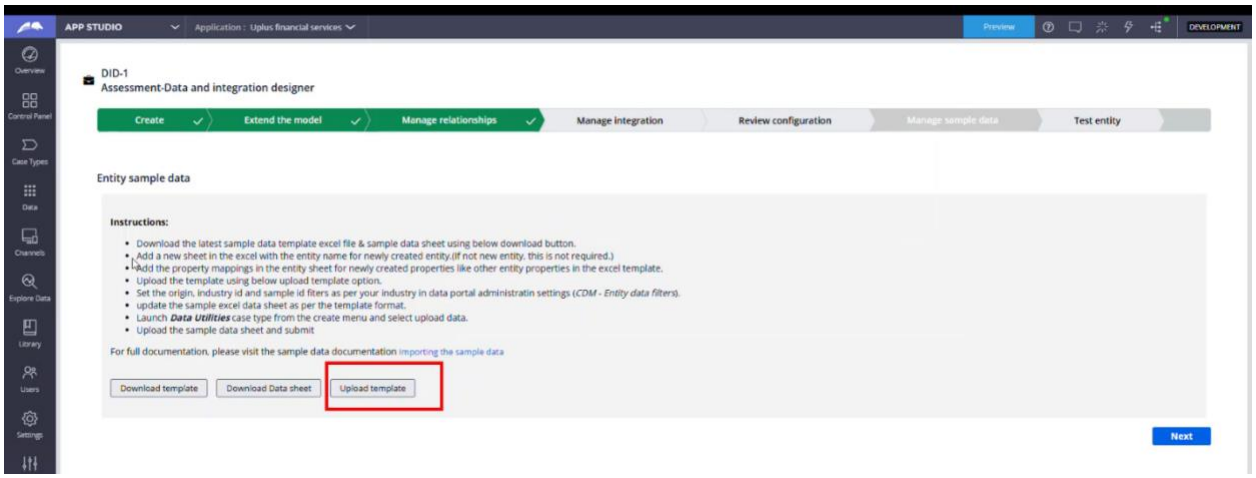
To add new fields (properties) in the sample data template, follow these steps:

1. Download the template using the *Download Template* option.
2. Update the sheet by adding the desired property field.
   For example, add an Eye Color field in the Contact sheet:
   - ➢ Navigate to the *Contact* sheet.
   - ➢ Add a new column at the end of the sheet.
   - ➢ Set the label as EyeColor and provide the value in the following format: {.pxResults().EyeColor input}



3. Save the updated sheet.

4. Upload the template to the branch by clicking **Upload template** in the Designer case. This ensures that all the new properties of the entity and relationship are taken care of.



Adding new fields to the Sample Data Spreadsheet

1. Click **Download Data sheet** which downloads an existing sample data spreadsheet.
2. Add the same new fields that were added in the template spreadsheet along with the sample data for these fields.
3. Upload the sample data spreadsheet using Common's "Data utilities" case in the Data Portal.
   For more information, see Importing the Pega Common Data Model sample data

Note: To add sample data for new entity or new relationship, add a new sheet and follow the same steps mentioned above to add the fields.

For more information on adding entities and relationships, see:

- Adding an entity to the sample data template spreadsheet
- Adding an entity to the sample data spreadsheet

Manage integrations stage

The Manage Integration stage provides a guided approach for configuring external integrations for existing entities within a selected profile. This stage empowers users to seamlessly connect their application to external systems by capturing connection details and defining mappings necessary for rule generation.

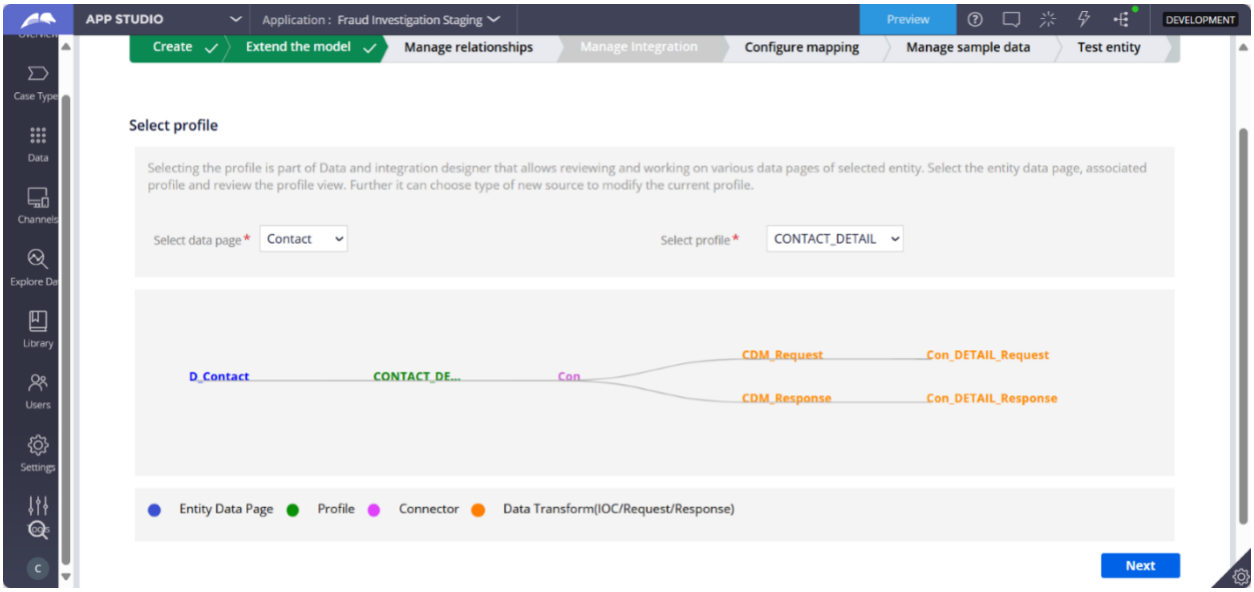
Through this process, the Designer case automates creating essential integration artifacts, including classes, properties, data transforms, a connector and optionally, an authentication profile, and an OAuth 2.0 provider, and updates to necessary decision tables and data page sources.

By streamlining these configurations, the Manage Integration stage reduces manual effort, ensures consistency, and accelerates the integration setup - enhancing overall development efficiency.

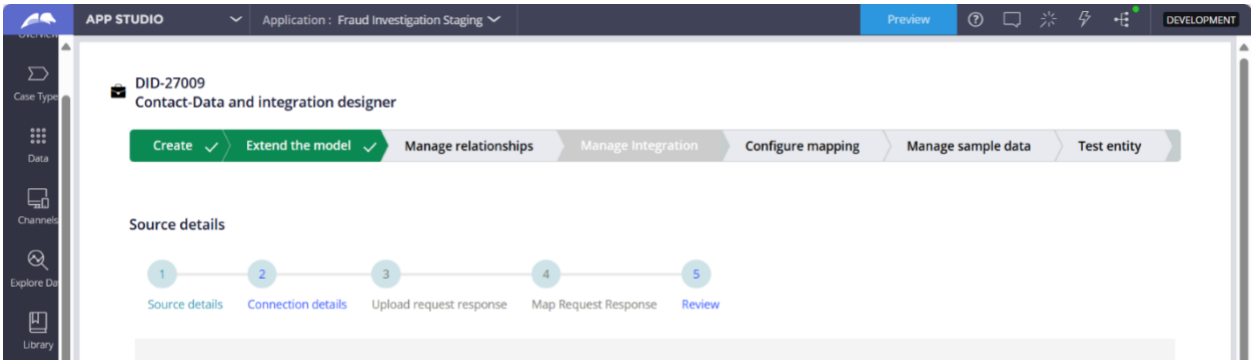
Configuring an external integration

1. Select the **Data Page** and one of its associated **Profiles** you want to configure the external integration.
2. click **Next**.

