# LIDL Modeling Tool user manual

We will take Counter as an example to introduce how to use the LIDL modeling tool to start from model element definition and finally draw Counter's interaction architecture diagram.

LIDL Modeling Tool path:

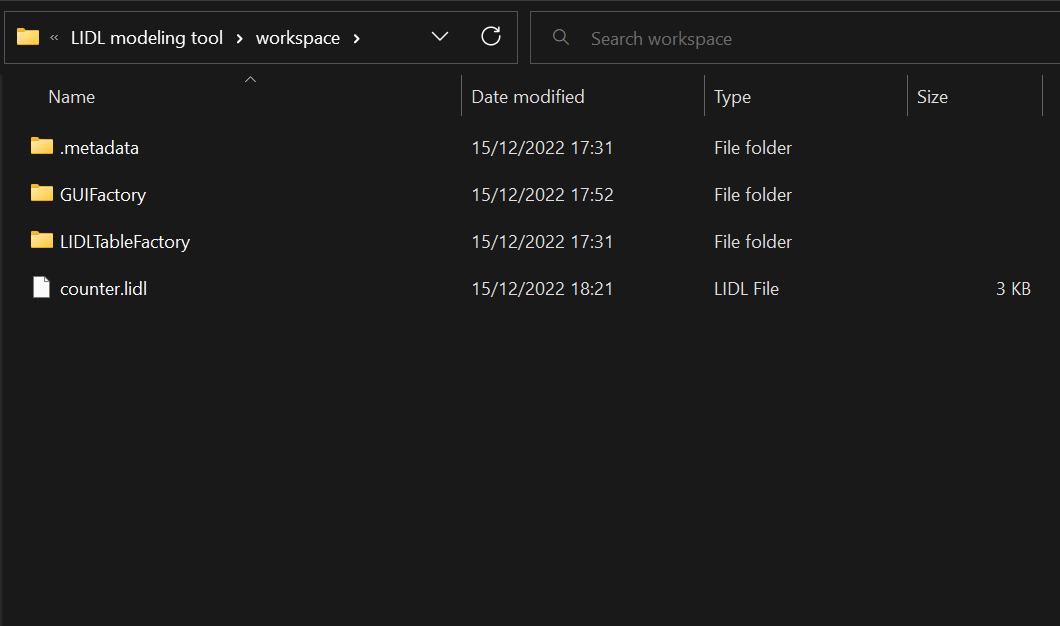
Zenodo: /LIDL MEC Toolkit/software/Modeling tool/LIDL modeling tool/

GitHub:/LIDL-MEC-Toolkit/software/Modeling tool/LIDL modeling tool/

## Sample data

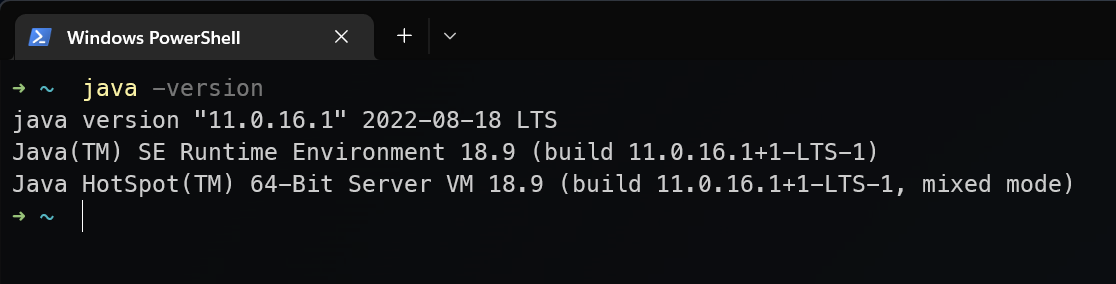
The model data files corresponding to the examples demonstrated in this user manual are in the directory of the LIDL modeling tool.

Under the workspace directory, the GUIFactory folder contains the model data for the interaction architecture diagram, the LIDLTableFactory contains the model data for the model element definition table, the .metadata is the software runtime log file, and the counter.lidl is the LIDL code file generated for this example.



## Software operating environment

We use java 11.0.16.1 when writing this manual.



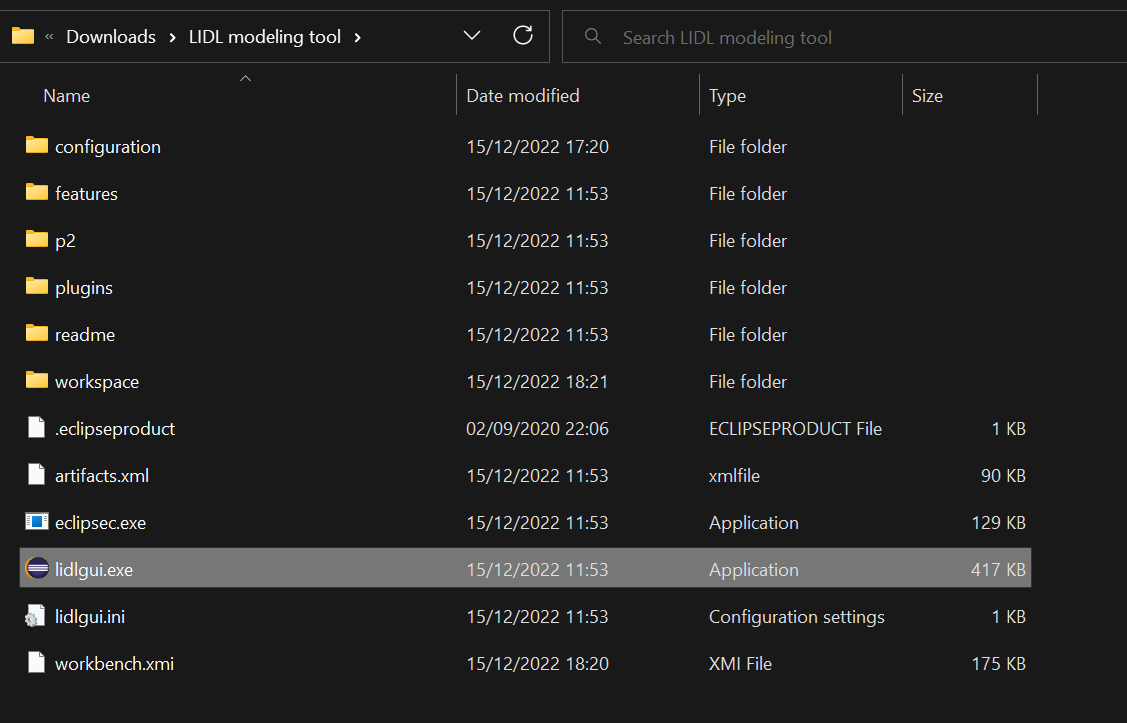
## Launch Modeling Tool

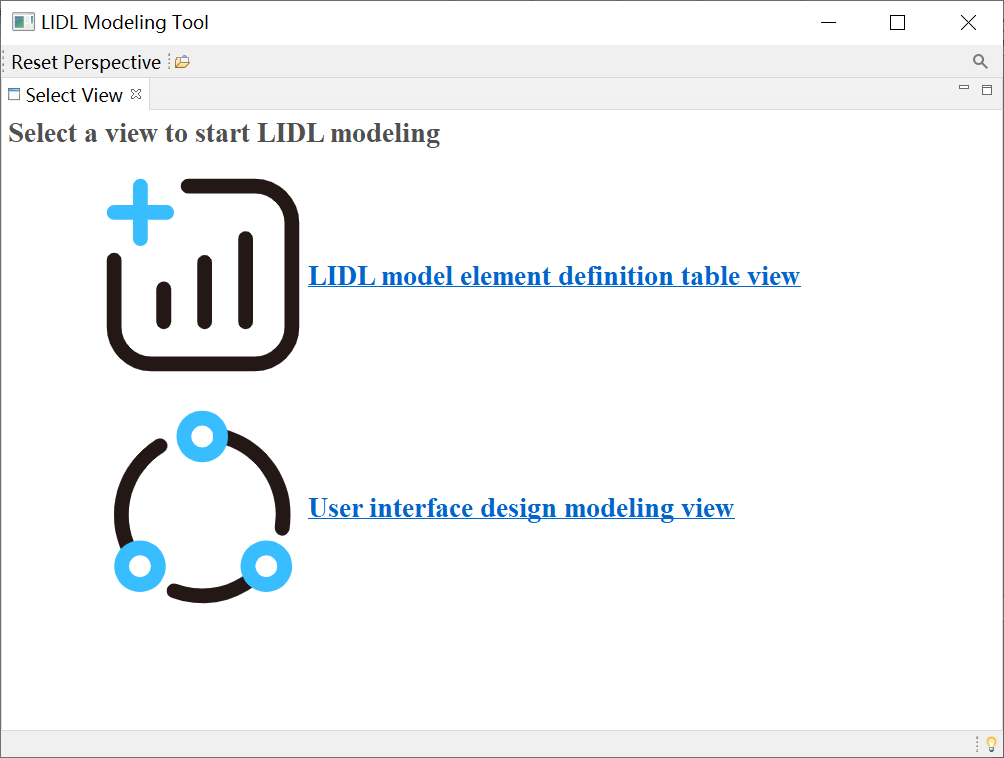
We provide the modeling tool in LIDL modeling tool, and it runs on Windows operating systems.

Zenodo: /LIDL MEC Toolkit/software/Modeling tool/LIDL modeling tool/

GitHub:/LIDL-MEC-Toolkit/software/Modeling tool/LIDL modeling tool/

Double-click lidlgui.exe in the extracted file to launch the modeling tool.





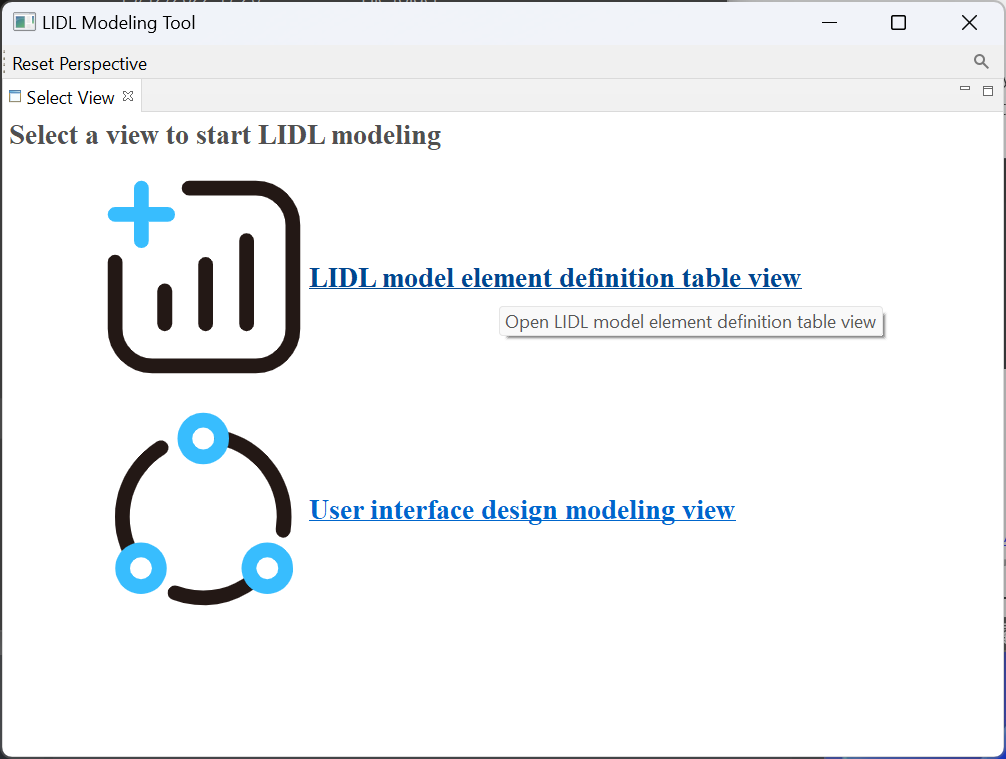
The modeling view in the above figure includes two views,

In the LIDL model element definition table view, users can define the required data types, interfaces, and interactions.

