



Using PCore and PConnect Public APIs '24.2

5 September 2025

CONTENTS

Invoking a savable data page	3
------------------------------	---

Invoking a savable data page

Store data in a data page by invoking a savable data page.

1. In the navigation pane of Dev Studio, click Records > Data Model > Data Page, and open a savable data page.
2. In the Definition tab of the data page, under Data Sources, enter the required data, and click Save.

The screenshot shows the 'Edit Data Page: Savable Movie' configuration in Pega Dev Studio. The interface includes a top navigation bar with tabs like Home, OPG272-Cred-D..., and D_MovieSavable. Below the navigation bar, there's a red warning banner stating 'This record has 1 severe or moderate warning (including 1 unjustified)' with a 'Review/Edit' button. The main content area is divided into two sections: 'Data page definition' and 'Data Sources'.

Data page definition:

- Structure: Page
- Object type: OPG272-Cred-Data-Movie
- Mode: Savable
- Scope: Thread

Data Sources:

If there are multiple sources, the first IF that evaluates to true is used. If none evaluate to true, the OTHERWISE source is used.

Source	Class name	Response Data Transform
Lookup	Movie	

A red box highlights the 'Lookup' source configuration. Below the table, there's a '+ Add alternative source' link.

Data sources

3. In the Parameters tab of the data page, add the required parameters.
4. Select the Is this page used for alternate key storage? checkbox.
 - If the data object is configured with an auto-generated key, enter the `pyGUID` key in the Linked field text box, and click Save.

Edit Data Page: Savable Movie [Available]
ID: D_MovieSavable RS: SkyC11n:01-01-01

This record has 1 severe or moderate warning (including 1 unjustified) [Review/Edit](#)

Definition Load Management Security **Parameters** Pages & Classes Test cases Usage Specifications History

☒ Is this page used for alternate key storage?

Parameters

Name	Description	Data type	Linked field	Required	In/Out	Default value	Smartprompt type	Validate as
1 pyGUID	Globally unique ID	String	pyGUID	Yes	In		OZXRLM-SkyC11n-D	

+

Alternate key details - auto-generated key

- If the data object is configured with custom keys, enter the names of the custom keys in the Linked field text box, and click Save.

Edit Data Page: Savable Movie [Available]
ID: D_MovieSavable RS: SkyC11n:01-01-01

This record has 1 severe or moderate warning (including 1 unjustified) [Review/Edit](#)

Definition Load Management Security **Parameters** Pages & Classes Test cases Usage Specifications History

☒ Is this page used for alternate key storage?

Parameters

Name	Description	Data type	Linked field	Required	In/Out	Default value	Smartprompt type	Validate as
1 MovieId	Id	String	MovieId	Yes	In		OZXRLM-SkyC11n	
2 MovieName	Name	String	MovieName	Yes	In		OZXRLM-SkyC11n	

+

Alternate key details - custom keys

5. In the Definition tab of the data page, under Data save options, configure the save plan in the order shown in the below figure.

Data save options

If there are multiple save options, every WHEN is evaluated and the OTHERWISE save option is used only if all WHENs evaluate to false.

1	WHEN	pylsDelete	Parameters
	Save type*	Class name*	Data transform
	Database delete	Movie	
		OPG272-Cred-Data-Movie	
	<input type="checkbox"/> Local save		
2	WHEN	pylsPatch	Parameters
	Save type*	Class name*	Data transform
	Database update	Movie	
		OPG272-Cred-Data-Movie	
	<input type="checkbox"/> Local save		
3	OTHERWISE		
	Save type*	Class name*	Data transform
	Database save	Movie	
		OPG272-Cred-Data-Movie	
	<input type="checkbox"/> Local save		

+ Add save option

Configuring save plan

- In the UX tab of the corresponding data type, click the Other Views tab, and under the **Form** section, click Edit.

The screenshot shows the Pega Dev Studio interface. On the left, the 'Data types' panel lists various data types, with 'Movie' selected. The main workspace shows the 'UX' tab for the 'Movie' data type. The 'Other Views' sub-tab is active, displaying a list of views: Full, List, Partial, and Form. Under the 'Form' section, the 'Edit' button is highlighted with a red box.

Edit form

- In the Edit View, configure the submittable fields of the data type and click Save.

The screenshot shows the Pega UI interface for configuring a data type. On the left, a sidebar lists various data types like EmployeeDetails, Movie, Naya data, Product, StudentDetails, and Transactions. The main area is titled 'Data Type: Movie' and shows a 'Preview as' section with a 'Constellation' dropdown and a 'Movie Name' input field. On the right, the 'Edit View: Edit' panel is open, showing a 'Fields' section with a list of fields. The 'Movie Name' field is highlighted with a red box. Below the fields list, there is an 'Add' button and an 'Advanced' section.

Configure submittable fields



NOTE: Since primary fields exist in the edit view by default, the fields marked as primary can be updated using this procedure. Since DX APIs allow modifying these fields, mark only those fields as primary that can be modified by the client.

- To store the data, call the `PCore.getRestClient().invokeRestApi` API as shown below.

```
PCore.getRestClient().invokeRestApi("updateDataObject", {
    queryPayload: {
        data_view_ID: "< datapageName >"
    },
    body: {
        data: {
            formdata,
            key
        }
    },
});
```

Example

In this example, the API stores the `MovieName` property in the `D_MovieSavable` data page by passing the `pyGUID` key as a linked field parameter.

```
PCore.getRestClient().invokeRestApi('updateDataObject',{
    queryPayload: {
        data_view_ID: "D_MovieSavable"
    },
    body:{
        data: {
            pyGUID: 'ee793054-a766-49e7-b3f7-fdb44cb266a7',
            MovieName: 'Avengers'
        }
    }
}).then(response => {
    // Below is the response structure
    /**
    {
        data: {
            responseData: {
                // Record Data
                "MovieName": "Avengers",
                "MovieId": "234",
                "pyGUID": "ee793054-a766-49e7-b3f7-fdb44cb266a7",
                "MovieDescription": "Description Avengers",
                ...
            }
        },
        status:200,
        headers:{
            content-type: "application/json;charset=UTF-8"
        },
```

```
statusText: ""  
}  
*/  
});
```

Result:

The data is now stored in the data page.

Related information

- [Configuring CRUD data endpoints with Constellation DX API](#)