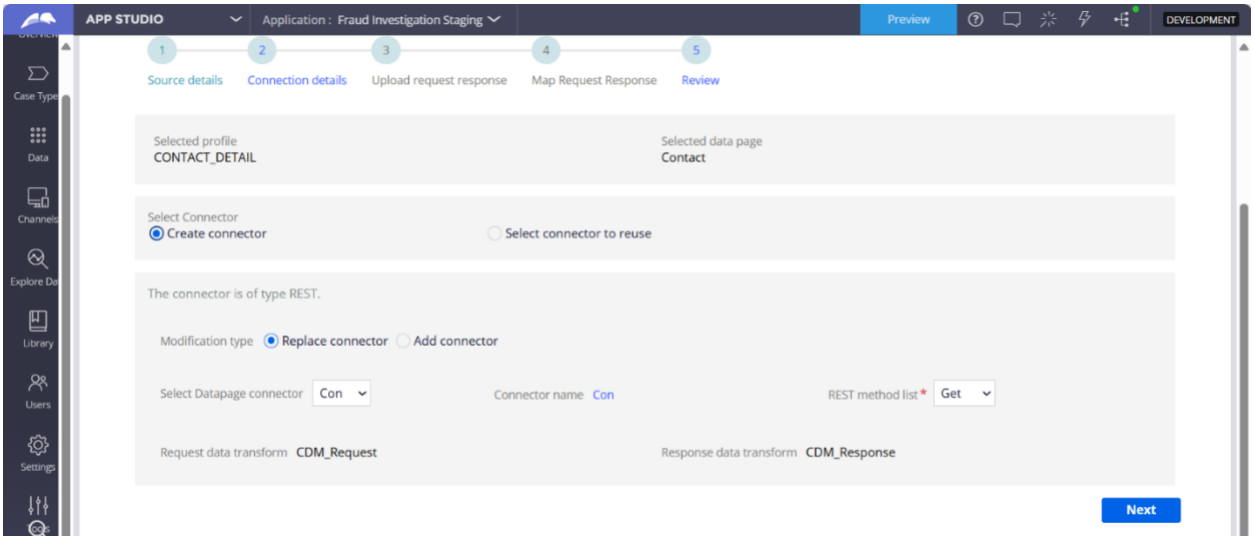
   For the selected profile, a **Profile Viewer** is displayed, visually illustrating the flow of calling rules. These elements are clickable and open the respective rule forms for easy access.



3. Complete the 5 Main Steps of Integration Configuration



a. **Source details** – This step involves selecting and configuring the source of the integration.

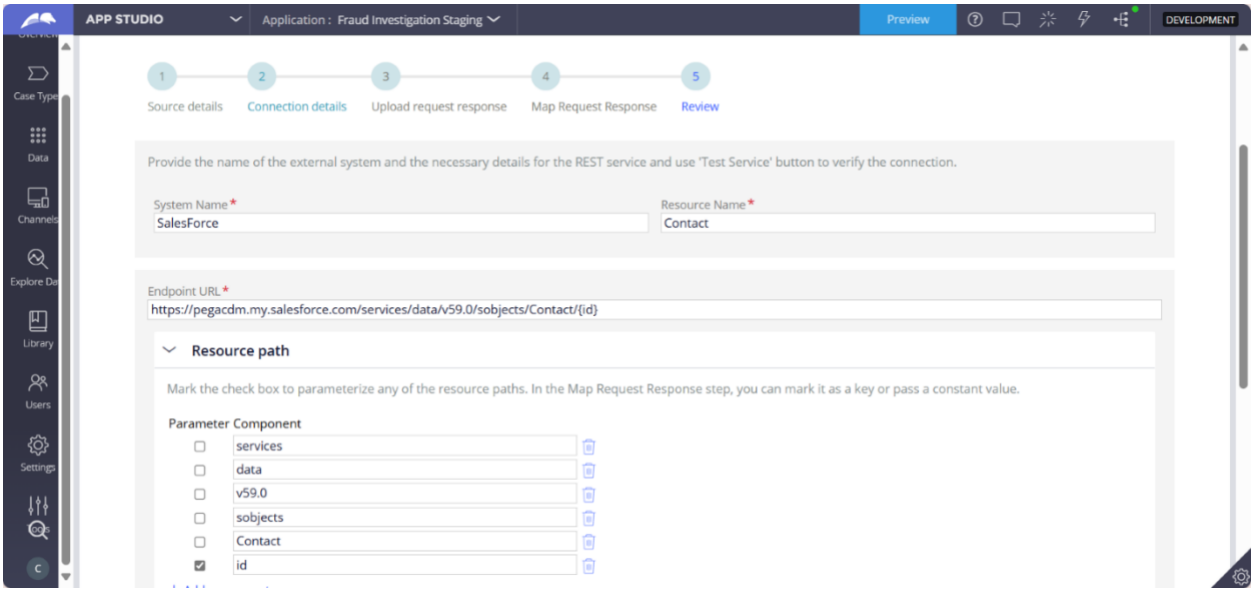


- **Select Connector**
  - *Create Connector*: Allows you to create a new connector by providing connection details.
  - *Select Connector to Reuse*: Lets you reuse an existing connector, automatically populating connection details from the selected connector.
- **Modification Type**
  - *Replace Connector*: Replaces the existing connector with the newly generated one.

- *Add Connector*: Adds a new connector as an additional source (available only for aggregation-supported sources; not available for data save profiles).

- **Select Data Page Connector**
  - Only available if *Replace Connector* is selected. Allows replacement of a connector in the chosen profile with the new one.
- **REST Method List**
  - Select the REST method (GET, POST, PUT, etc.) for the new source.

b. **Connection details**



- **System Name** - Enter the name of the source system. This field represents the name of the system that hosts the external REST service.
  - For example, if the external REST service is a weather service such as Google Weather API, you can enter Google Weather in this field.
- **Resource Name** - Enter the name of the resource that you want to access by using the external REST service.
  - For example, if the REST service is an eCommerce API such as Best Buy and you want to use this API to access the product categories, you can enter Product categories in this field.
- **Endpoint URL** - Enter the endpoint URL of the external REST service.

- Enter the endpoint URL from the service REST rule that you already have set up or that you created in Creating a Service REST rule . The system analyzes this URL and suggests the elements of the URL that represent resource path parameters and query string parameters.

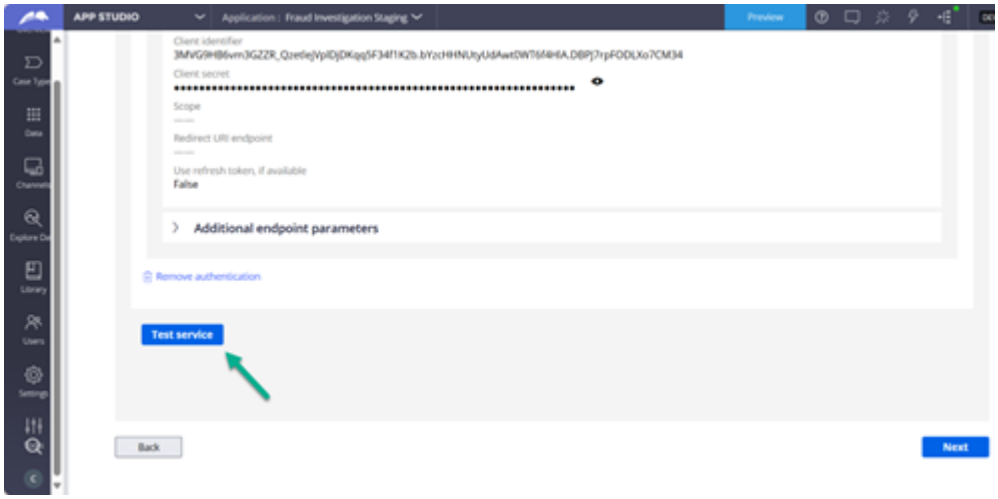
- **Resource path** - To update the suggested resource path names, in the Resource path section, click Add component, and then select the Parameter check box next to each resource path name that is a parameter.
    - The resource parameters identified by the system are static and not going to change. If there is a component that might change, select the check box next to it to consider it a parameter. The system updates the endpoint URL accordingly. The parameter is displayed in curly braces in the endpoint URL. If you mark a resource path name as a parameter, the system generates a property as part of the request data model and substitutes the property's value for the value of the parameter at run time.
- **Query string parameters** – Update the suggested query string parameters and add more by clicking. The system considers query string parameters as part of the request and creates properties for them.
- **Headers** - Add custom headers for the external REST service by clicking **Add header**.
    - The system adds each request header to the REST connector rule that is generated for the method that you selected on the **Source details** (step a) page. The value for each header is the value that you specify in the first sample that you collect on the **Data model** page of the wizard.

NOTE - The parameterized resource path parameters/ Query string parameters/ headers mappings can be managed in Map Request Response(step d) within this flow.

