Creatio Academy

Creatio IDE

User task

Version 8.0

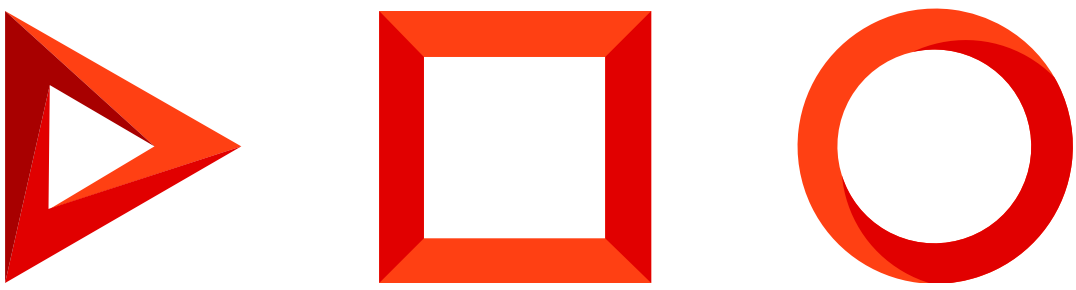


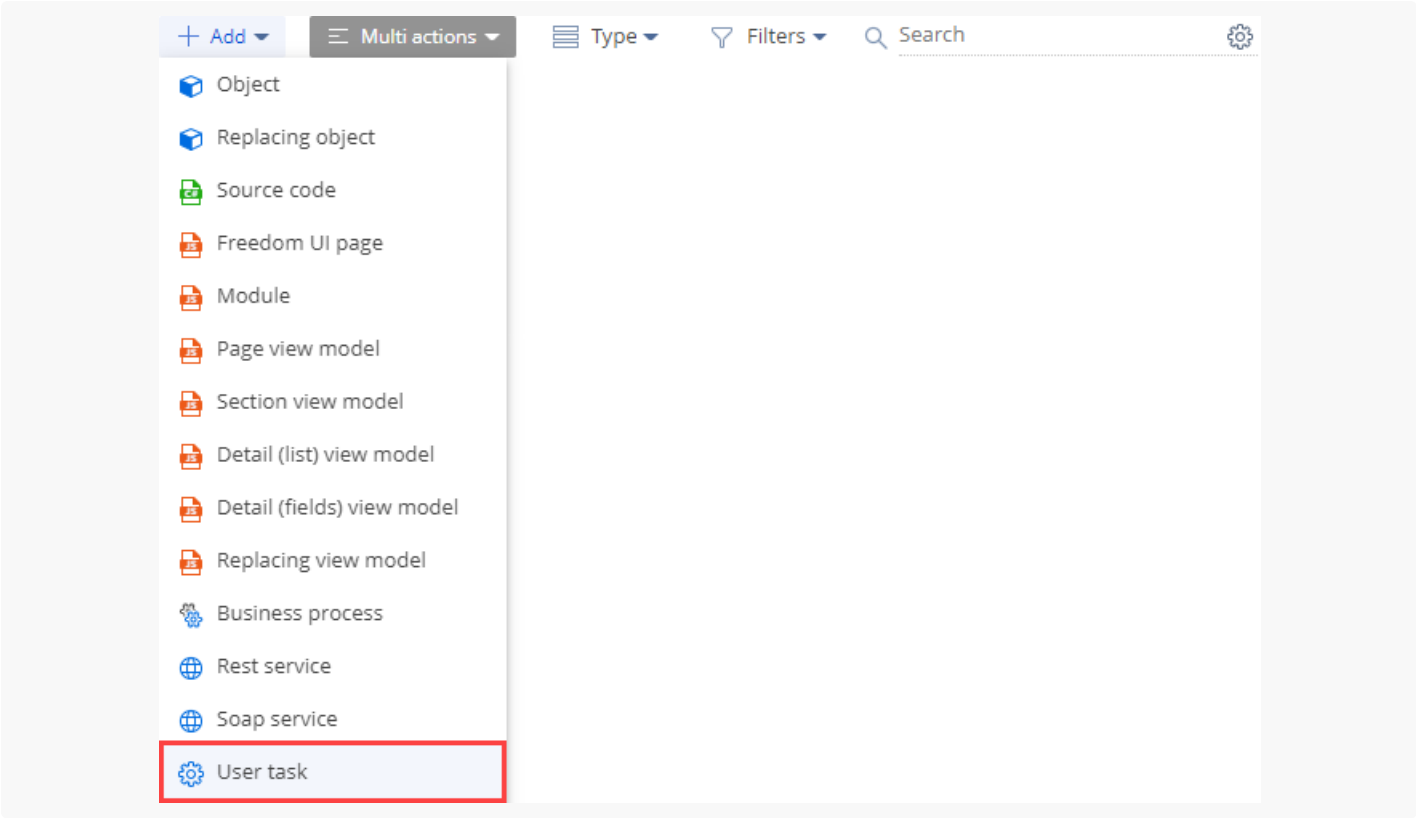
Table of Contents

User task

Beginner

Configuration element of the [*User task*] type is an entity that lets you implement routine operations and execute them in business processes. Learn more about business processes in a user documentation guide: BPM tools. [*User task*] process element lets you select the needed task type. The schema of the [*User task*] type configuration element implements this task type. Learn more about the [*User task*] element in user documentation: [*User task*] process element.

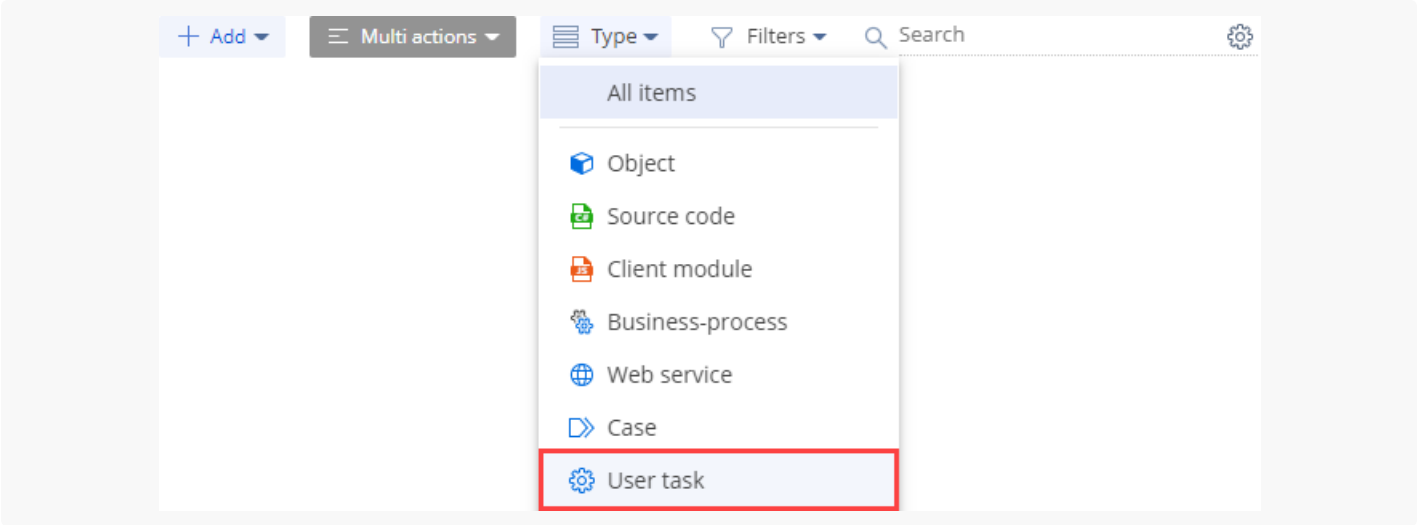
The item of the [*Add*] drop-down list in the toolbar of the [*Configuration*] section workspace represents the user task schema you can add in Creatio IDE.