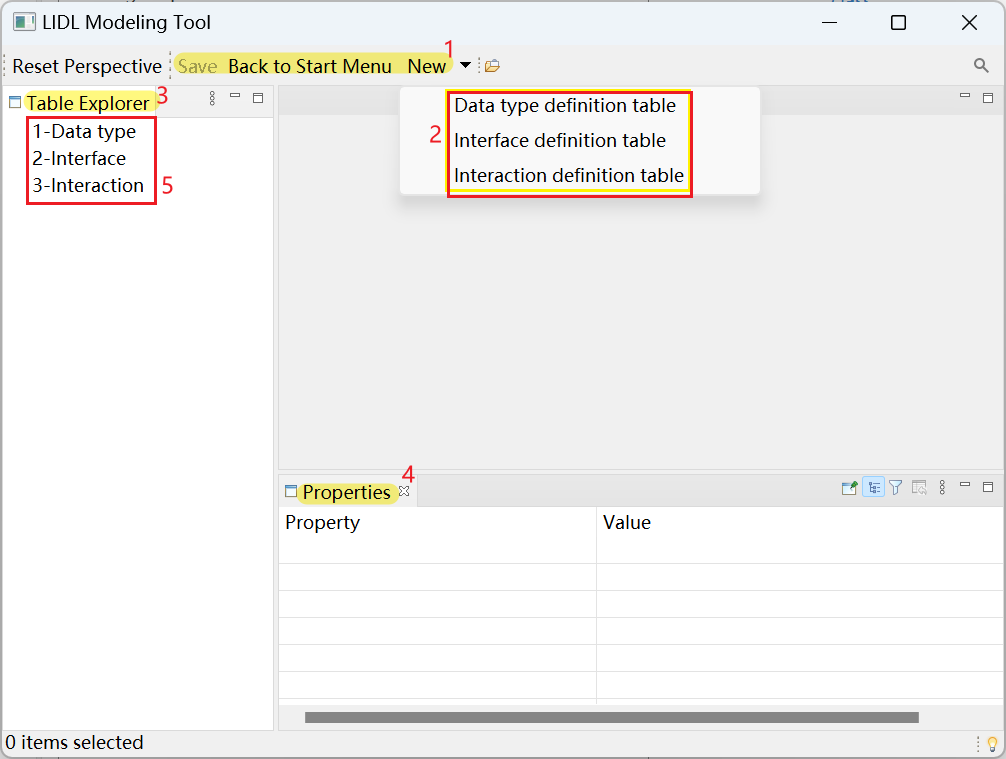
In the User interface design modeling view, users can draw the interaction architecture diagram according to the defined interactions.

## Definition table view

First, click LIDL model element definition table view.



Now we entered the Definition table view shown in the figure below to define data types, interfaces, and interactions.



In area 1 in the above figure, you can return to the main page and create a new table.

In area 2 in the above figure, you can select the types of tables to be created, including the data type definition table, interface definition table, and interaction definition table.

In area 3 in the above figure, you can get an overview of the table.

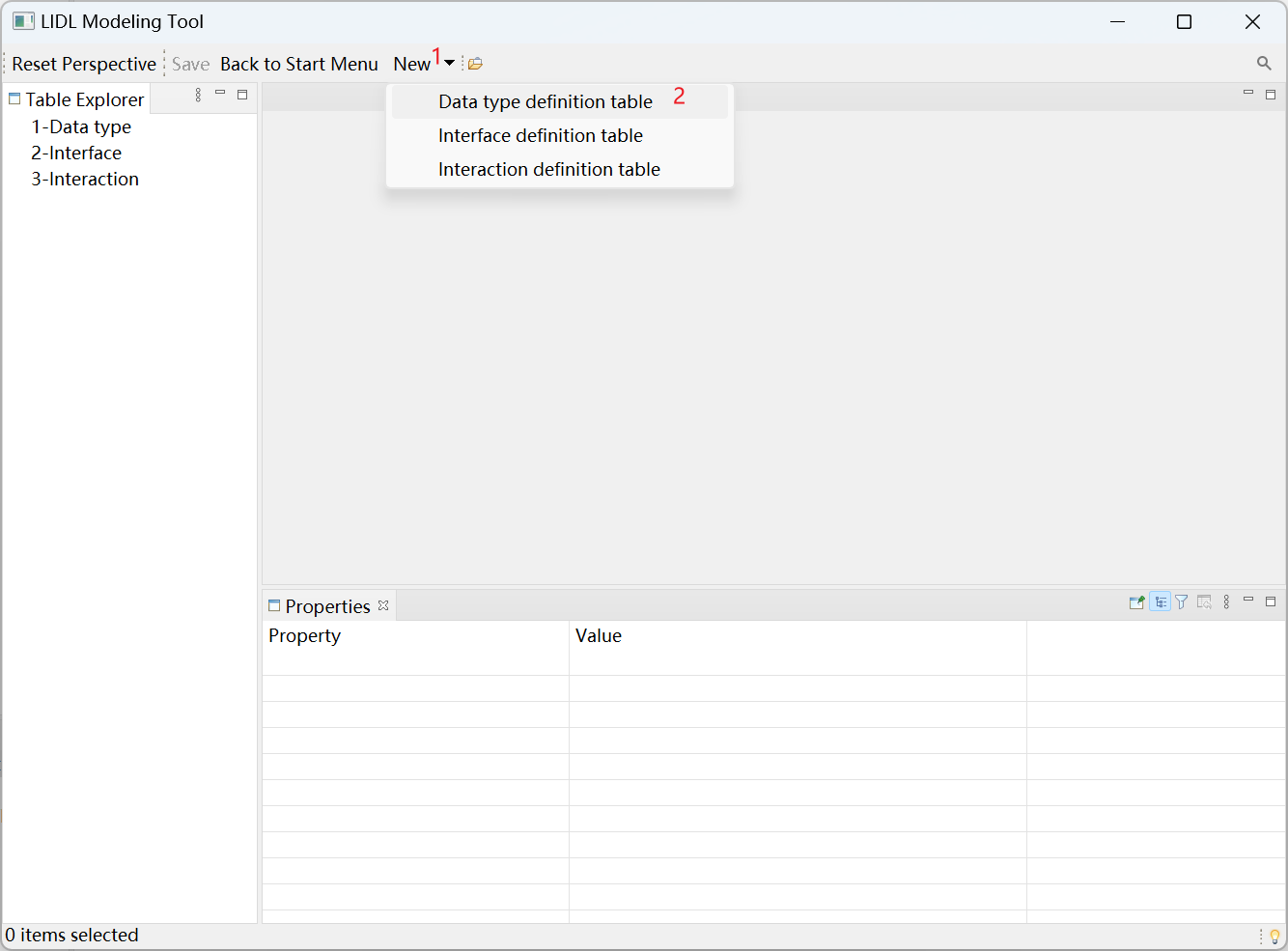
In area 4 in the figure above, you can see the configuration of data type, interface, and interaction.

In area 5 in the figure above, you can see the summary of three types of tables.

## Definition tables

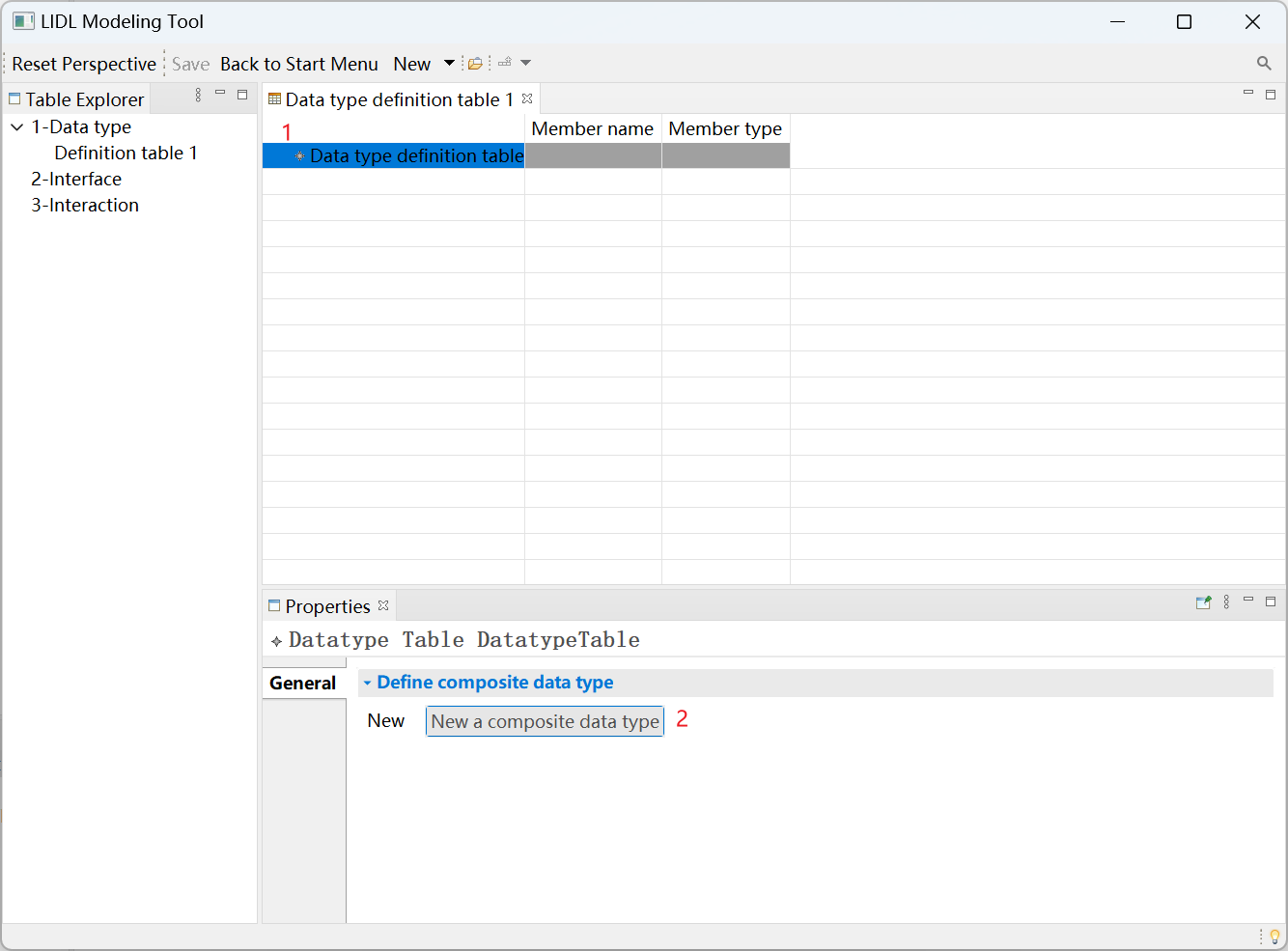
### Data type definition table

After entering the Definition table view, click New, and then click Data type definition table.



Automatically create Data type definition table 1.

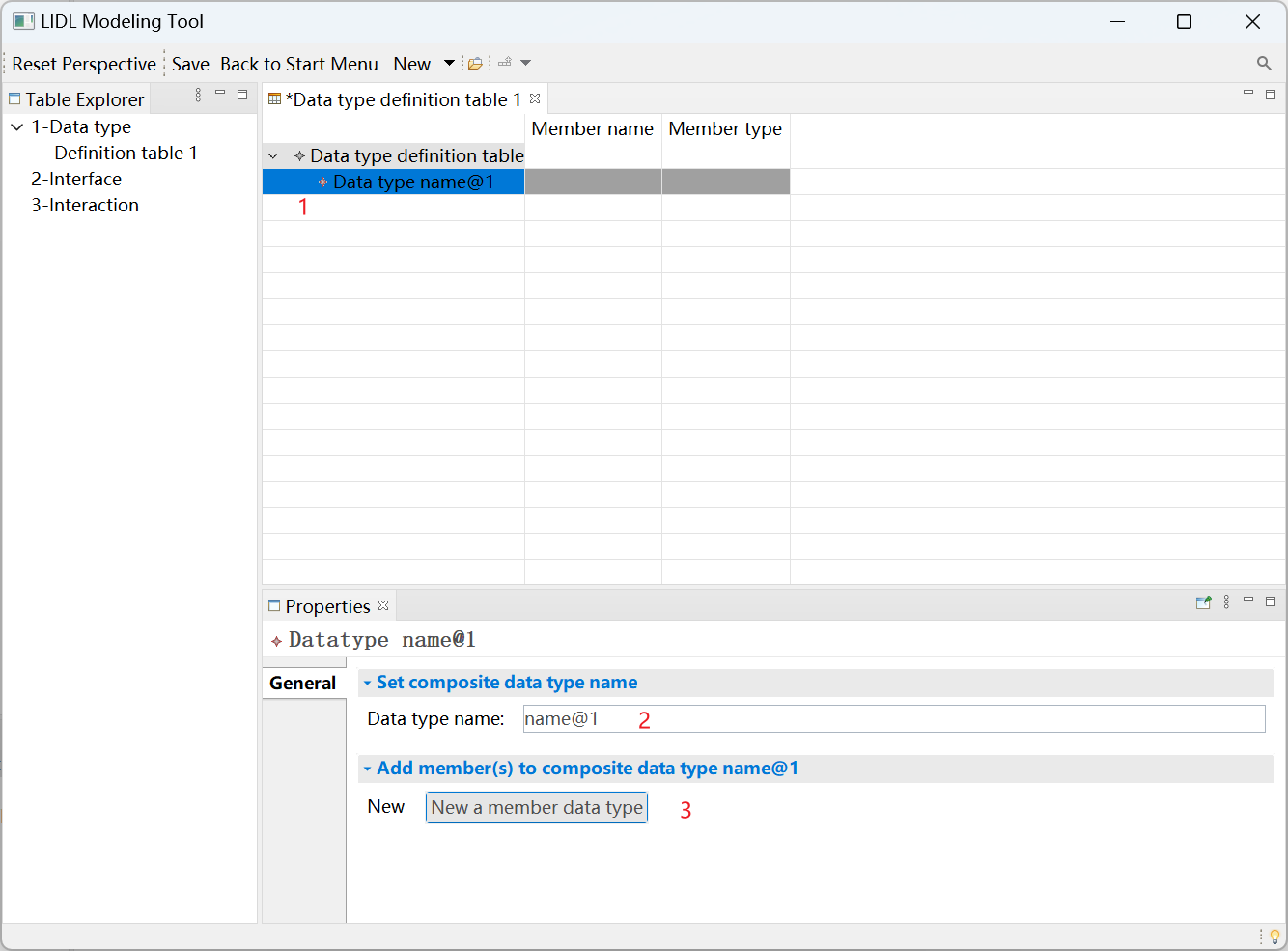
Click the Data type definition table cell, and click New a composite data type in the Properties view below.