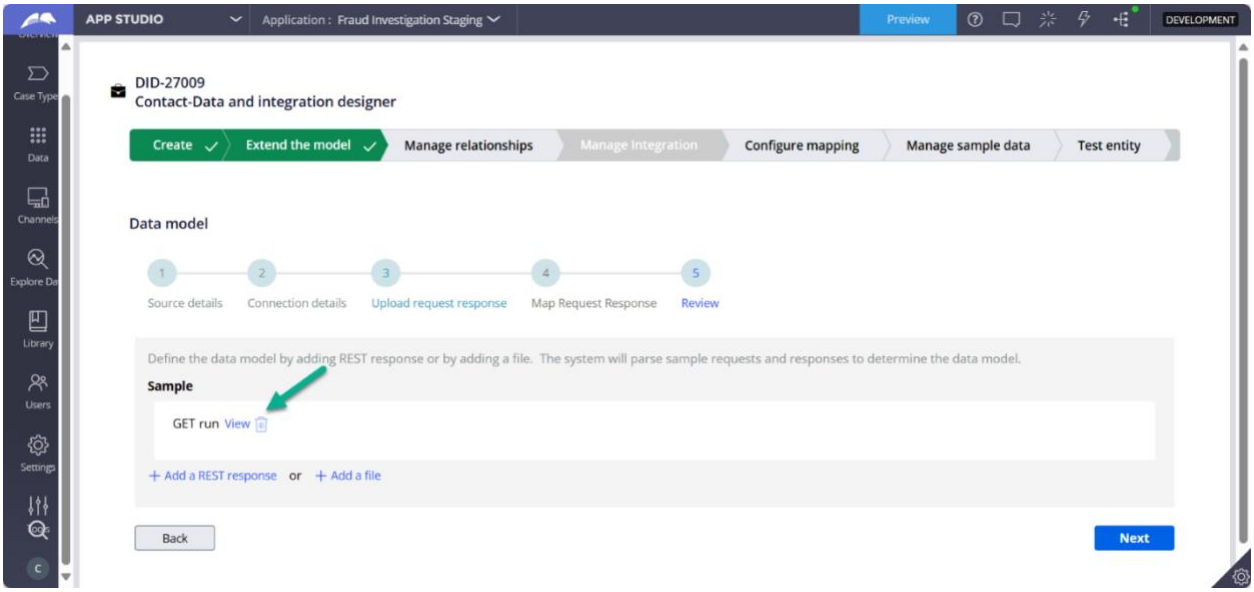
- **Authentication** - Specify or create an authentication profile by selecting one of the following options:
    - Select **Use Existing** to choose an existing authentication profile and enter the profile name.
    - Select **Define New** to create a new authentication profile and select an authentication scheme:

- Basic authentication scheme: Enter the username and specify a password by clicking **Set Password**. Enter the realm and host name. Enable preemptive authentication by selecting the check box if the external service requires preemptive authentication.
- HTLM authentication scheme: Enter the username and specify a password by clicking **Set Password**. Enter the domain and host name.
- OAuth 2.0 authentication scheme: Create an OAuth 2.0 authentication profile by following the steps that you use to complete the **OAuth 2.0** tab of an Authentication Profile data instance. Create an OAuth 2.0 provider if needed.
- The **Revoke access tokens** button is not visible on this page. Instead, you can use the **Disconnect** button as explained below.
- If you select **Authorization code** in the **Grant type** list, click **Connect** to generate an access token. This action authenticates and authorizes your connection with the OAuth 2.0 service provider and allows you to access protected content. Click **Disconnect** to revoke the access token.

- **Test service** - After entering details, test the integration by invoking the service.

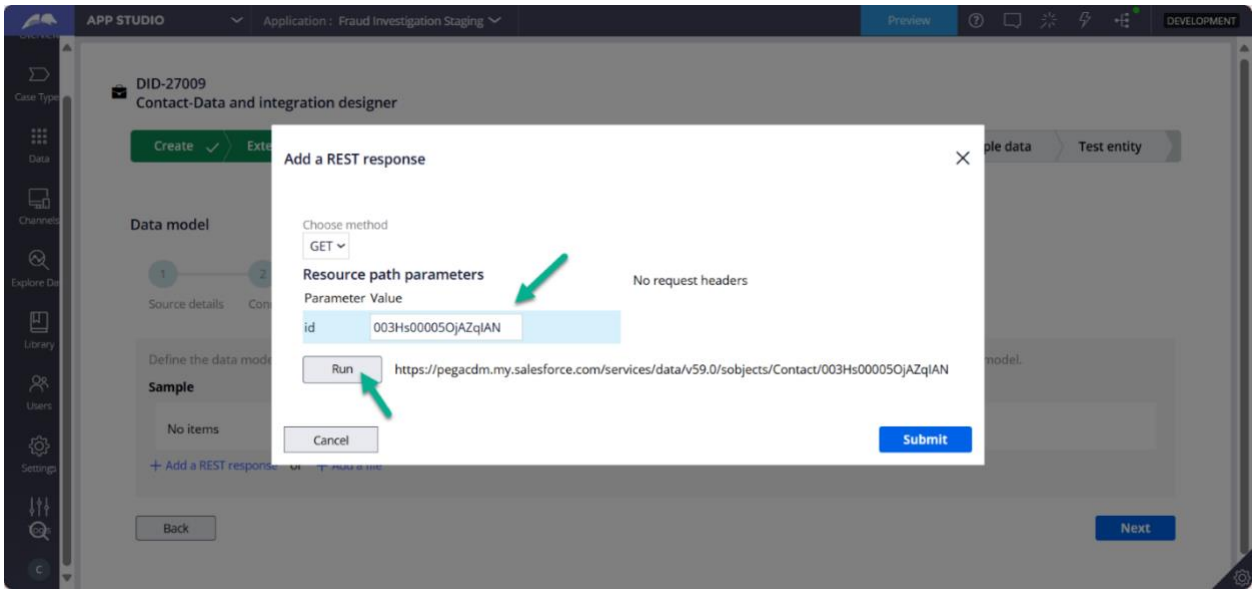


c. **Upload request response** – In this step, you can define the external data model for your REST integration by adding a REST response or uploading a file. The system parses sample requests and responses to determine the data model and populate in the next step to Map Request Response.

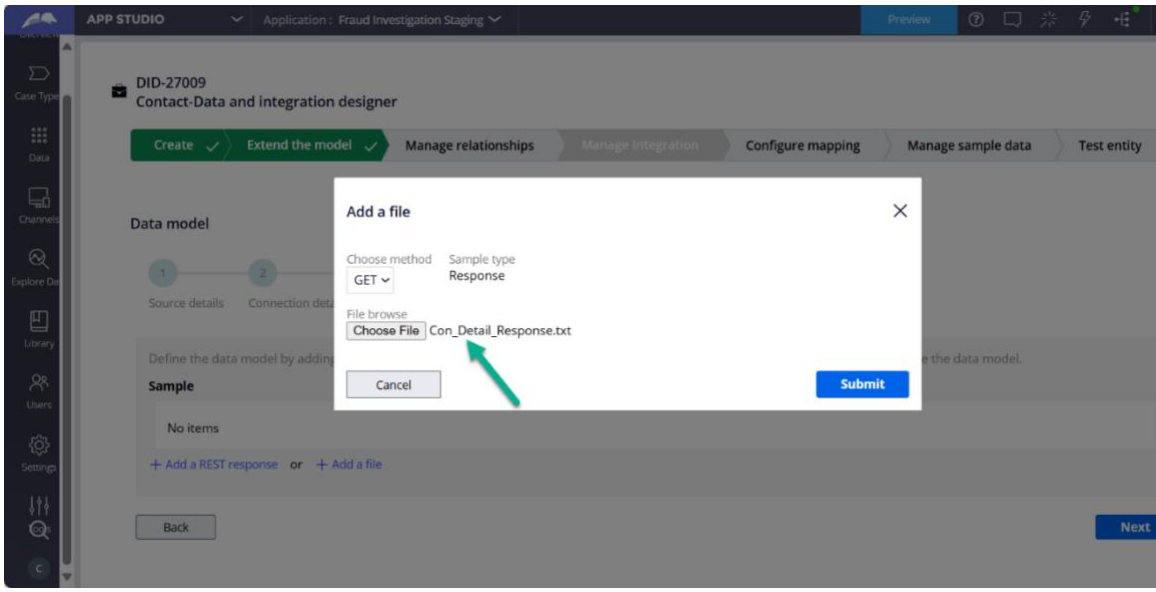


- **Add a REST response**
  - Click **Add a REST response**.
  - Select a method from the **Choose method** list.
  - Enter the values of individual elements under **Resource path parameters**, **Request headers**, and **Query string parameters**. The system updates the endpoint URL with the values of the resource path parameters.
  - For POST and PUT methods, select the content type for the request and configure the body of the request.
  - The parameter and header values are applicable only for the current instance of test execution for the method that you selected in step a.
  - If you are using OAuth 2.0 with the authorization grant type to authenticate your REST data source and if you did not connect to the OAuth 2.0 service provider on the **Connection** page of the wizard, click **Connect** to generate an access token. This action authenticates and authorizes your connection with the provider and allows you to access protected content.
  - Once you are connected to the OAuth 2.0 service provider, click **Run** to view the response. The header information is displayed on the **Headers** tab.