Learn more about configuration element types in a separate article: Operations in Creatio IDE.

The [*User task*] type schema in the [*Type*] drop-down list in the toolbar of the [*Configuration*] section workspace represents the configuration element of the [*User task*] type. A **schema** is the basis of Creatio configuration.

View the user task schema **type** in the figure below.



Learn more about configuration element types in a separate article: Operations in Creatio IDE.

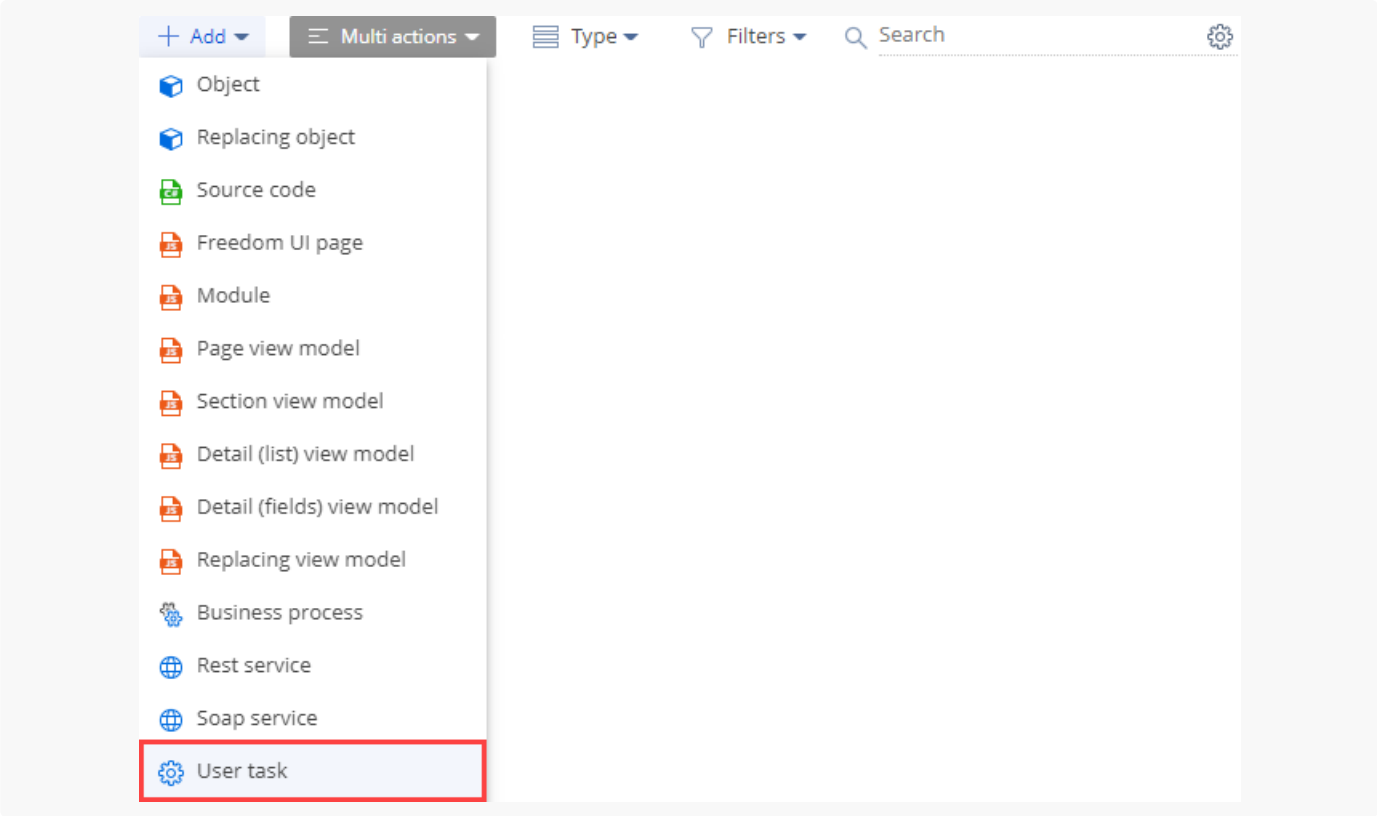
Implement a user task

By default, Creatio includes a set of preconfigured custom user tasks. Creatio IDE lets you implement a custom user task. In its simple implementation, the user task mechanism is partially similar to the [*Script task*] process element. Learn more about the [*Script task*] element in user documentation: [*Script task*] process element.

Creatio lets you use the implemented custom user task in various business processes. When you change the custom user task, Creatio applies the changes to every business process that contains the task.

To **implement a user task**:

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*User task*] on the section list toolbar.



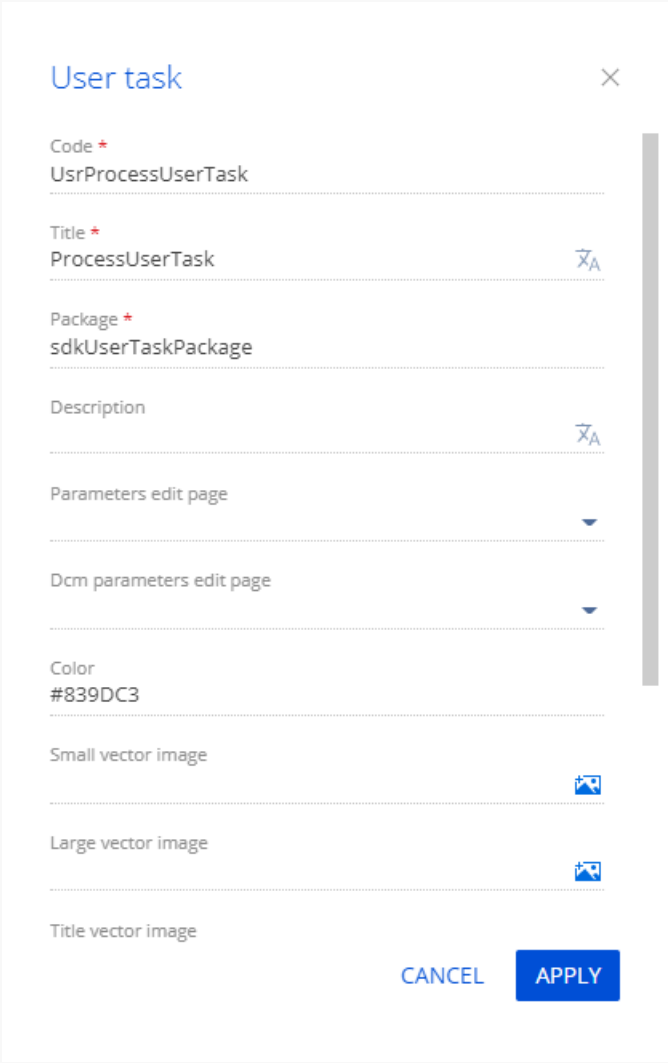
3. Fill out the schema properties in the User Task Designer.