Click the newly created Composite Data type: data type name@1.

In the Properties view below, edit the newly created data type name and name it Point2D

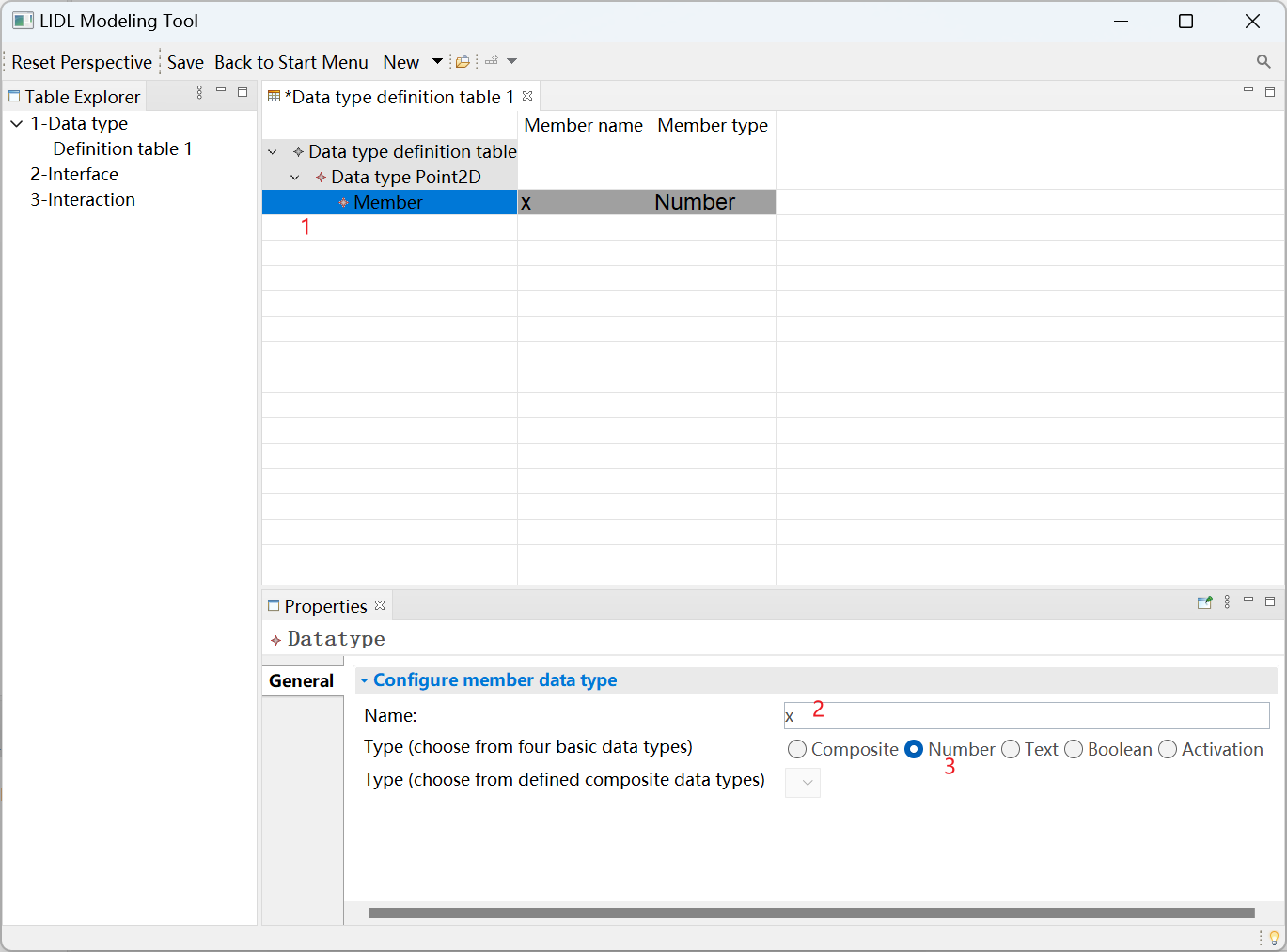
Click New a member datatype to add members to the newly created composite data type.



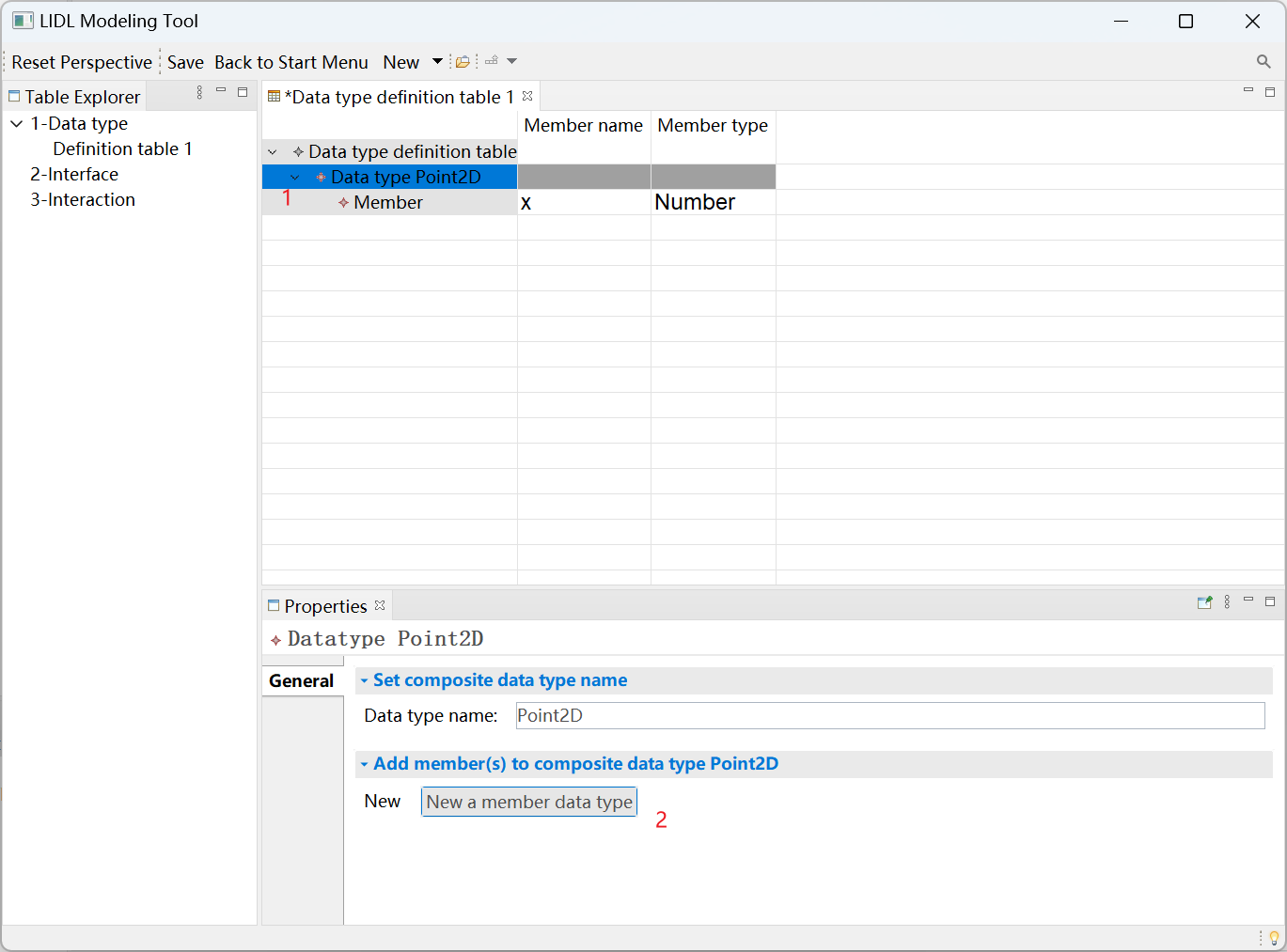
Click the new member.

Edit the member’s name as x in the Properties view below.

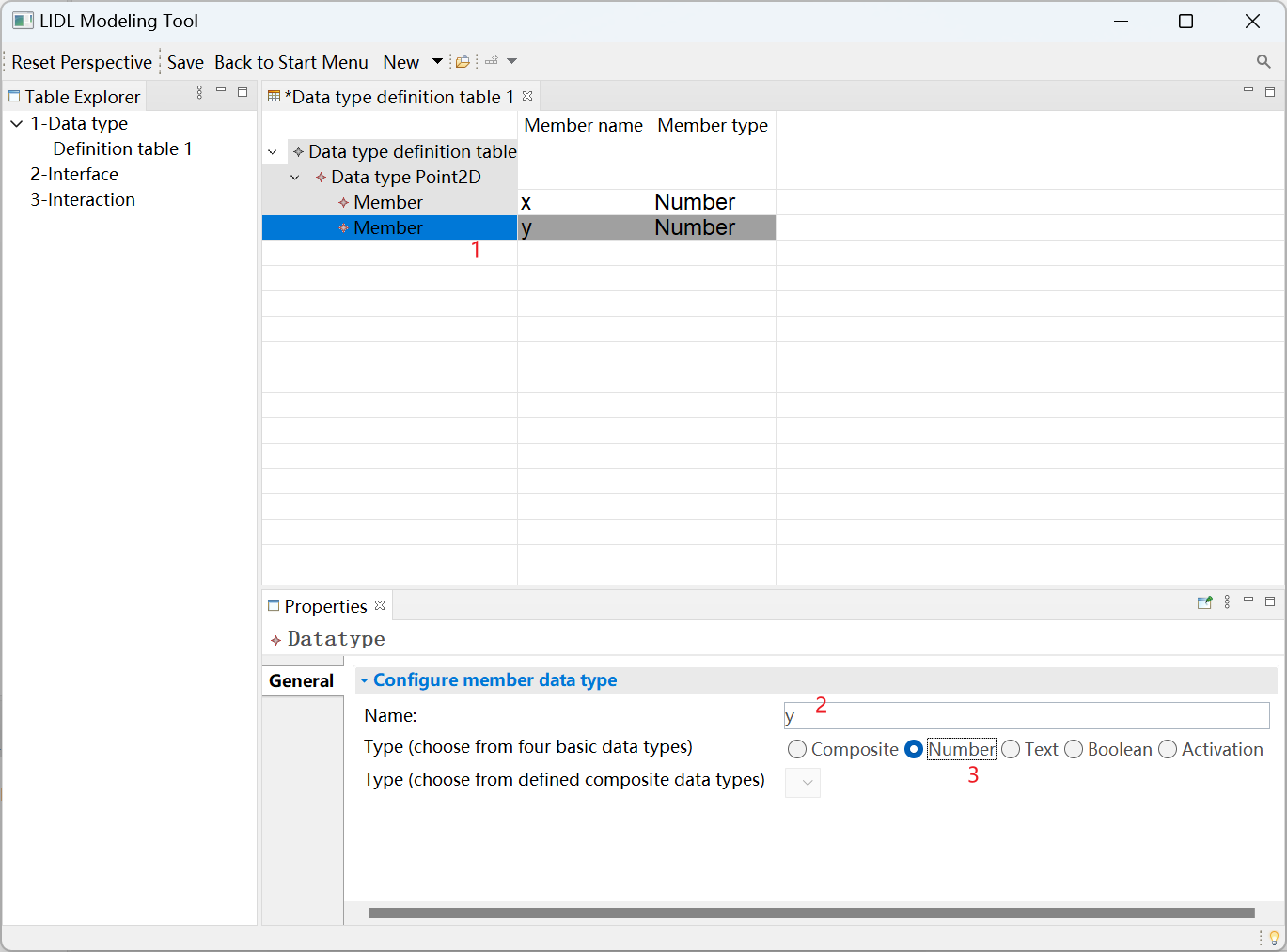
Select member type as Number.