- Click on submit to save the response.



- **Add a file**
  - Click **Add a file**.
  - Select a method from the **Choose method** list.
  - For POST and PUT methods, select the type of data that you want to upload.
  - Click **Choose File** to browse for a file of sample request or response data and click **Open** to upload the file.
  - Click submit to save the sample data that you uploaded.

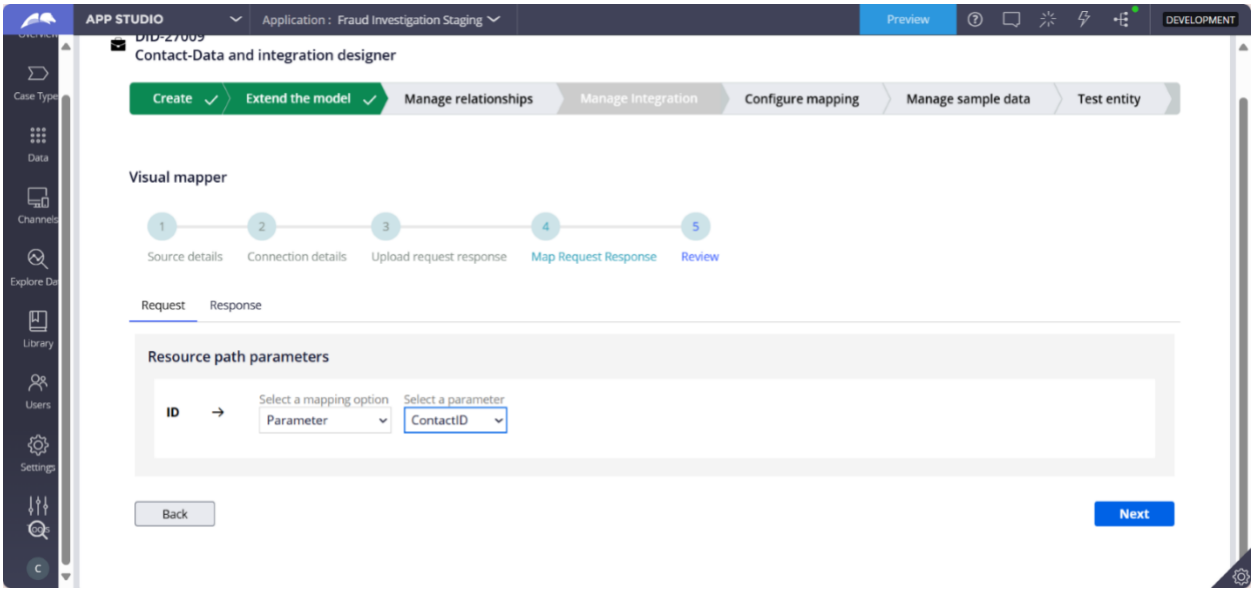


d. **Map Request Response** - The Map Request/Response step allows users to map the request and response structures of the external source with the Common Data Model using a simplified and intuitive mapping UI.

- **Request**

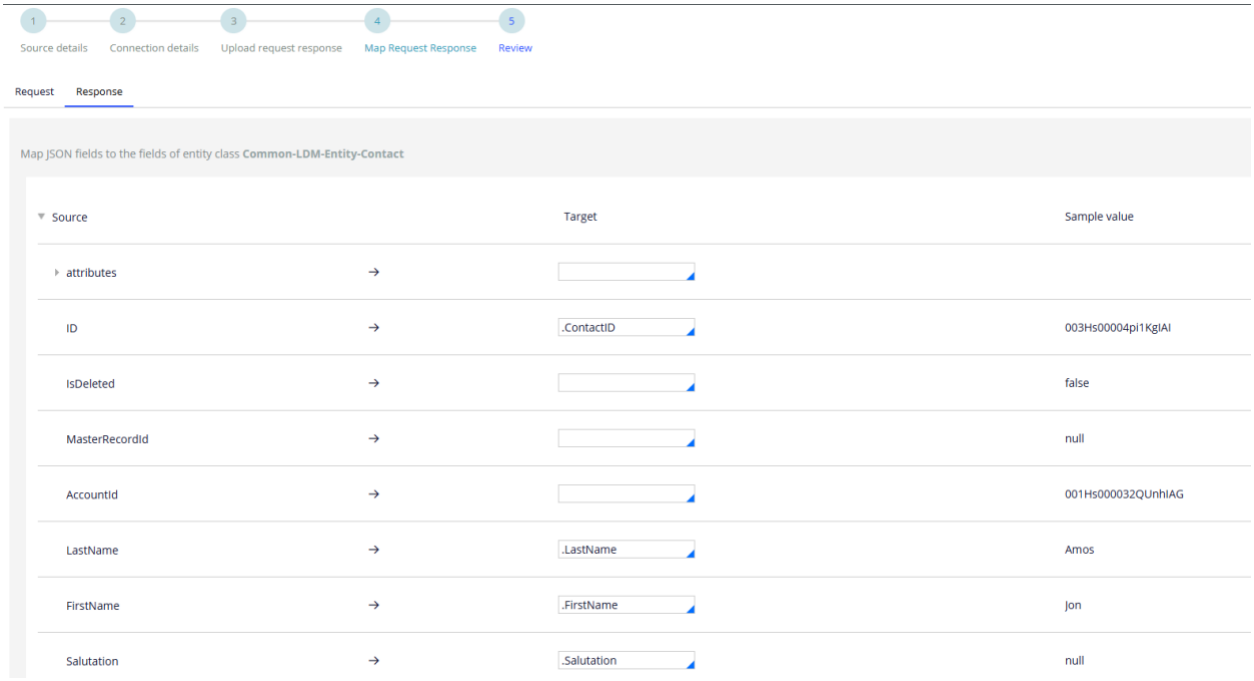
  - Within the Request tab, users can configure request parameters—such as resource path, query parameters, and headers—by setting them to either a constant value or a data page parameter.

  - If a request body is present, mappings can be established between the external request structure and the Common Data Model properties. The mapping UI presents a table where each row displays a Source, representing the external data elements, and a Target, representing the corresponding property in the Common Data Model. Additionally, sample data is shown for each row to aid in accurate mapping and validation.