   The **main schema properties** are as follows:

   - [*Code*] is the schema name. Required. Starts with the prefix specified in the [*Prefix for object name*] (`SchemaNamePrefix` code) system setting, `Usr` by default. Can contain Latin characters and digits. When a configuration element schema is created, the prefix specified in the [*Prefix for object name*] (`SchemaNamePrefix` code) system setting is added to the current field automatically. Creatio checks for the prefix and whether it matched the system setting value when saving the schema properties. If the prefix is missing or does not match, the user receives a corresponding notification.
   - [*Title*] is the localizable schema title. Required. The title of the configuration element schema is generated automatically and matches the value of the [*Code*] property without a prefix.
   - [*Package*] is the custom package where you create the schema. The property is populated automatically and non-editable.
   - [*Description*] is the localizable schema description.
   - [*Parameters edit page*] is the user task's parameter setup page. Creatio displays the page when you set up an element in a business process. Create the page before setting up the user task. Otherwise, the list of user task parameters will be displayed.
   - [*Dcm parameters edit page*] is the user task's parameter setup page. Creatio displays the page when you set up the element in the Case Designer. Create the before settings up the user task. Otherwise, the list of user task parameters will be displayed.
   - [*Color*] is the hex color code. It affects the appearance of the current configuration element in the Business Process Designer. By default, "`#839DC3`."
   - [*Small vector image*] is the user task icon in the action menu. Size: 24x24 pixels or larger proportionally.
   - [*Large vector image*] is the user task icon in the business process schema. Size: 69x55 pixels or larger

proportionally.

- [*Title vector image*] is the user task icon in the element setup area of the Business Process Designer. Size: 42x42 pixels or larger proportionally.
- [*Small vector image for DCM*] is an image that affects the appearance of the user task in the Business Process Designer.



Click [*Apply*] to apply the properties.

As a result, Creatio will save the properties of the [*User task*] type configuration element. The User Task Designer will close. When you reopen the schema of the [*User task*] type configuration element, you will see the C# code generated upon saving in the workspace of the User Task Designer.

The **properties area** of the User Task Designer lets you:

- edit the main schema properties (button)
- specify the additional schema properties (button)

The **additional schema properties** are as follows:

- [*Localizable strings*]

- [*Parameters*]. User task parameters return the task result. To add parameters, follow the instructions in a different step: Add a user task parameter.

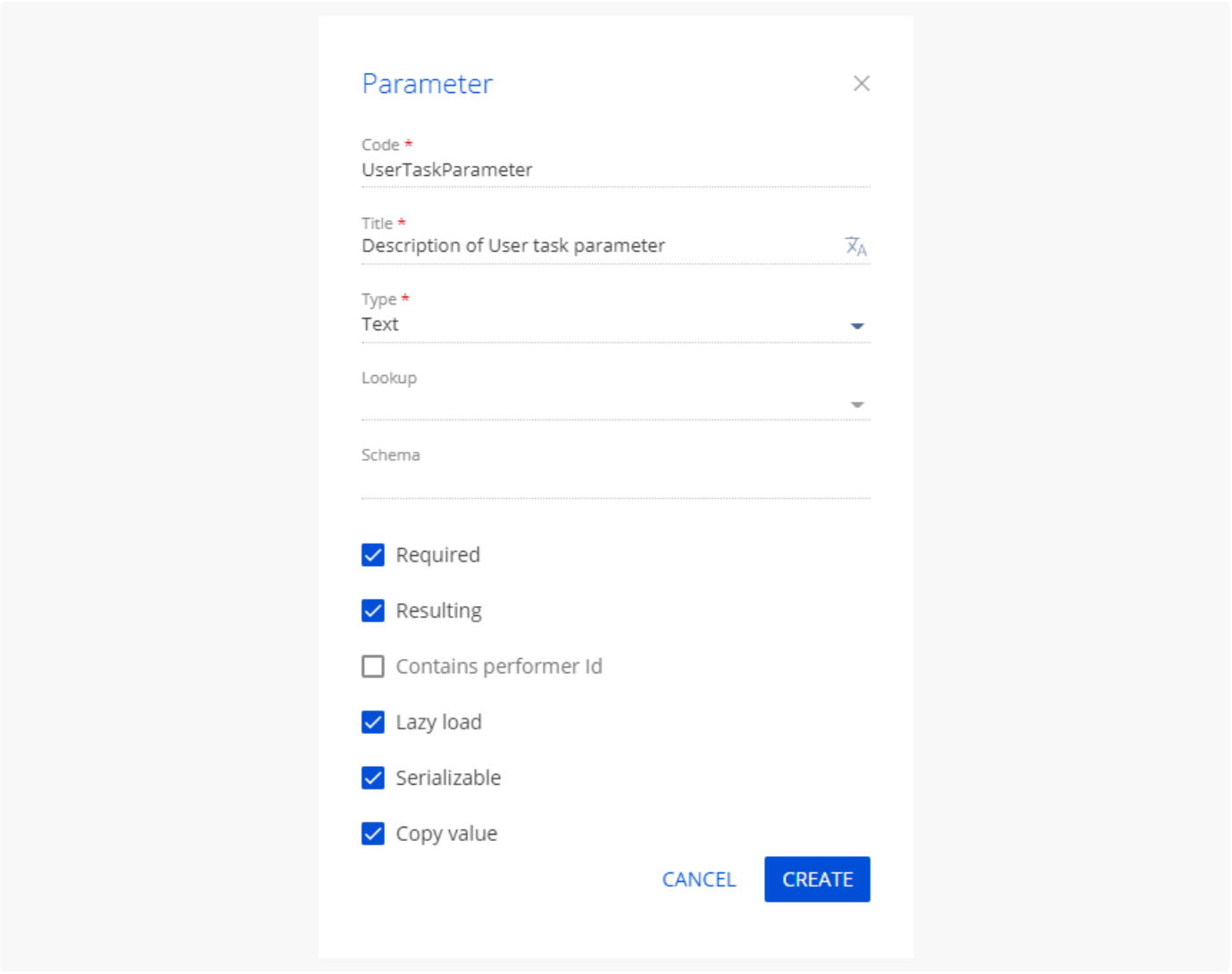
4. Add the source code in the User Task Designer. To implement the business logic of the user task, follow the instructions in a different step: Implement the business logic of the user task.
5. Click [*Save*] on the User Task Designer's toolbar to save the changes to Creatio metadata temporarily.
6. Click [*Publish*] on the User Task Designer's toolbar to apply the changes to the database level.

Add a user task parameter

1. Click + in the properties area of the [*Parameters*] node's context menu.
2. Fill out the parameter properties in the User Task Designer.

   The **main parameter properties** are as follows:

   - [*Code*] is the parameter name. Required.
   - [*Title*] is the localizable parameter title. Required.
   - [*Type*] is the parameter type. Required. Depending on the selected value, you can link the current parameter to other user task parameters.
   - [*Lookup*] is a lookup. The property becomes available if you select "Lookup" in the [*Type*] property.
   - [*Schema*] is the name of the schema connected to the parameter. The property is non-editable.
   - [*Required*] specifies the parameter is required.
   - [*Resulting*] specifies the parameter is resulting, i. e., to enable users to use the values of the current parameter in the condition of a conditional flow.
   - [*Contains performer Id*] specifies the ID of the contact who performs the task in the parameter. The property becomes available if you select "Lookup" in the [*Type*] property.
   - [*Lazy load*] specifies the load the process diagram without delays. If you select the checkbox, Creatio loads the parameter immediately before using it. Learn more about lazy loading in Wikipedia.
   - [*Serializable*] specifies the save the value of the current parameter as part of the user task execution, i. e., between `InternalExecute()` and `CompleteExecuting()` method calls. Required for interactive user tasks. Learn more about user task types in a different step: Implement the business logic of the user task.
   - [*Copy value*] specifies the enable copying the parameter value when reusing it. The property is deprecated. We do not recommend using it.



Click [*Create*] to add a parameter.

User Task Designer lets you perform the following parameter **actions**:

- Edit parameter properties (button).
- Delete the parameter (button).

Implement the business logic of the user task

The following custom **classes** let you implement the business logic of the user task:

- The class that inherits from the `Terrasoft.Core.Process.ProcessUserTask` class. The **purpose** of the class is to implement the base user task mechanism. The class matches the [*Code*] property of the user task schema.
- The class that inherits from the `Terrasoft.Core.Process.ProcessUserTaskSchemaExtension` class. The **purpose** of the class is to implement additional capabilities of the user task, for example, parameter synchronization, dependency setup, etc. To **create a name of the inheritor class**, add a `SchemaExtension` suffix to the name of the user task class. For example, the name of the inheritor class for the `PerformHardWorkUserTask` user task is `PerformHardWorkUserTaskSchemaExtension`.