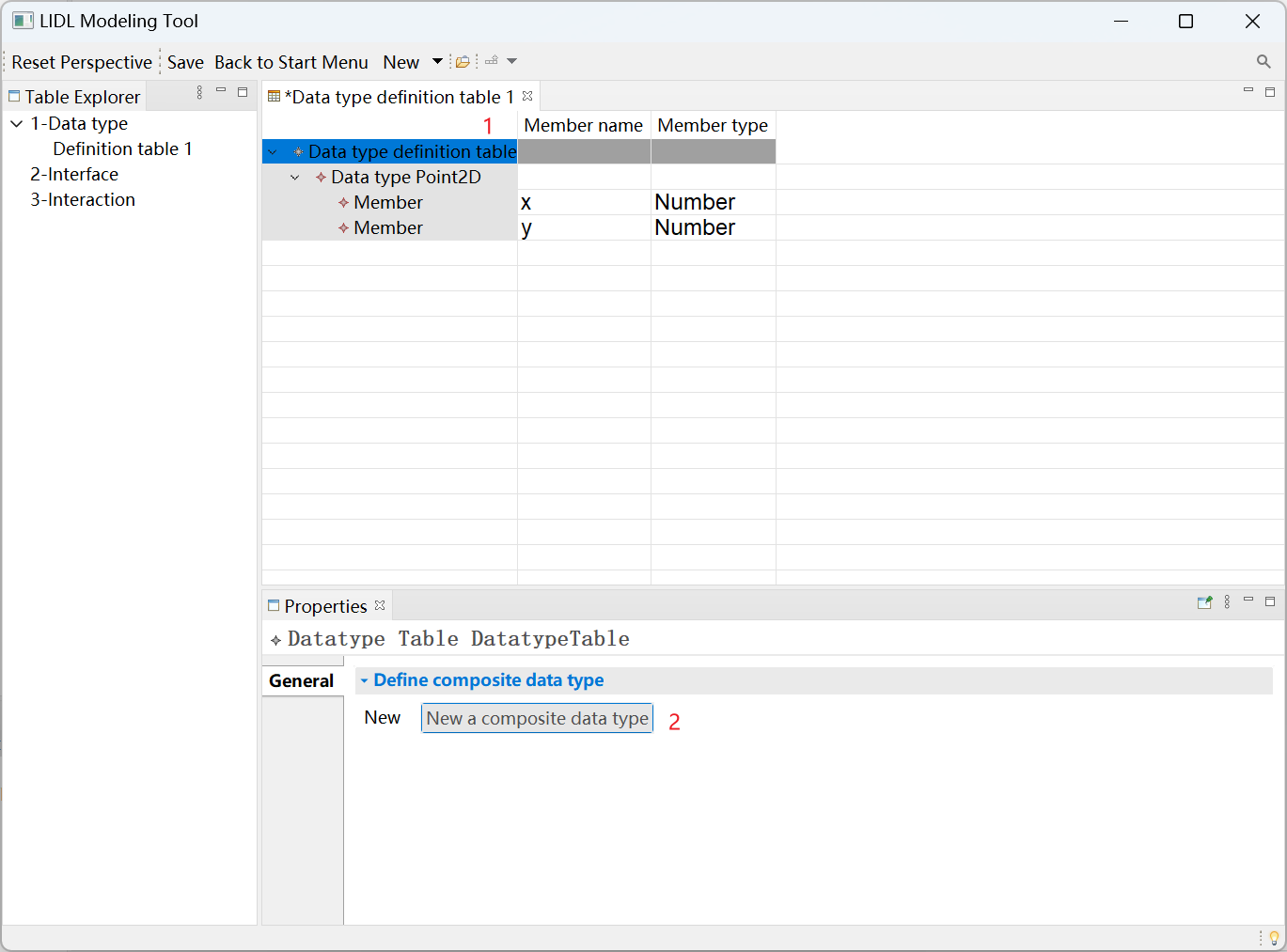
Click Data type Point2D again, and then click New a member data type to continue adding members.



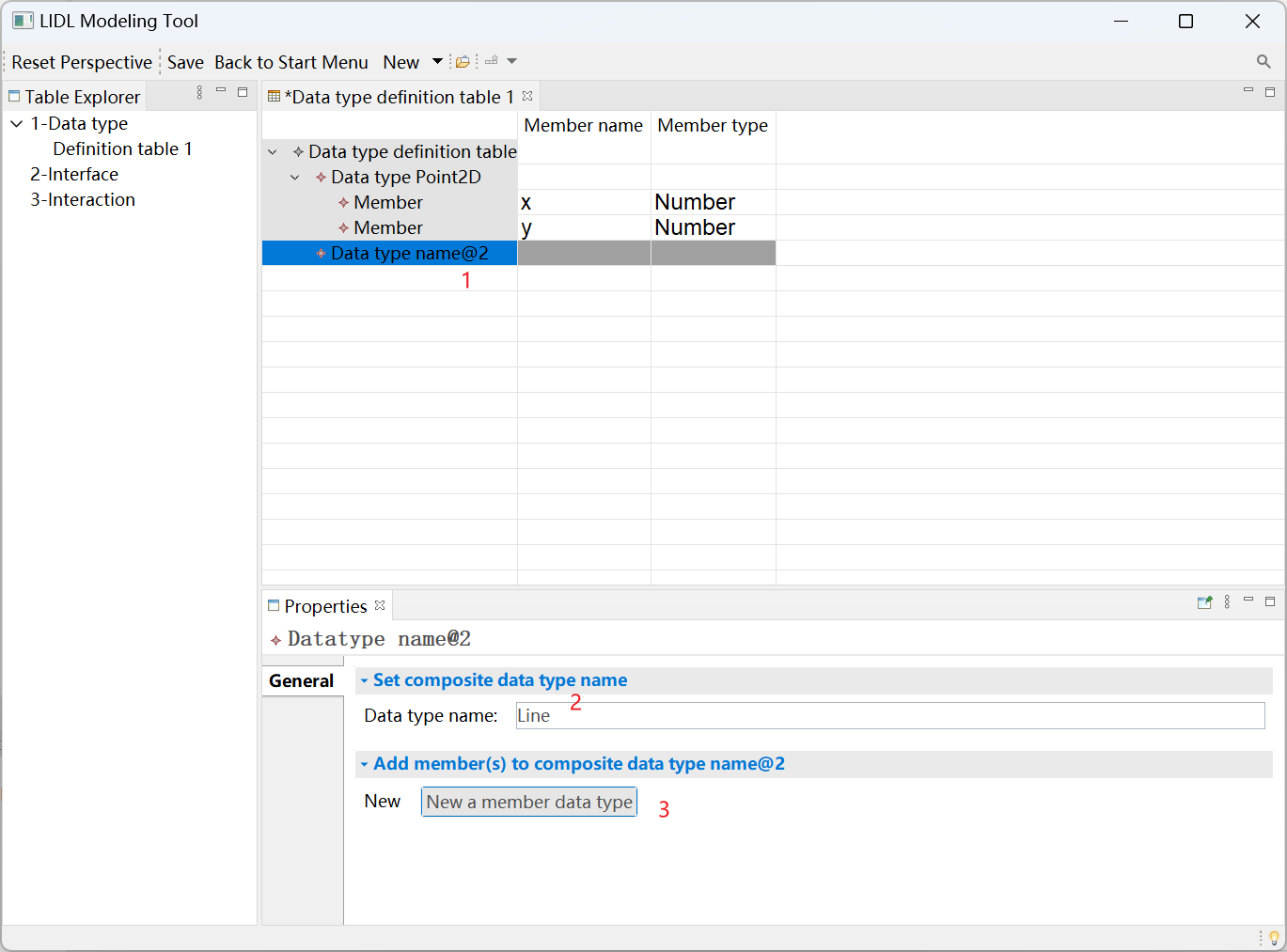
Name the newly added member y, and select the type as Number.

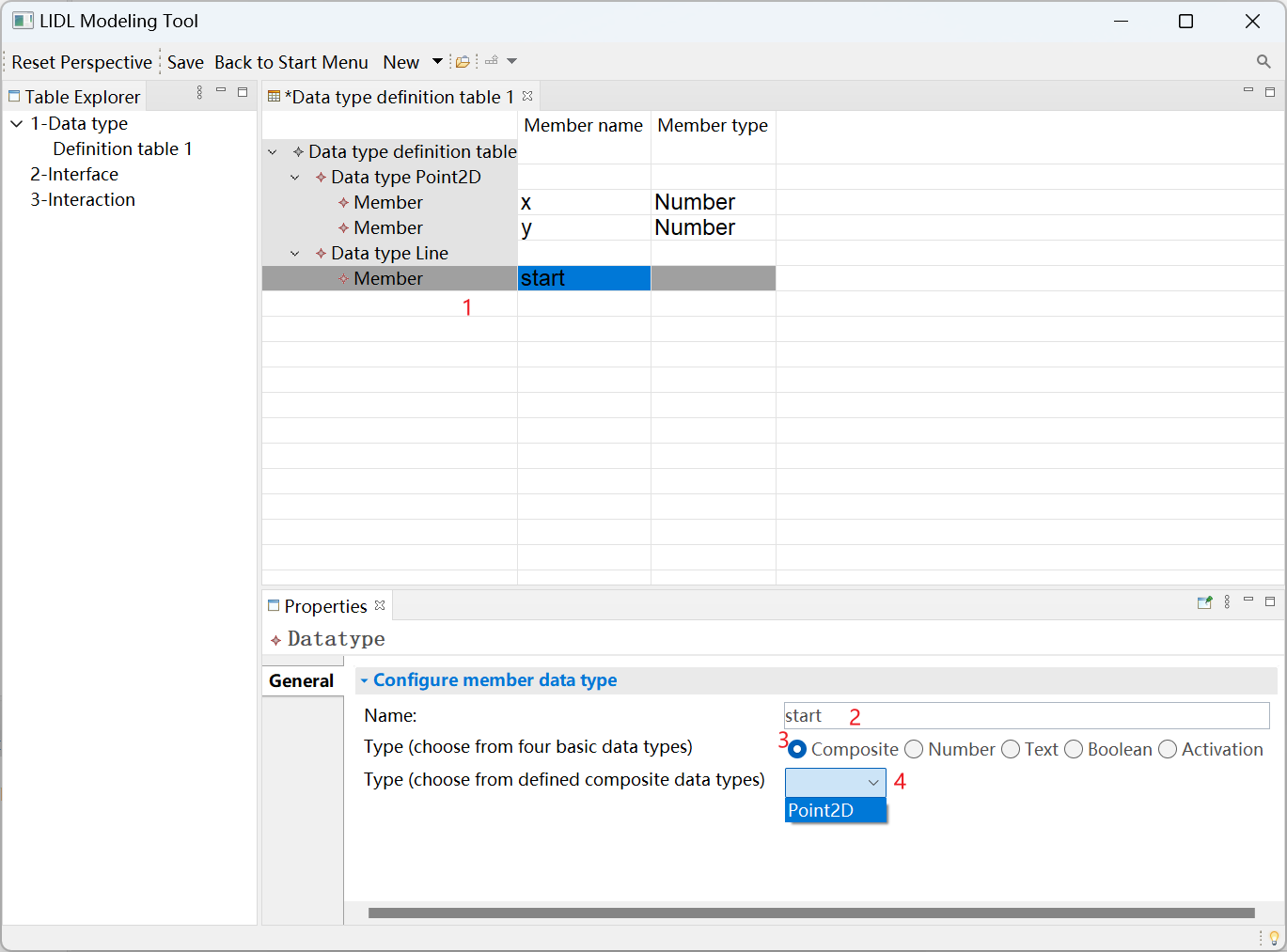


Continue to click the Data type definition table, and click New a composite data type to add a new composite data type.