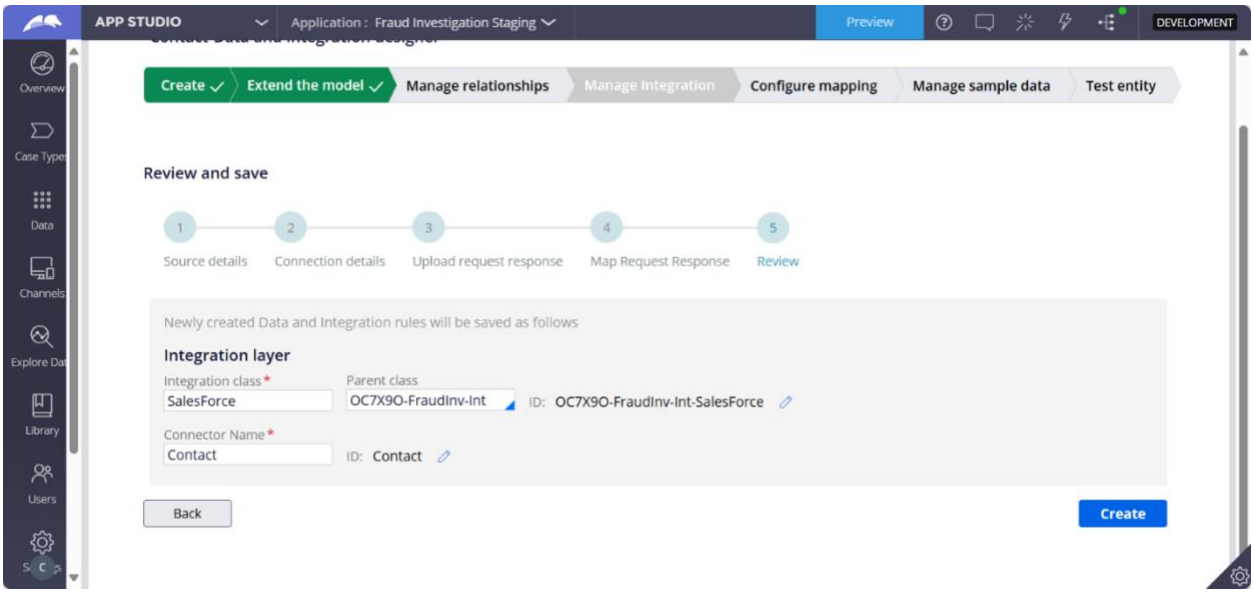


- **Response**

  - If a response body is present, mappings can be established between the external request structure and the Common Data Model properties. The mapping UI presents a table where each row displays a Source, representing the external data elements, and a Target, representing the corresponding property in the Common Data Model. Additionally, sample data is shown for each row to aid in accurate mapping and validation.



e. **Review** - On the Review page, you can review and optionally update the information for the integration layer of your REST integration in the **Integration layer** section.

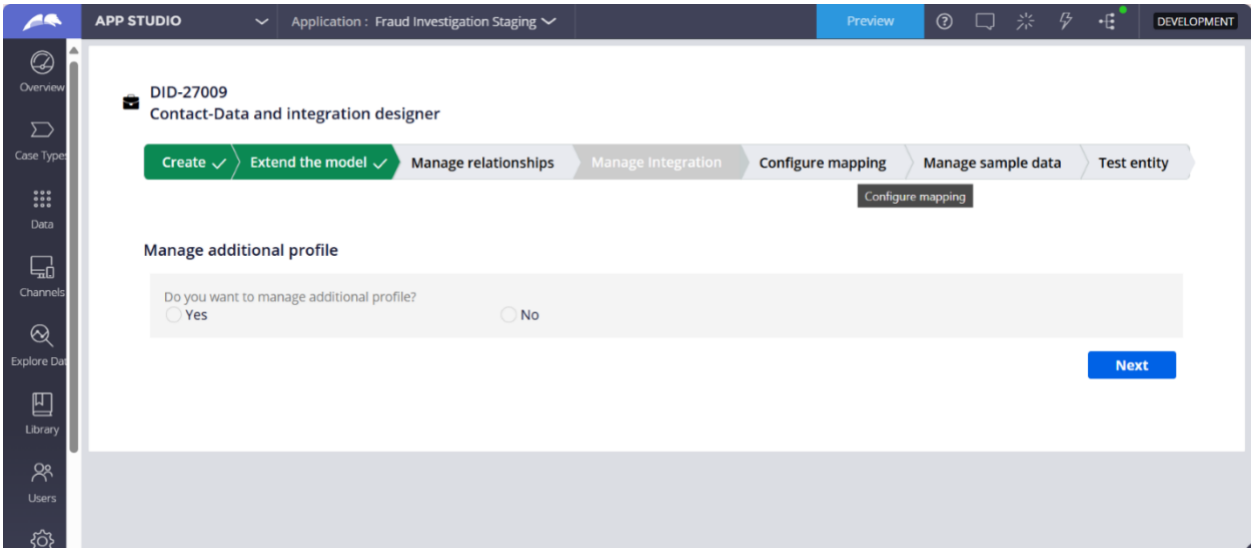


- In the **Integration class** field, update the name for the base class of your REST integration.
- In the **Parent class** field, use the autocomplete control to update the parent class for the integration. Choosing the correct integration layer is a

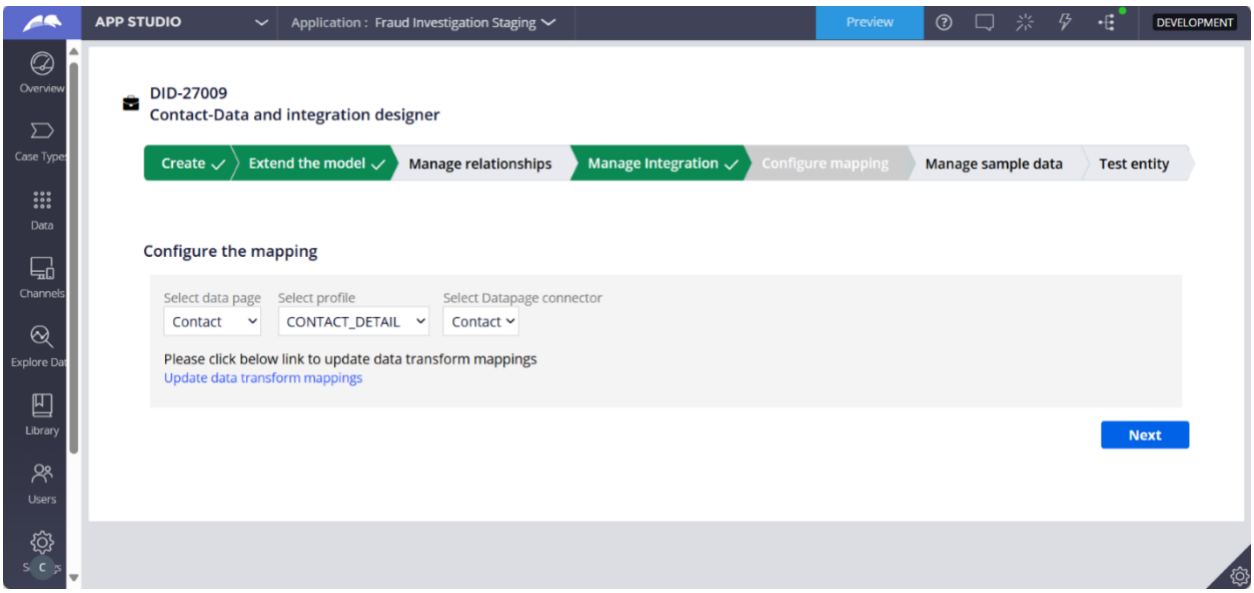
critical decision that affects whether a connector can be reused across different applications.

- If a connector is specific to a single application, it should be created in the **Application Integration layer**. This ensures it remains tightly coupled with the application it supports.
- On the other hand, if the connector is intended for use across multiple systems or business units, it should be placed in the **Organization Integration layer** or
- Another option is to use a **modular and reusable integration layer.** For example, since we are using a Common Data Model, connectors can be placed in the common-int layer, allowing them to be reused across multiple applications.
- In the **Connector Name** field, update the name of the REST connector. The default name is the name that you entered on the **Resource methods** page of the wizard. Click the **Edit** icon next to the **ID** field for the REST connector to update the identifier for the connector.
- Click **Create**.

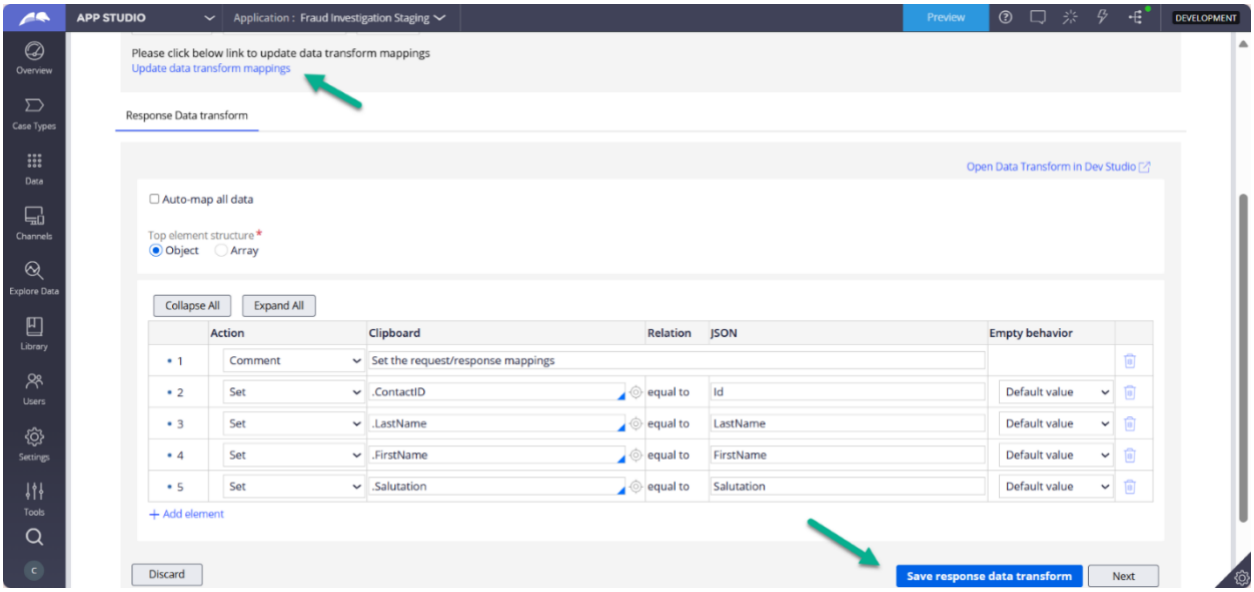
7. **Generate Integration Artifacts**
   Click **Create** to generate all integration rules and artifacts.
8. **Optional: Configure Integration for Another Profile**
   Click on Yes if needed, which returns to Step 5 to configure another profile using the same flow.