Learn more about the `ProcessUserTask` class methods in a separate article: ProcessUserTask class. Learn more about the `ProcessUserTaskSchemaExtension` class methods in a separate article: ProcessUserTaskSchemaExtension class.

The methods that the User Task Designer lets you replace in the inheritor class of the `ProcessUserTask` class depend on the user task type. View user task **types** in the table below.

User task types

User task type	Description	Replaced methods

Interactive	User or Creatio input is required to complete the user task. Interaction includes entering additional data, selecting a mode, and waiting for the object signal.	`InternalExecute()`
		`CompleteExecuting()`
		`CancelExecuting()`

Non-interactive	User input is not required to complete the user task.	`InternalExecute()`

Add a user task to the [*Process element*] tab

If you are going to use the custom user task element often, add it to the [*Process element*] tab of the Process Designer to streamline the workflow.

To **add a user task** to the [*Process elements*] tab of the **Process Designer**:

1. Execute the SQL script provided below in the database.

SQL script

Microsoft SQL

```
insert into SysProcessUserTask(SysUserTaskSchemaUId, Caption)
select s.UId, s.Caption from SysSchema s
where s.Name = 'UsrSomeProcessUserTask'
```

PostgreSQL

```
INSERT INTO "SysProcessUserTask" ("SysUserTaskSchemaUId", "Caption")
VALUES
(
    SELECT s."UId", s."Caption" FROM "SysSchema" AS s
    WHERE s."Name" = 'UsrSomeProcessUserTask'
)
```

`UsrSomeProcessUserTask` is the name (the value of the [*Code*] property) of the user task schema.

2. Log out of and log back in to Creatio or compile Creatio. Learn more about compilation in a separate article: Operations in Creatio IDE.

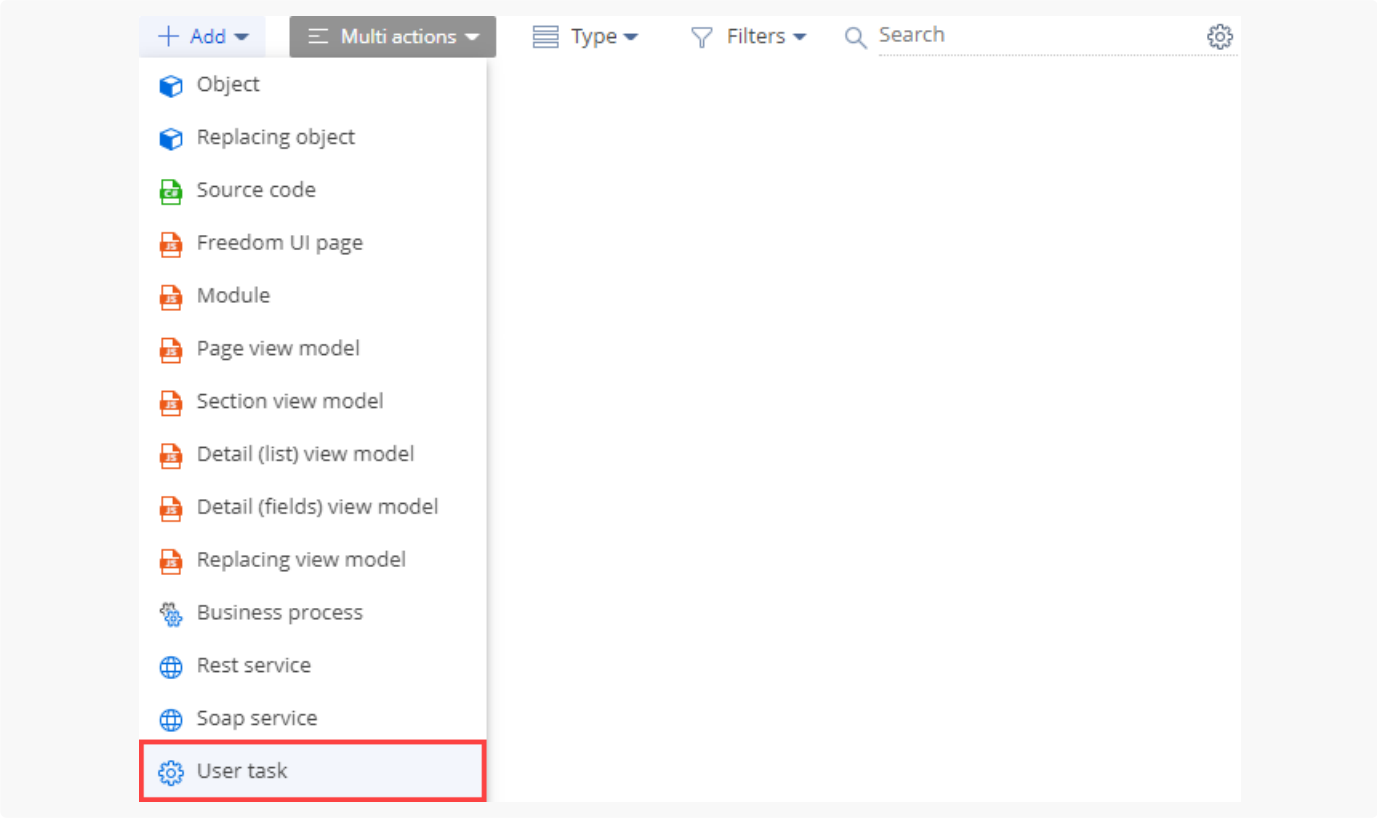
Implement a custom user task

Medium

Example. Implement a custom [*User task of sum numbers*] user task. The task must calculate the sum of two integers that are specified in the task parameters. Display the calculation result on an auto-generated business process page.