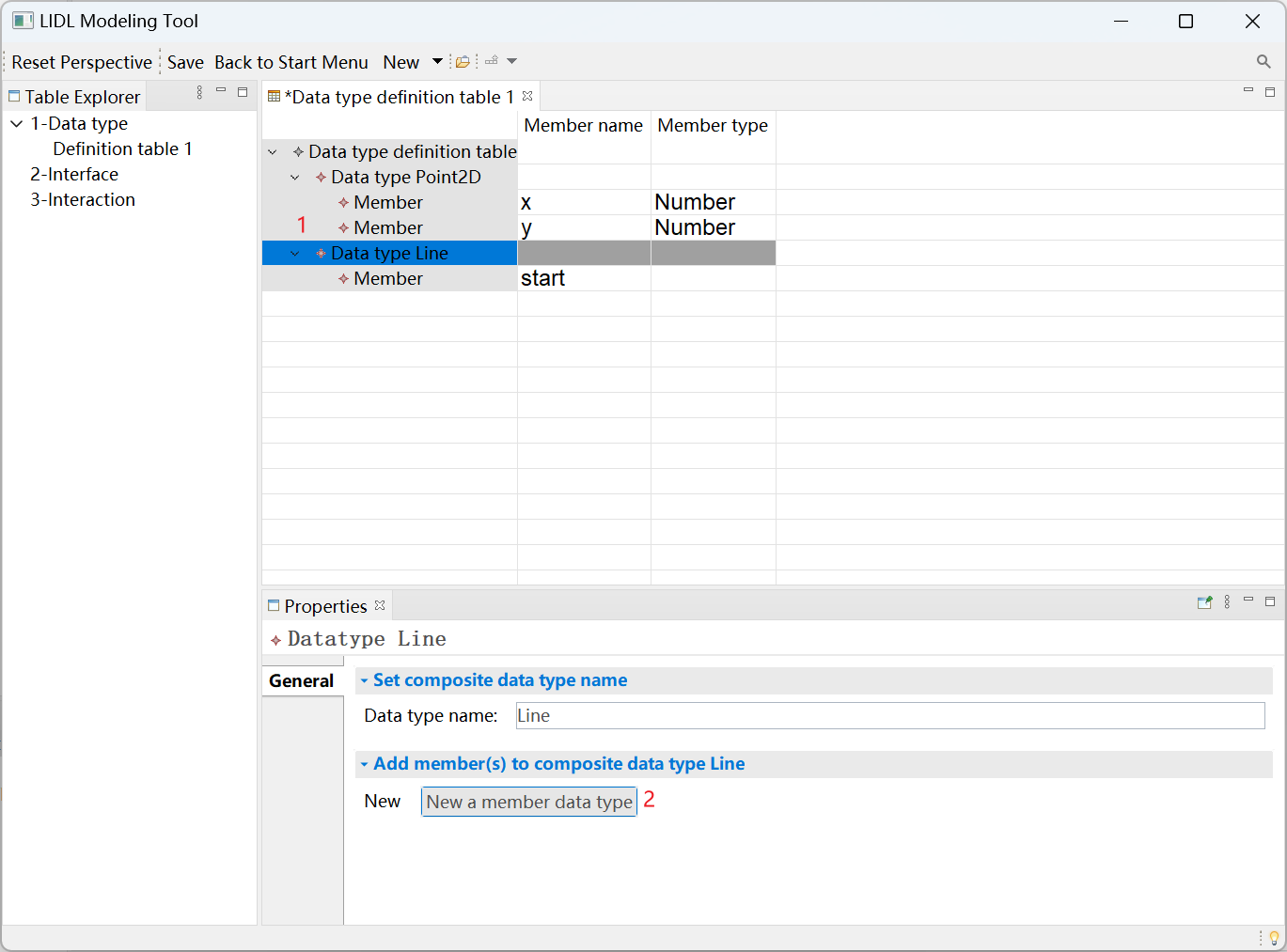


Click the newly created data type name@2, name it Line, and click New a member data type to add a member.

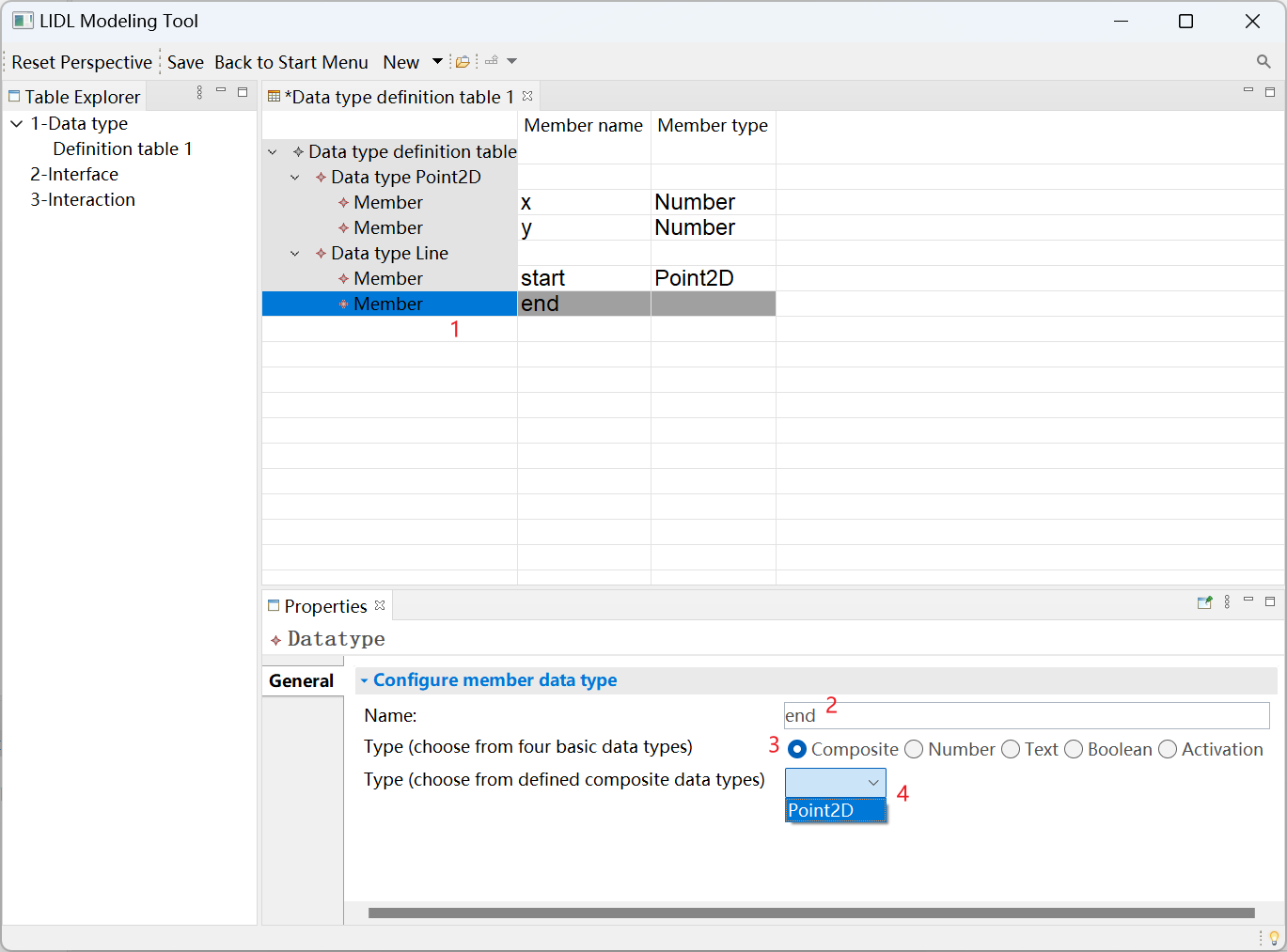


Name the member start, select Composite as the type, and then select Point2D just defined.

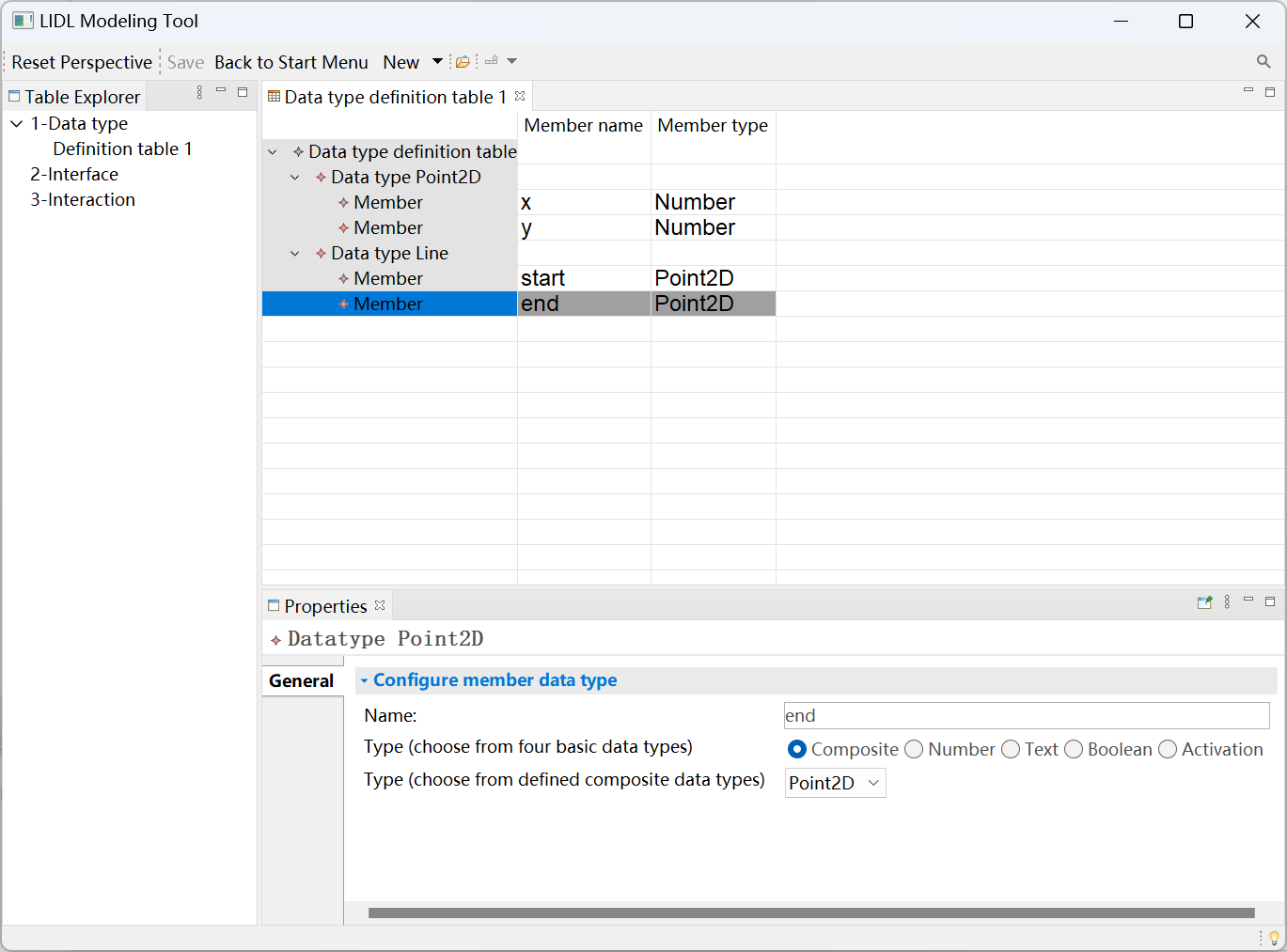
Continue to add member data type for Data type Line.



Add a Member with the name end and the type Point2D.