9. **Switch to Configure Mapping Stage**
   Click the **chevron icon** to move to the **Configure Mapping** stage and make any additional or update mappings.

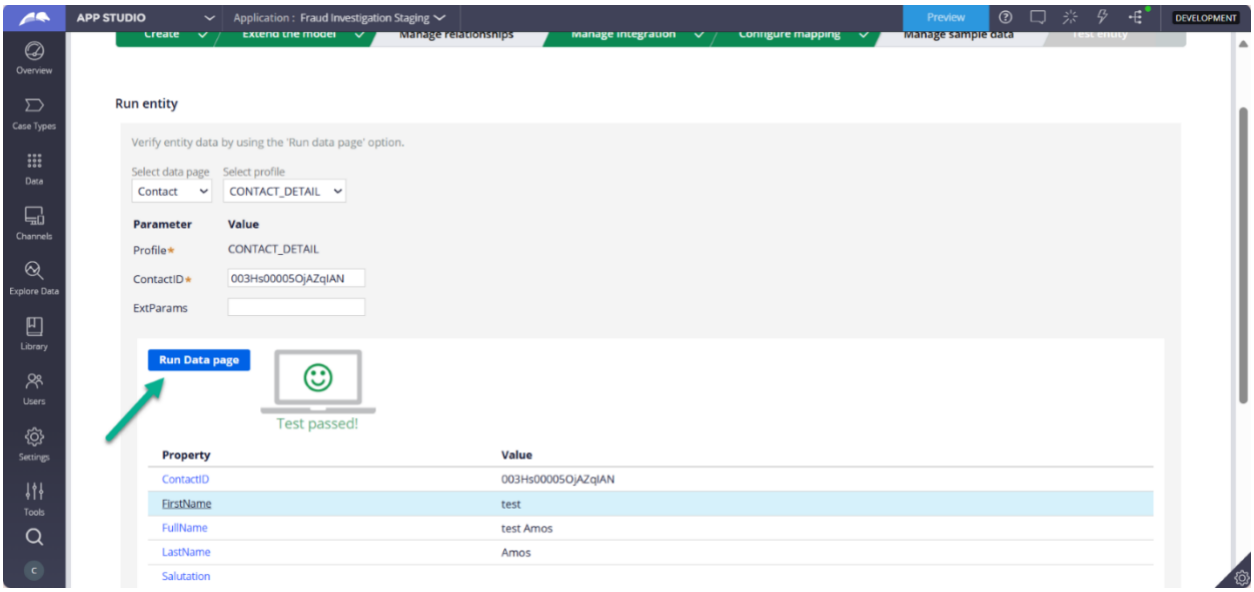


Click on Update data transform mappings to view the current request/response mappings



10. **Test the Integration**

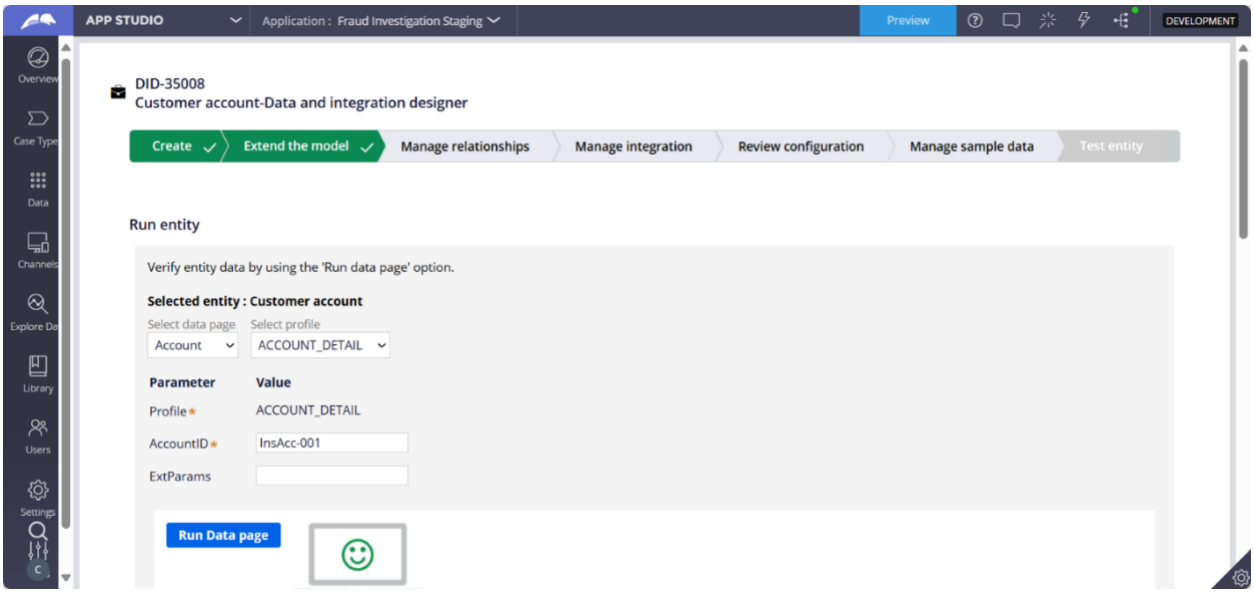
    Switch to the **Test Entity** stage and run the data page to validate the integration results.



Test entity stage

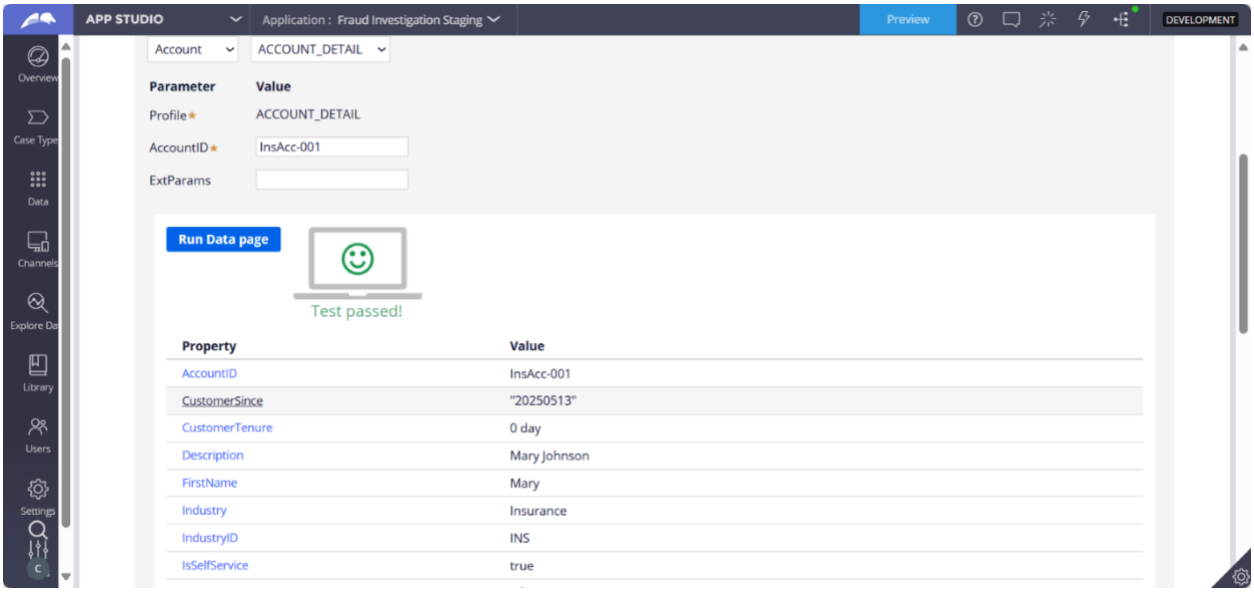
In the Test Entity stage of the Designer case, users can test an entity for any selected profile and run a data page to verify the results directly within the designer, without needing to switch to Dev Studio.