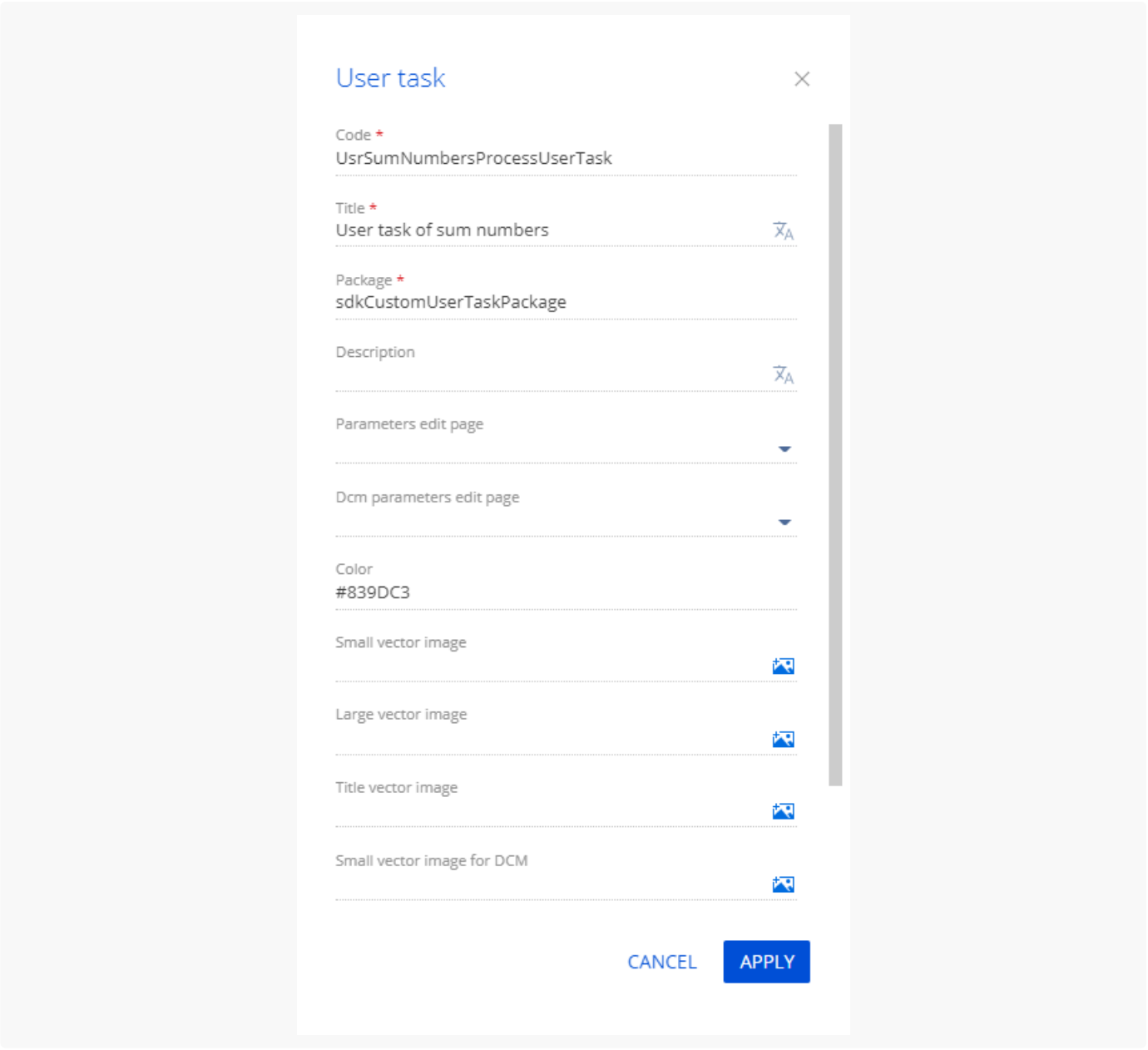
1. Create a user task schema

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*User task*] on the section list toolbar.



3. Fill out the **schema properties** in the User Task Designer.

   - Set [*Code*] to "`UsrSumNumbersProcessUserTask`."
   - Set [*Title*] to "`User task of sum numbers`."



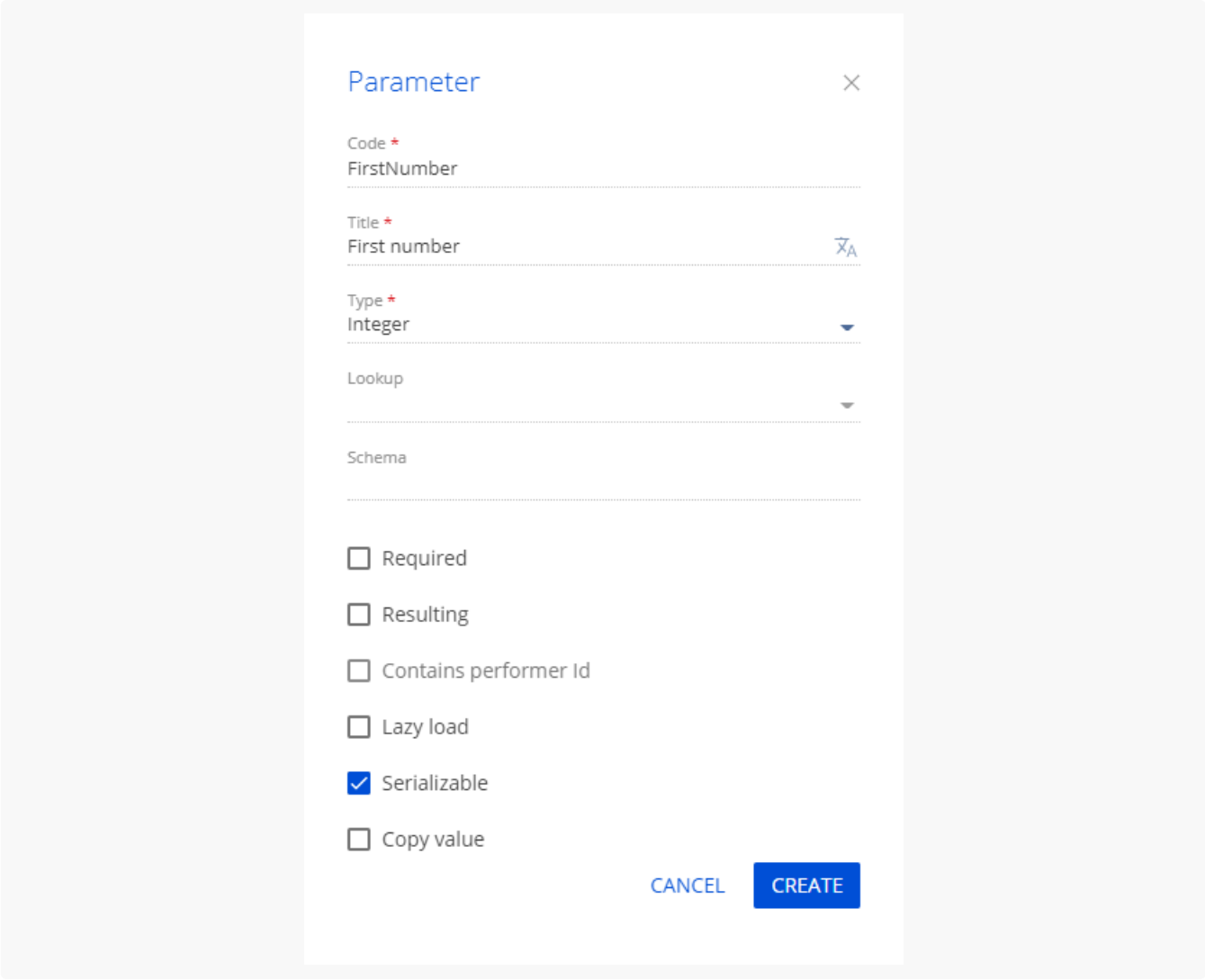
Click [*Apply*] to apply the properties.

4. Add a **user task parameter that contains the first integer**.

   a. Click + in the properties area of the [*Parameters*] node's context menu.

   b. Fill out the **parameter properties** in the User Task Designer.

   - Set [*Code*] to "`FirstNumber`."
   - Set [*Title*] to "`First number`."
   - Set [*Type*] to "`Integer`."
   - Set the [*Serializable*] checkbox.



g. Click [*Create*] to add a parameter.

5. Add **user task parameters that contain the following** in a similar way:

   - second integer
   - integer sum

   View the properties of the parameters to add in the table below.

Parameter property values

Parameter	Property	Property value
Parameter that contains the second integer	[*Code*]	`"SecondNumber"`
	[*Title*]	`"Second number"`
	[*Type*]	Select `"Integer"`
	[*Serializable*] checkbox	Set
Parameter that contains the integer sum	[*Code*]	`"SumNumbers"`
	[*Title*]	`"Sum of numbers"`
	[*Type*]	Select `"Integer"`
	[*Serializable*] checkbox	Set

6. Implement the **logic to add integers**. To do this, implement the `InternalExecute()` method of the user task schema's automatically generated source code.