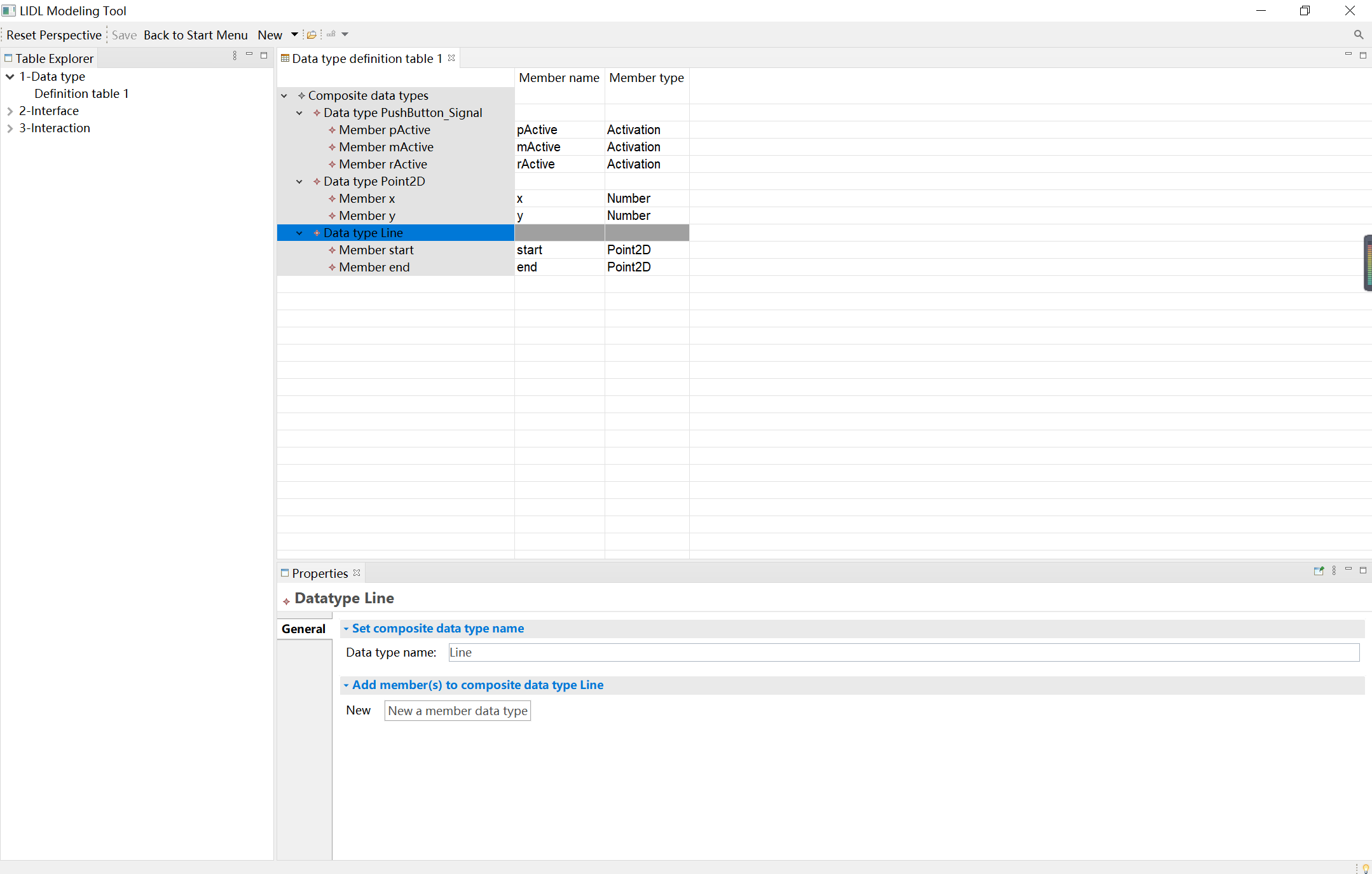


Finally, press Ctrl and s to save the Data type definition table 1.



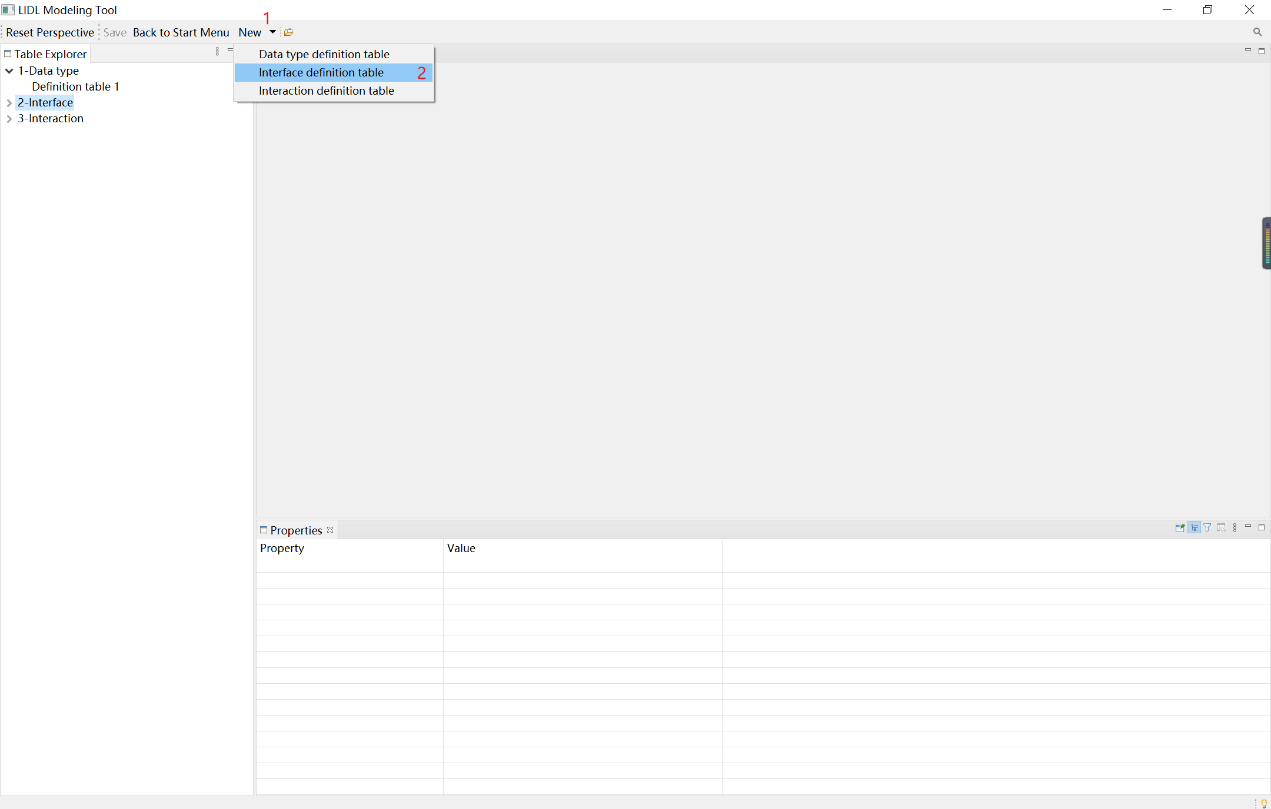
The above modeling process demonstrates the definition process of Point2D and Line data types in detail, and you can define PushButton\_Signal in the same way.

You can see the final data type definition table in the following figure.



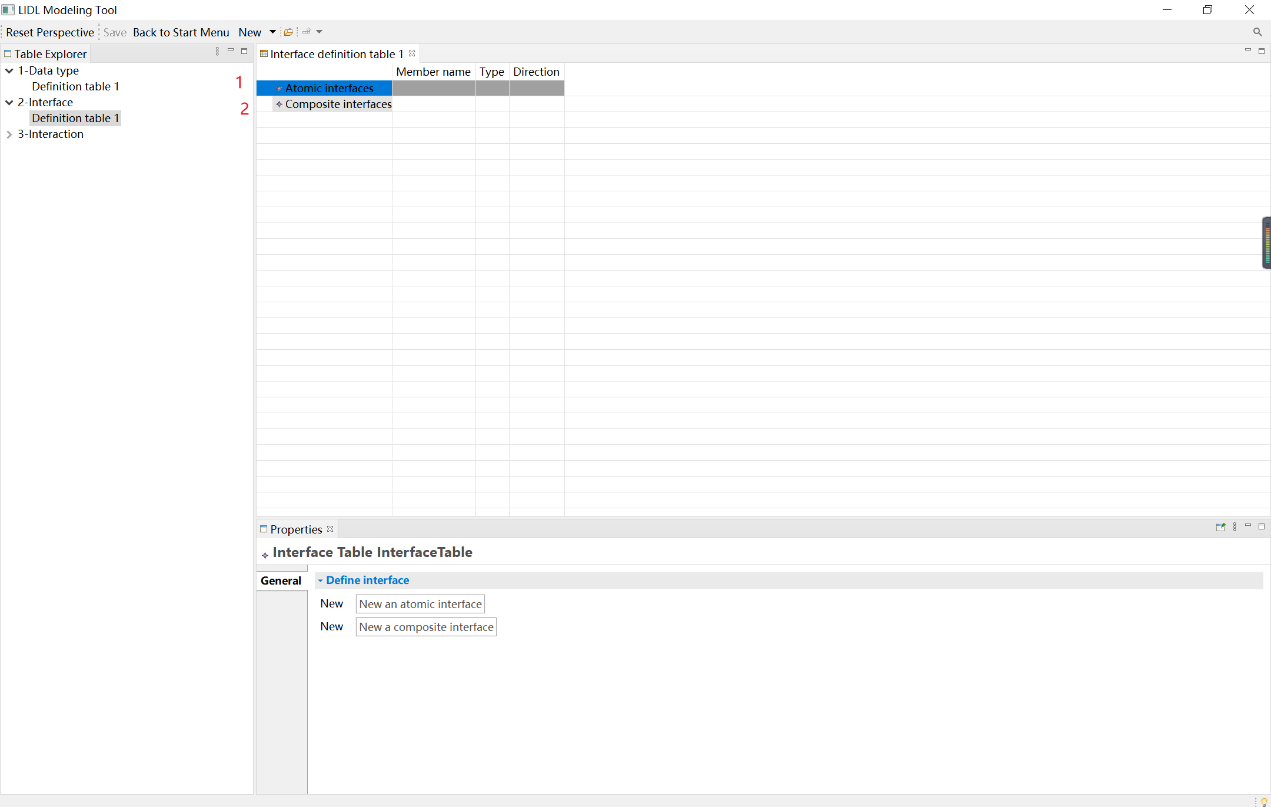
### Interface definition table

In the Definition table view, click New, and then click Interface definition table.



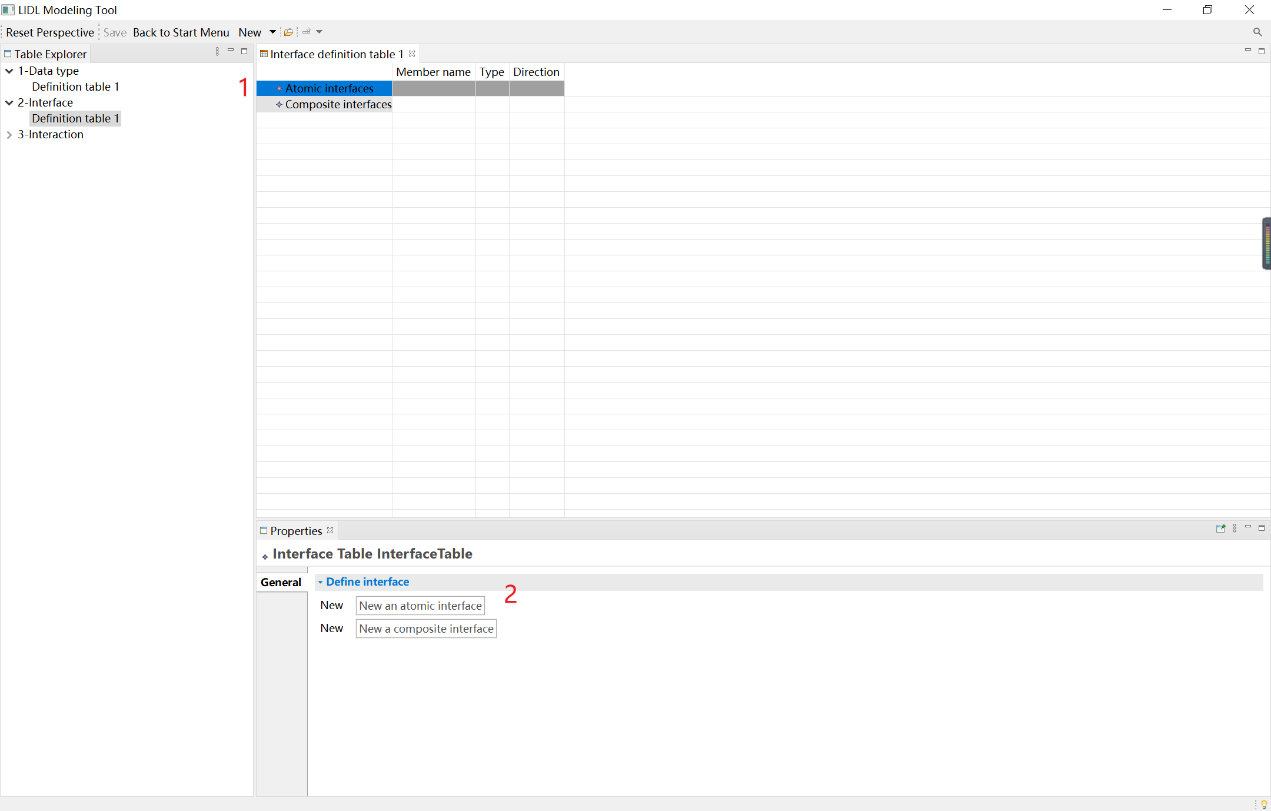
Automatically create interface definition table 1.