For example, after adding a new property to an existing entity or setting up an external data source integration, users can immediately verify the entity's functionality.

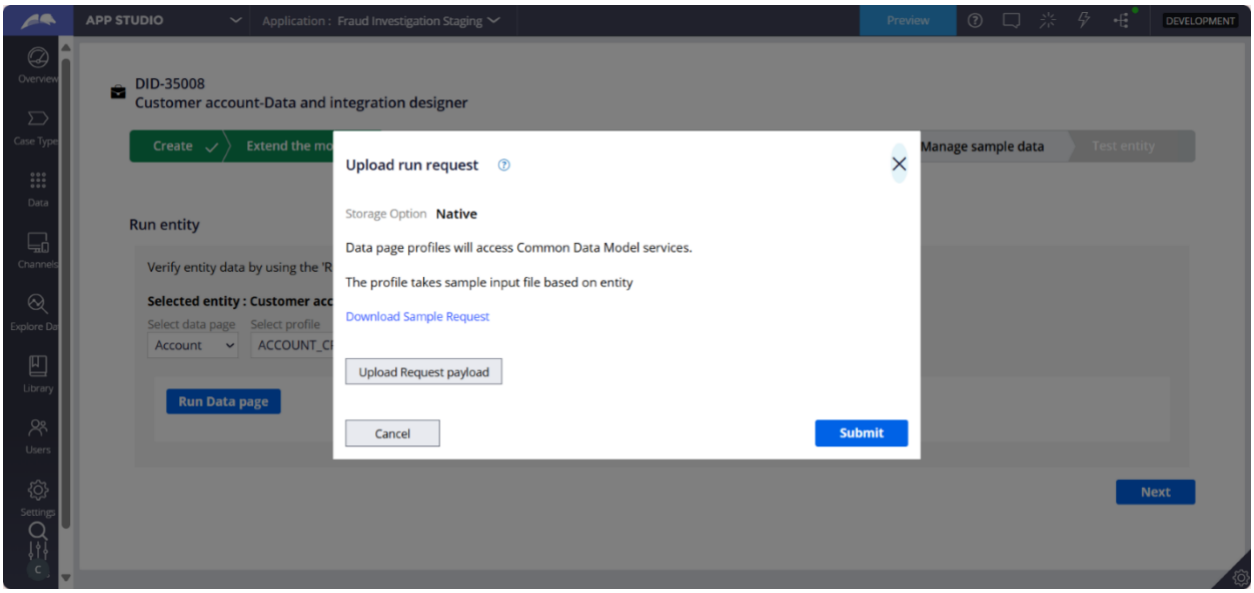


Testing an Entity

1. Select a Data Page and choose one of its Profiles to test.
2. For a **GET** method profile, provide the input parameters on the data page to populate the data.
3. Click on the "Run Data Page" button to verify the results.



4. For **POST/PUT/PATCH** method profiles, click on the "Run Data Page" button, which will open a pop-up to upload the request. For Native sources (PegaSoR), you can download a sample request file for reference by clicking the "Download Sample Request" option.

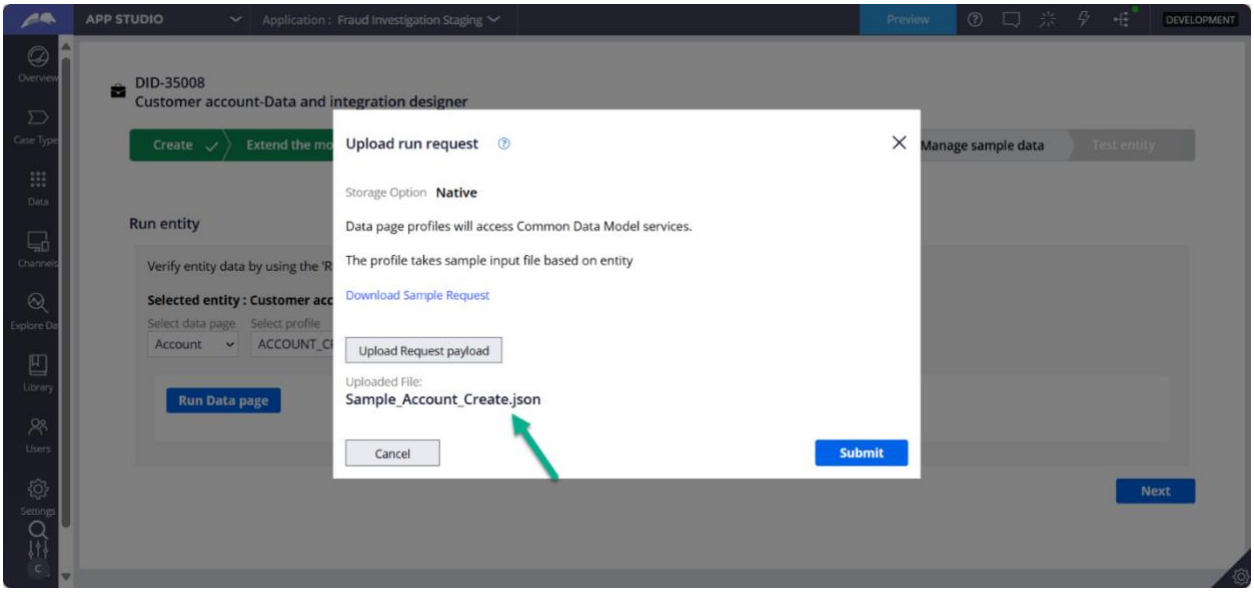


5. Modify the downloaded sample request file with the appropriate request body or create a new one

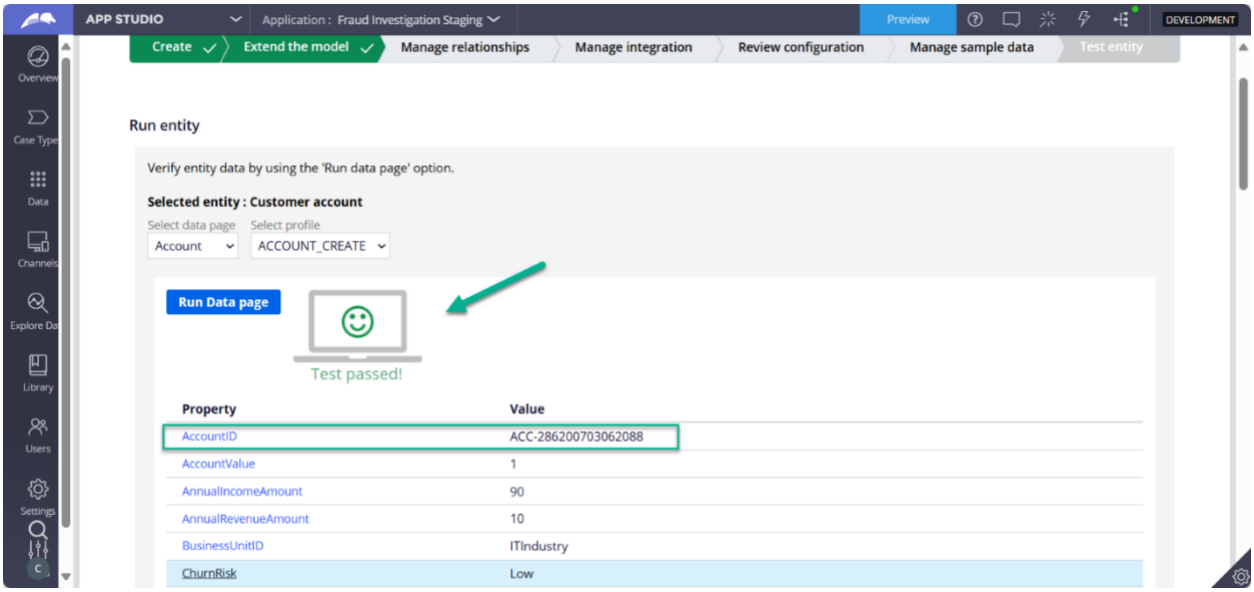
File Edit View


```
{
      "Type": "Consumer",
      "CountryID": "USA",
      "AnnualIncomeAmount":90,
      "IncorporationState":"New",
      "CreditScoreLevel":"High",
      "IncorporationCountry":"IND",
      "RevenueChangeAmount":30,
      "LegalName": "Test Acc Legal",
      "LegalFormType": "Corporation",
      "ParentAccount":{ "AccountID": ""},
      "AnnualRevenueAmount":10,
      "IncorporationDate":"20200103",
      "BusinessUnitID": "ITIndustry",
      "Description": "Tata Consultancy",
      "RelationshipType": "BusinessPartner",
      "Ticker": "DOW",
      "Website": "www.tCDM.com",
      "Name": "Tata Consultancy",
      "Industry": "Construction",
      "Revenue": 9000,
      "IsTarget": false,
      "RegionCode": "Region",
      "OrganizationType": "Commercial",
      "TotalAssetsAmount": 456,
      "TotalLiabilitiesAmount": 78,
      "DoingBusinessAs": "TCDM Group",
      "Rating": 5,
      "EmployeeCount": 80000,
      "LifetimeValue": 100,
      "StartDate": "20100121",
          "CustomerSince": "20101207"
```