`InternalExecute()` method

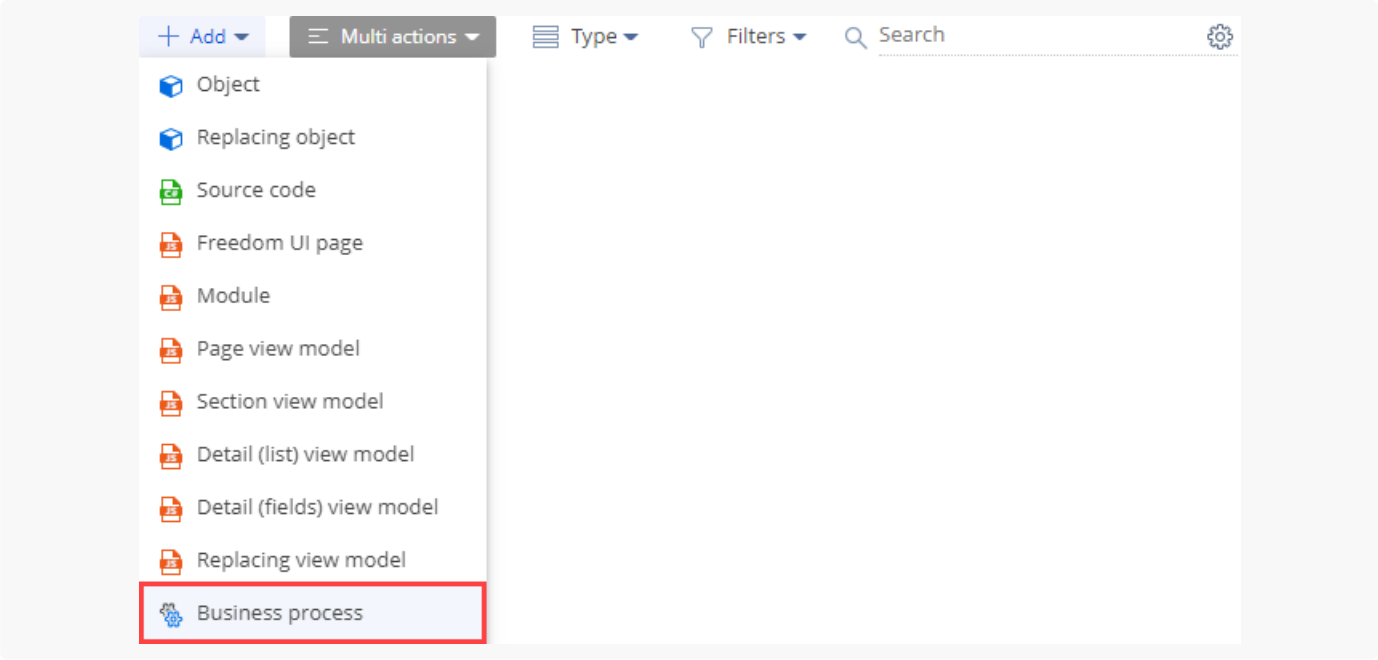
```
protected override bool InternalExecute(ProcessExecutingContext context) {
    /* Execute operations with task parameters. */
    SumNumbers = FirstNumber + SecondNumber;
    /* Specify that the task was completed successfully. */
    return true;
}
```

Full source code of the user task schema

7. Click [*Publish*] on the User Task Designer's toolbar to apply the changes to the database level.

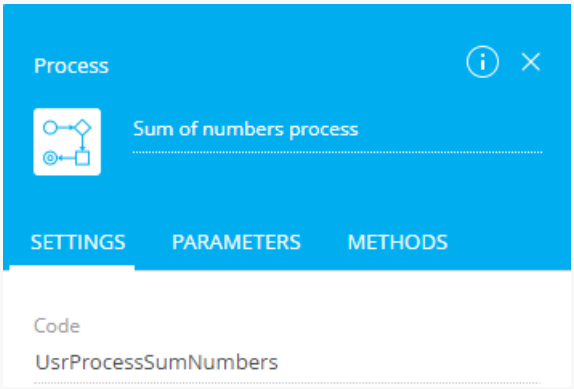
2. Implement a business process

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Business process*] on the section list toolbar.



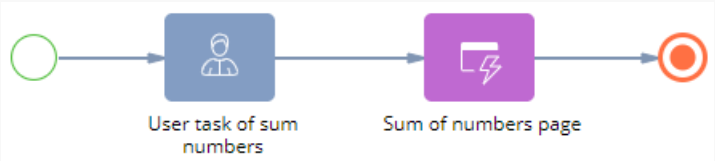
3. Fill out the **properties of the business process**:

   - Set the [*Title*] property in the element setup area to "`Sum of numbers process`."
   - Set the [*Code*] property on the [*Settings*] tab of the element setup area to "`UsrProcessSumNumbers`."



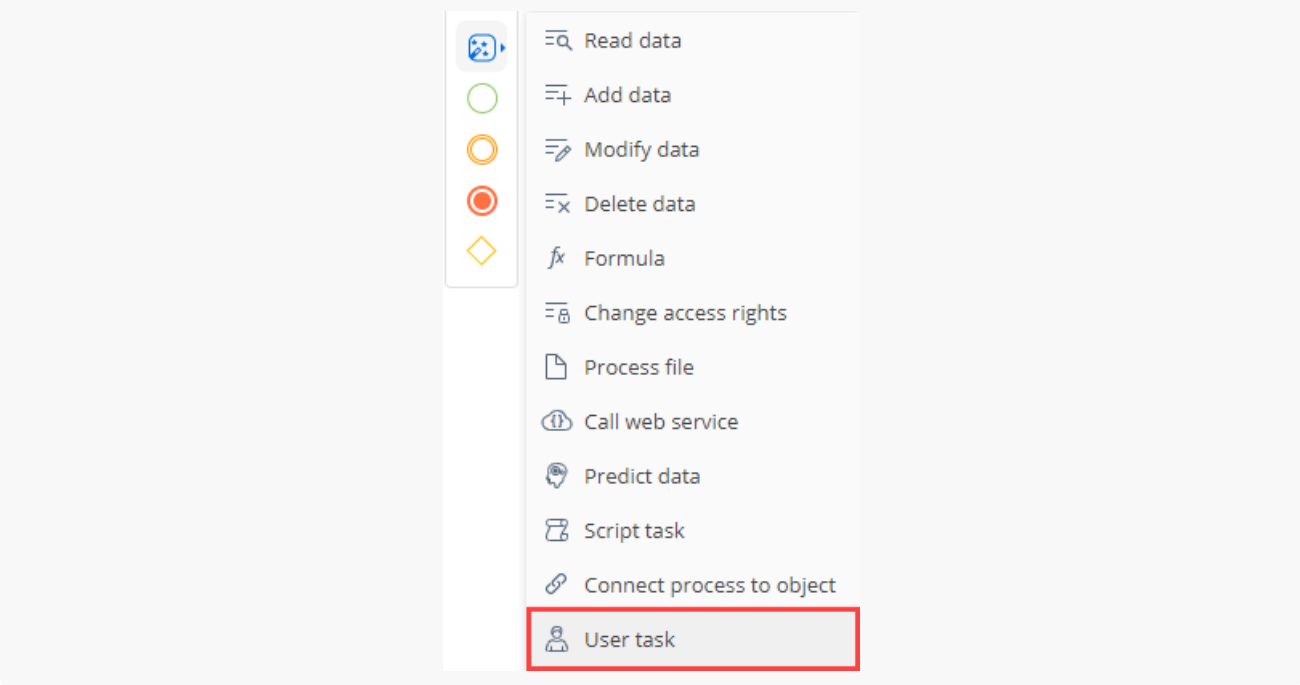
4. Implement the **business process**.

   View the business process in the figure below.