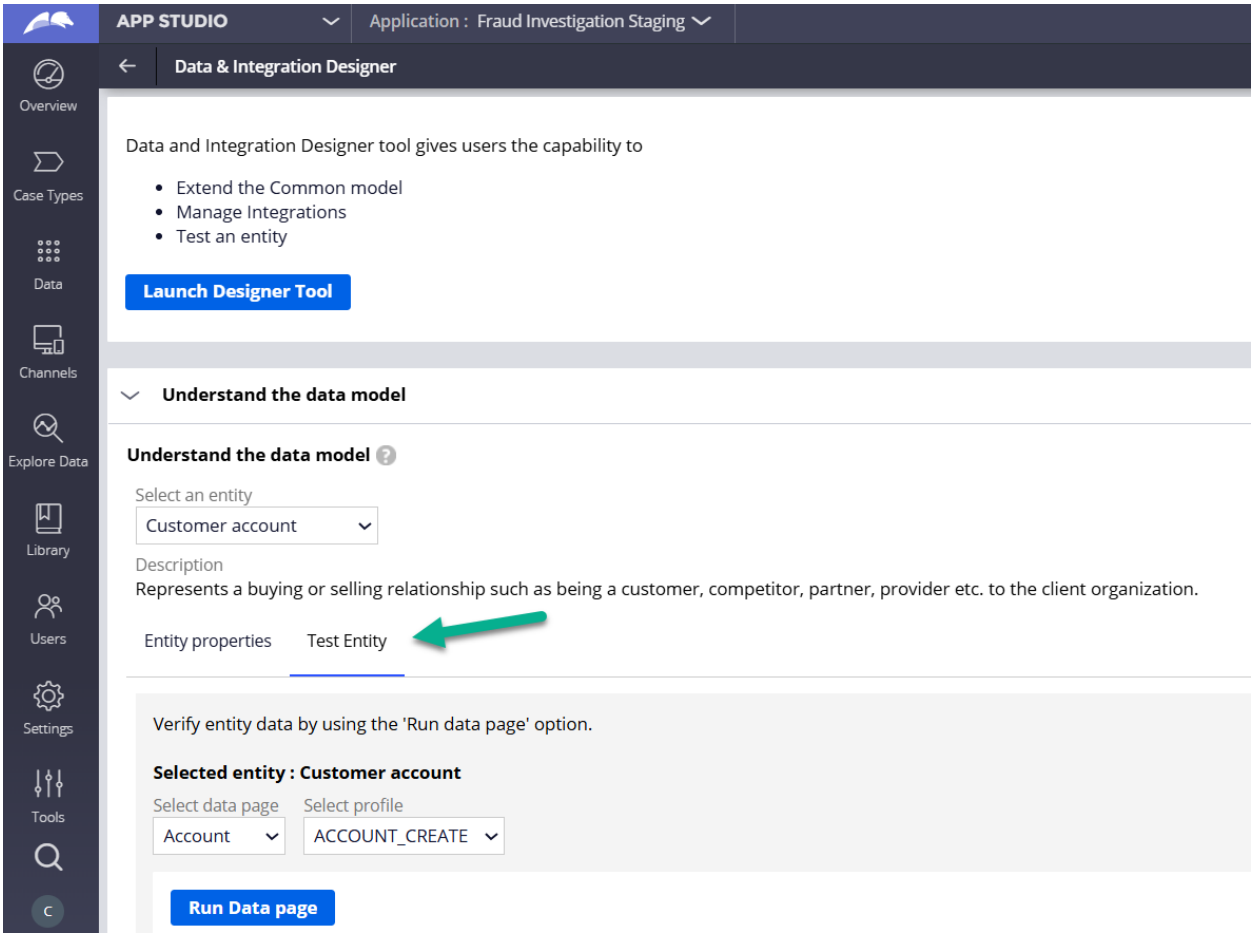

6. Upload the updated or newly created request file and click "Submit" to invoke the source.



7. Once submitted, the results will be displayed on the screen.



NOTE - The **Test Entity** feature can also be accessed directly from the **Understand the Data Model** section of the Data and Integration Designer landing page, without needing to initiate a Designer case.



Known Limitations

Generic limitations

- When creating multiple entities using different branches, ensure that all branch conflicts are resolved for rules such as the `ClassSettings` data transform, `Determine API Source` decision table, etc.

- If Entity 1 is created in Branch A and Entity 2 is created in Branch B, updating the portal rule for Entity 2 will fail, as portal rules cannot be saved across multiple branches. As a result, Entity 2 will not appear in the portal's create menu.
  Platform bug: BUG-877652

  Workaround: Use the same branch for both entities, or merge the first entity's branch before starting to work on the second entity.

- Within the Common Data portal, understand that the Data Model landing page will not be functional.
  Workaround: In App Studio, Understand the Data Model landing page is available on the Designer landing page.

Extend the model stage

- If a class exists in Branch A, but the branch preference is set to Branch B when creating a property in that class, it will result in errors.
  Workaround: Use the same branch for creating properties as the one in which the class is defined.

- Generating an entity in one branch and attempting to manage it from another by changing the branch preferences is not supported, as rules created in one branch cannot be saved to another.
  Workaround: Use the same branch to manage the entity.

- Users may occasionally encounter a persistent loading screen when attempting to add a property or edit a view. This issue typically arises when a user continues working in an outdated session (or) when the application's PegaAAT token has expired. By default, session tokens expire every 15 minutes. Once a user encounters this issue, a fresh PegaAAT token will be generated, and the user can refresh the screen and open the previous Data & integration designer case and continue the work.

  Workaround: If a user plans to work on property addition or view editing while extending the model for longer than 15 minutes, it is advisable to manually increase the token's expiry duration to avoid interruptions. Please refer to the following documentation to increase the token expiry time.

  https://docs.pega.com/bundle/platform/page/platform/security/enhanced-refresh-token-strategy.html

Manage Relationship stage

- By default, PEGA does **not** automatically create BLOB (pzpvstream) columns for link tables (new relation class's table). In a scenario where a relationship includes **unoptimized properties** that require BLOB storage, users may need to manually create the corresponding BLOB column in the corresponding relation table.

Manage Integration stage

- In the **Manage Integration** stage, if no sample payloads are uploaded, visual mapping is not supported.
  Workaround: Upload request/response samples or use the **Review Configuration** stage to manage the mappings.

- In the **Map Request/Response** step within the Manage Integration stage, you cannot directly map a top-level property to a nested element without first defining the top-level page or page list.
  Workaround: Use the **Review Configuration** stage to configure advanced mappings.

- In the **Map Request/Response** step, nested fields such as `.Address.Line` cannot be directly referenced.
  Workaround: Use the **Review Configuration** stage to configure advanced mappings.

Manage Sample Data stage

- In **Manage Sample Data**, a template can be saved only **once** per Designer case.
  Workaround: Upload the template as a binary file from Dev Studio.