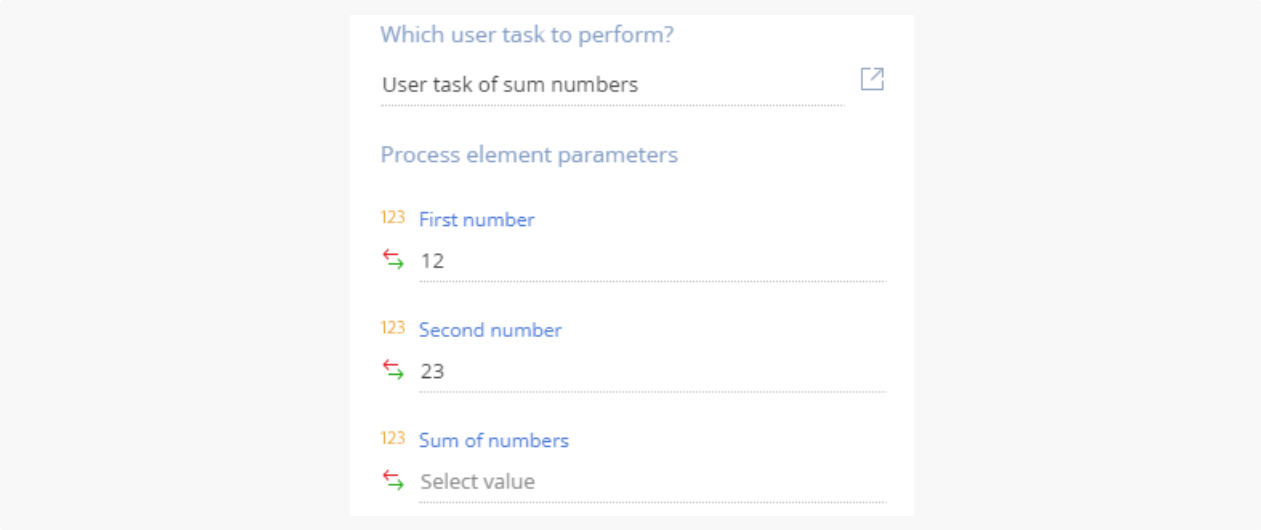
   a. Add the **user task**.

      a. Click [*System actions*] in the Designer's element area and place a [*User task*] element between the [*Simple*] start event and [*Terminate*] end event in the Process Designer's working area.



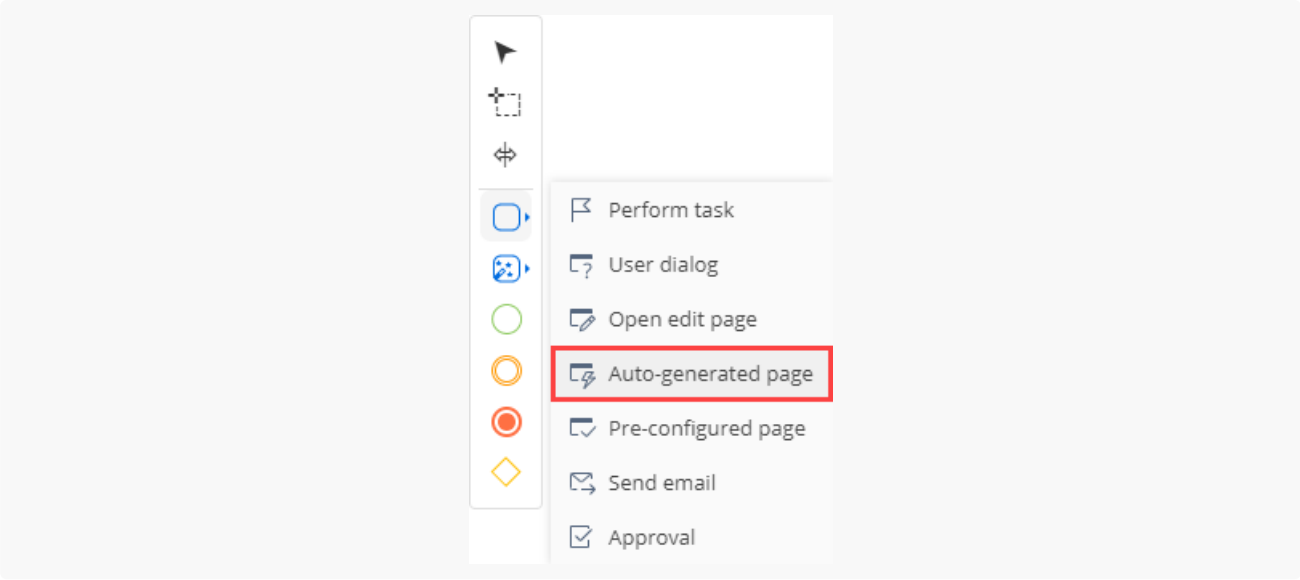
b. Fill out the **user task properties**.

- Select "`User task of sum numbers`" in the [*Which user task to perform?*] property.
- Fill out the **user task parameters**.
  - Set [*First number*] to "12."
  - Set [*Second number*] to "23."



b. Add an **auto-generated page**.

a. Click [*User actions*] in the Designer's element area and place an [*Auto-generated page*] element after the [*User task*] element in the Process Designer's working area.

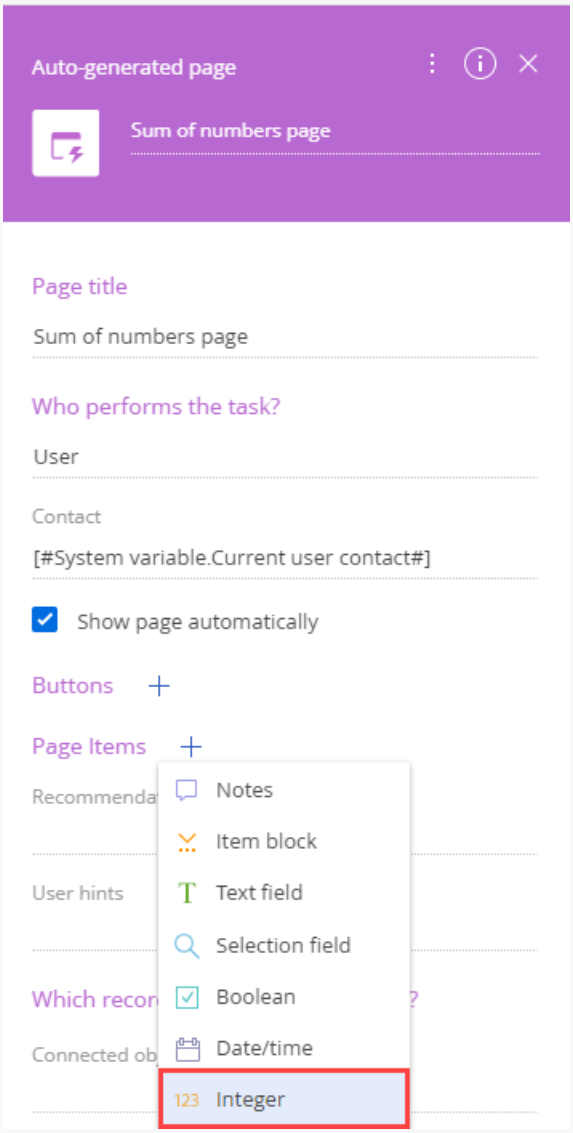


b. Fill out the **auto-generated page properties**.

- Set [*Title*] to "`Sum of numbers page`."
- Set [*Page title*] to "`Sum of numbers page`."

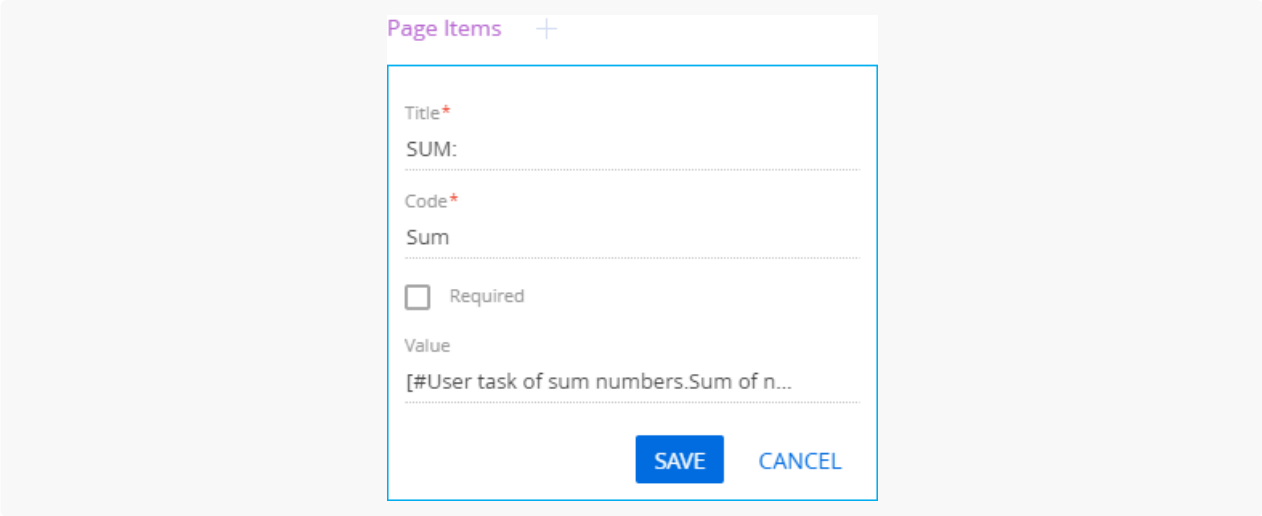
e. Add an **auto-generated page**.

a. Click + in the [*Page Items*] block and select "`Integer`."



b. Fill out the **item properties**.

- Set [*Title*] to "`SUM:`."
- Set [*Code*] to "`Sum`."
- Click → [*Process parameter*] → [*Sum of numbers*] in the [*Value*] property.



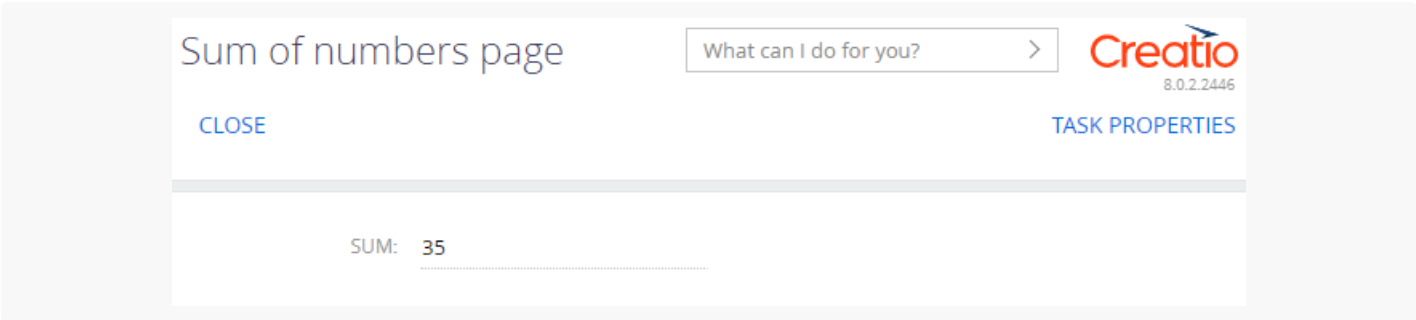
f. Click [*Save*] to add the parameter.

5. Click [*Save*] on the Process Designer's toolbar.

Outcome of the example

To **view the outcome of the example**, run the `Sum of numbers process` business process.

As a result, Creatio will open the [*Sum of numbers page*] auto-generated page. The page will display the sum of 2 integers that are specified in the business process parameters.



ProcessUserTask class C#

Medium

`Terrasoft.Core.Process` namespace.

`Terrasoft.Core.Process.ProcessUserTask` class implements the base user task mechanism. Replace methods in a custom class that inherits from the `ProcessUserTask` class. The methods that the User Task Designer lets you replace in the descendant class of the `ProcessUserTask` class depend on the user task type.

Methods

```
protected virtual bool InternalExecute(ProcessExecutingContext context)
```

Executes the base user task mechanism. If you need to complete the user task and call the `CompleteExecuting()` or `CancelExecuting()` method later, return `false`. Return `true` otherwise. If the method returns `true`, the element execution finishes (i. e., Creato sets the element status to [*Completed*]) and the process execution continues. If the method returns `false`, Creatio interrupts the execution of the element and process. However, the element retains the [*Running*] status. Replace the method for interactive and non-interactive user task types.

Parameters

`context`	The process execution scope. The `ProcessExecutingContext` type value.

`public virtual bool CompleteExecuting(params object[] parameters)`

Call after the user task completion. Must call the `base.CompleteExecuting(parameters)` base method. Returns `true`. Replace the method for interactive user task type.

Parameters

`parameters`	The array of parameters that are passed when calling `IProcessEngine.CompleteExecuting`.

`public virtual void CancelExecuting(params object[] parameters)`

Call when interrupting the process element execution. Performs a compensation, i. e., cancels activities created in the `InternalExecute()` method, linked records, etc. Replace the method for interactive user task type.

Parameters

`parameters`	Array of parameters. Required for backward compatibility.

`protected internal virtual void WriteExecutionData(IProcessExecutionDataWriter dataWriter)`

Passes additional data required when opening the page of the user task element. The `IProcessExecutionDataWriter` interface writes the values.

Parameters

`dataWriter`	Instance of the `IProcessExecutionDataWriter` interface. Serializes the values.

ProcessUserTaskSchemaExtension class

ProcessUserTaskSchemaExtension class C#

Medium

`Terrasoft.Core.Process` namespace.

`Terrasoft.Core.Process.ProcessUserTaskSchemaExtension` implements additional capabilities of the user task, for example, parameter synchronization, dependency setup, etc. Replace methods in a custom class that inherits from the `ProcessUserTaskSchemaExtension` class.

Methods

`public virtual Dictionary<Guid, string> GetResultParameterAllValues(UserConnection userConnectic`

Returns the index of available values for resulting user task parameters as `Dictionary<System.Guid, System.String>`.

Parameters

`userConnection`	User connection. The instance of the `UserConnection` class.
`userTask`	User task. The instance of the `ProcessSchemaUserTask` schema class. The class describes the schema of the element for which to call the current method.

`public virtual void SynchronizeDynamicParameters(UserConnection userConnection, ProcessUserTaskS`

Synchronizes dynamic parameters. **Dynamic parameters** are generated automatically for an element and are not included in the user task schema.

Parameters

`userConnection`	User connection. The instance of the `UserConnection` class.
`target`	User task. The instance of the `ProcessSchemaUserTask` schema class. The class describes the schema of the element for which to call the current method.

`public virtual void SynchronizeParameters(ProcessSchemaUserTask schemaElement)`

Synchronizes parameters created when implementing the user task in the Process Designer.

Parameters

`schemaElement`	The instance of the `ProcessSchemaUserTask` schema class. The class describes the schema of the element for which to call the current method.

```
public virtual void AnalyzePackageDependencies(ProcessSchemaUserTask schemaElement, IProcessSche
```

Sets dependencies of the linked user task schemas by calling the `ReportSchemaDependency()` method of the `dependencyReporter` parameter. Implemented for version 8.0.1 and later.

Parameters

`schemaElement`	The instance of the `ProcessSchemaUserTask` schema class. The class describes the schema of the element for which to call the current method.
`dependencyReporter`	An interface that includes methods for setting the schema or object column dependency.