It includes two parts (atomic interface and composite interface).



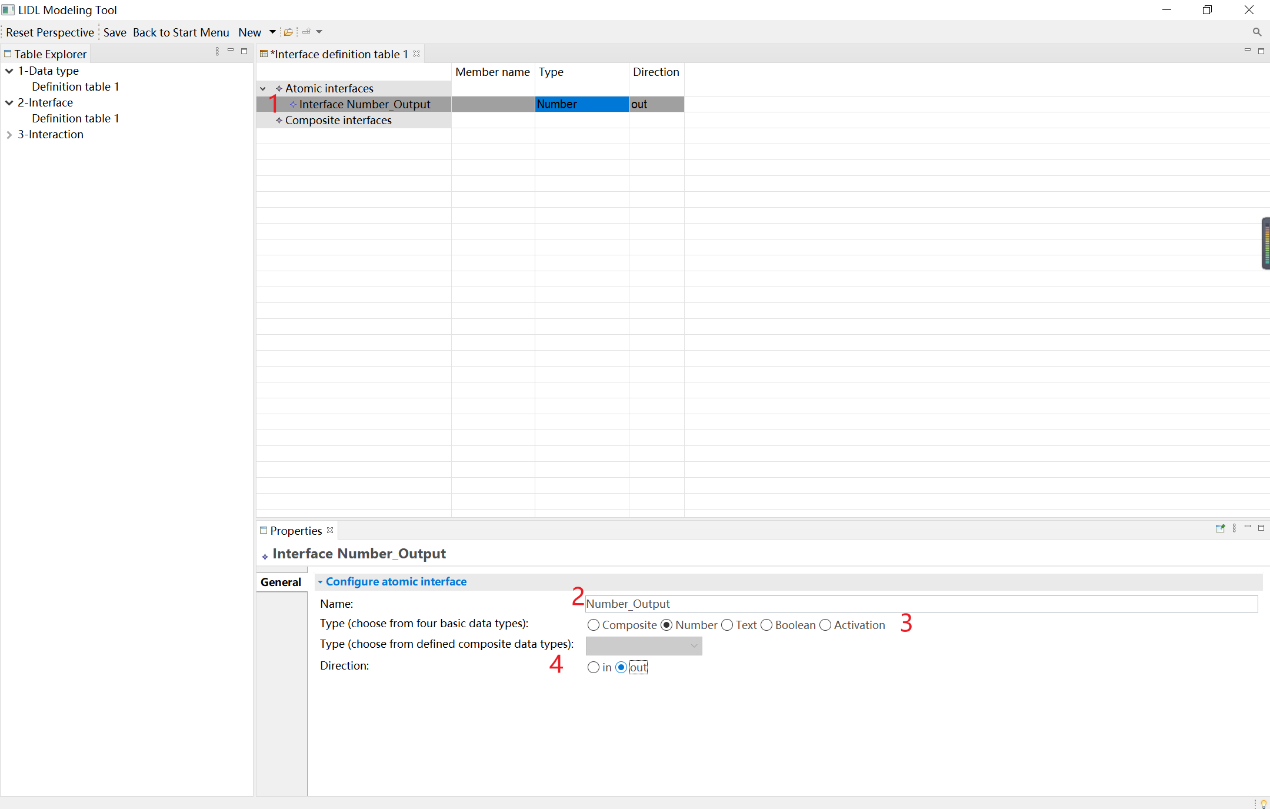
First, create the atomic interface.

Click the Atomic Interfaces cell, and click New an atomic interface in the Properties view below.



Click the newly created Atomic Interface: Interface MyInterface@1.

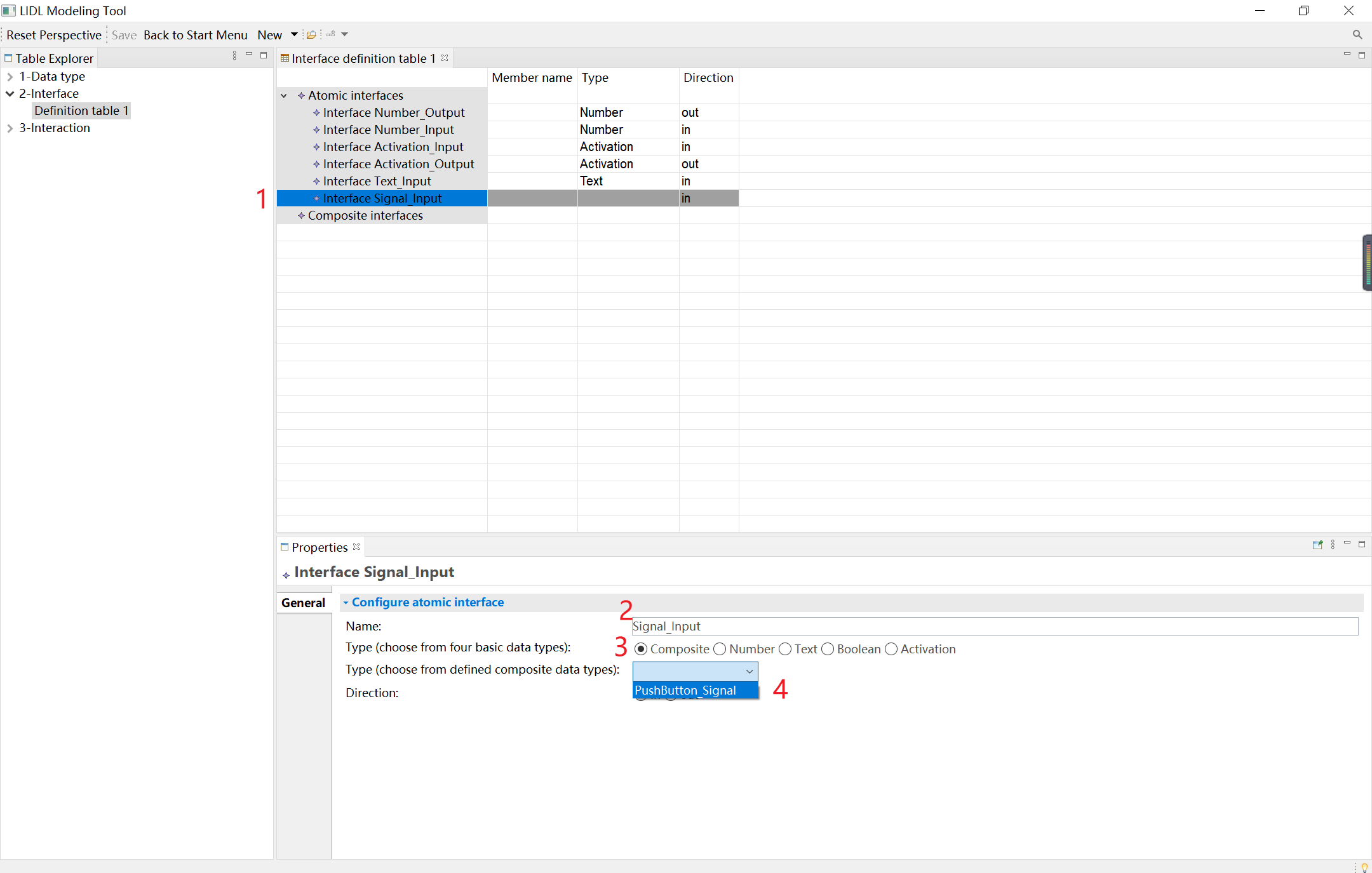
In the Properties view below, edit the newly created interface MyInterface@1, name it Number\_Output, and configure its Type as Number and Direction as out.



When configuring the data type of the interface, you can select not only the four basic LIDL data types but also the self-defined data types in the Data type definition table.

Click the newly created Atomic Interface: Interface MyInterface@2.

In the Properties view below, edit the newly created interface MyInterface@2, name it Signal\_ Input, select Composite as the Type (choose from four basic data types), and select PushButton\_Signal from the drop-down menu of Type (choose from defined composite data types), configure its Direction as in.



Then create several atomic interfaces in the same way:

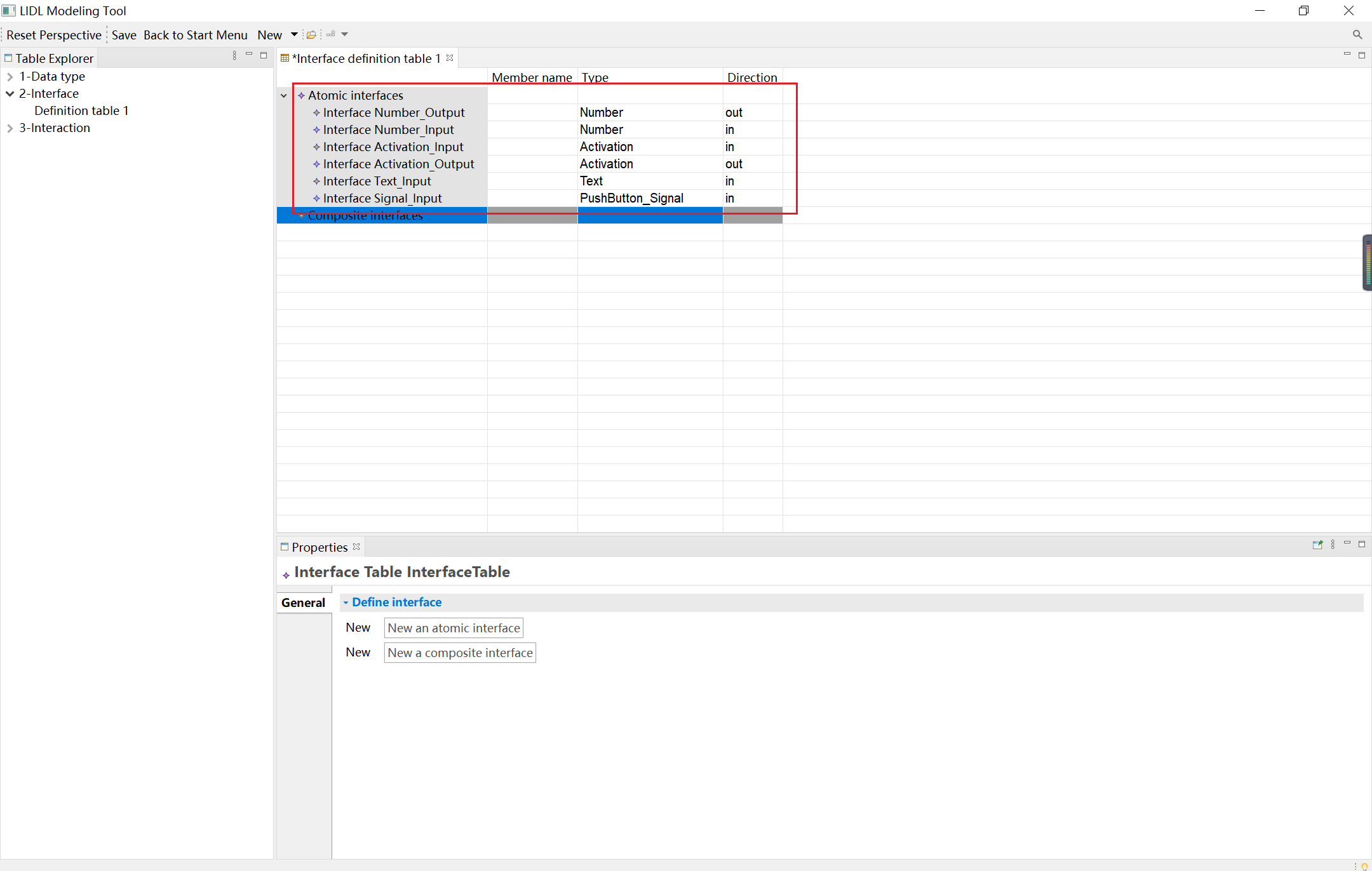
Number\_ Input,

Activation\_ Output,

Activation\_ Input,

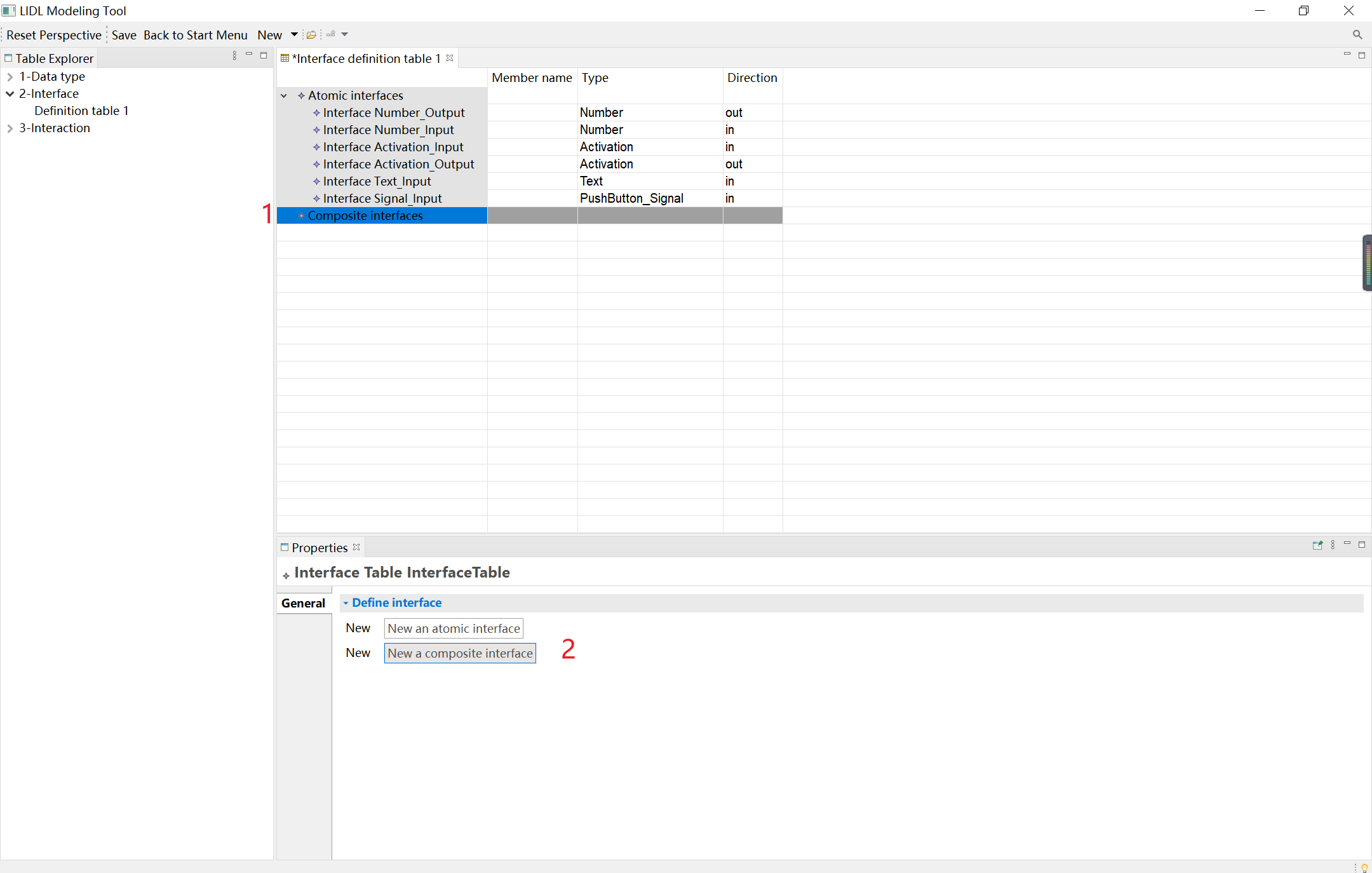
Text\_ Input。

You can see the final atomic interfaces table in the following figure:



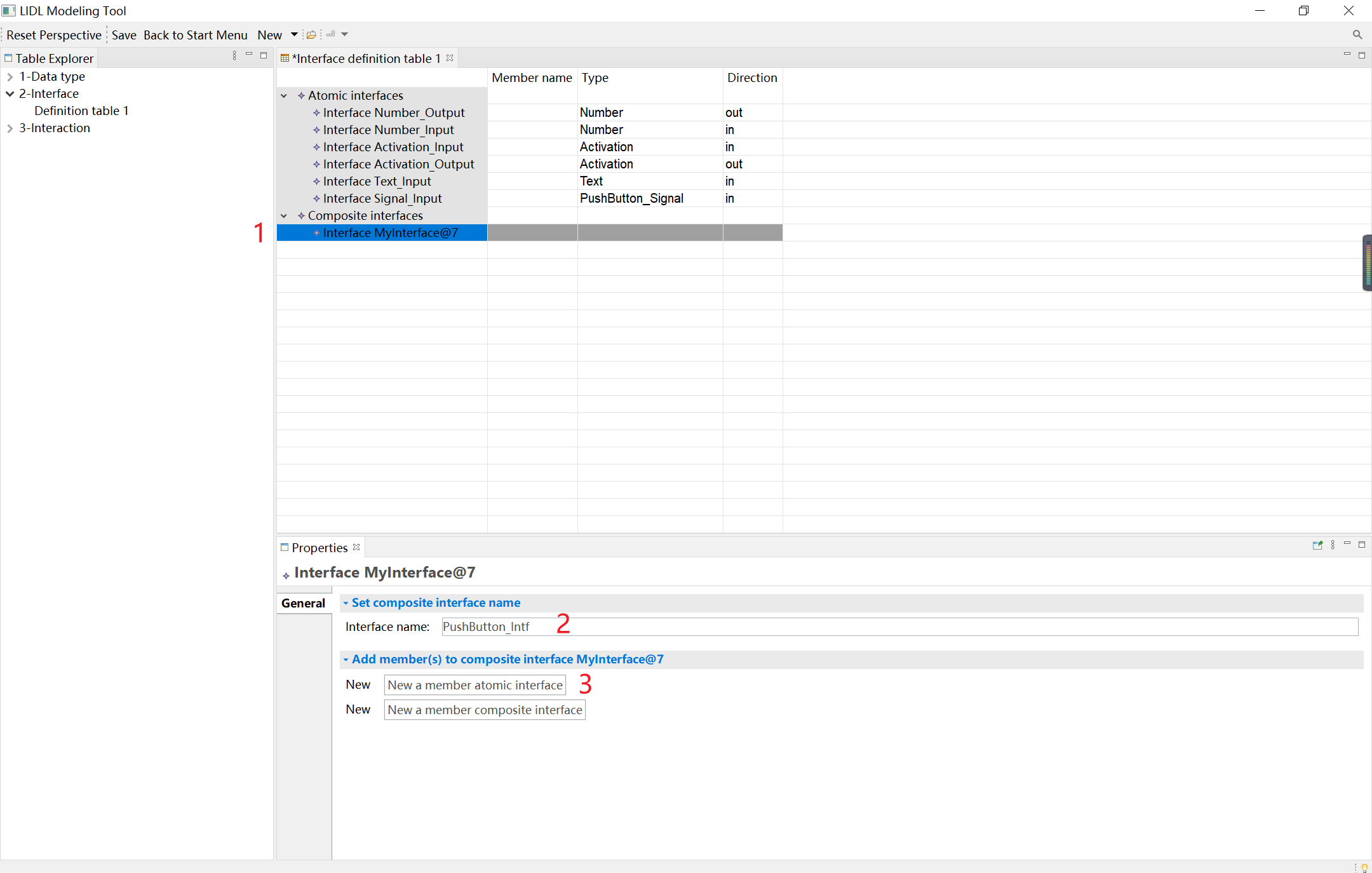
Next, create the composite interface.

Click the Composite Interfaces cell, and click New a composite interface in the Properties view below.



Click the newly created Composite Interface: Interface MyInterface.

In the Properties view below, edit the newly created interface MyInterface, name it PushButton\_Intf and click New a member atomic interface below to create a member atomic interface for the composite interface.



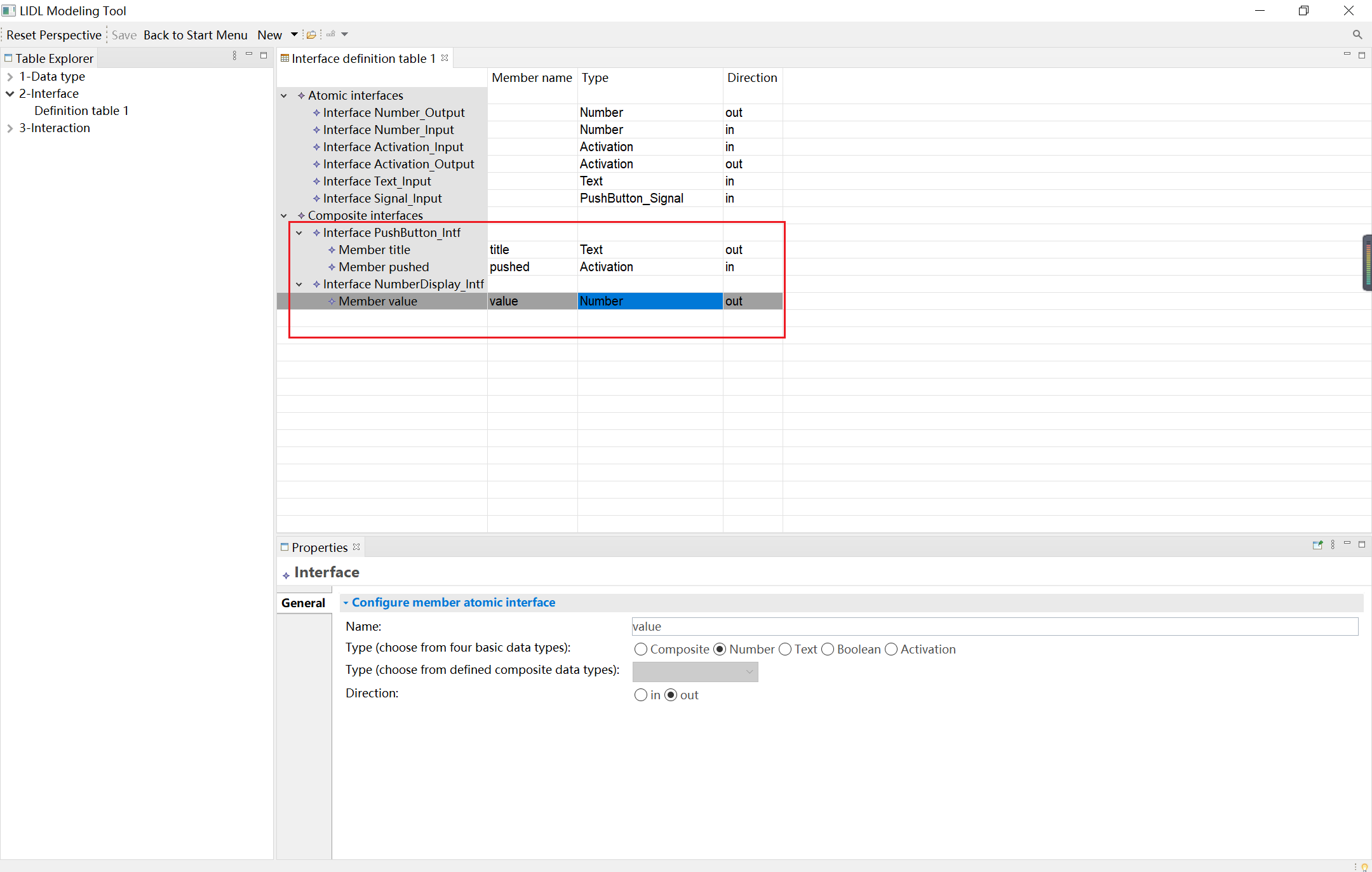
Click the newly created member atomic interface.

In the Properties view below, edit the newly created interface. Name the interface title, set the type to text, and set the direction to out.

You can create a member atomic interface (PushButton\_Intf) in the same way, and name it pushed, set type to Activation, and set direction to in.

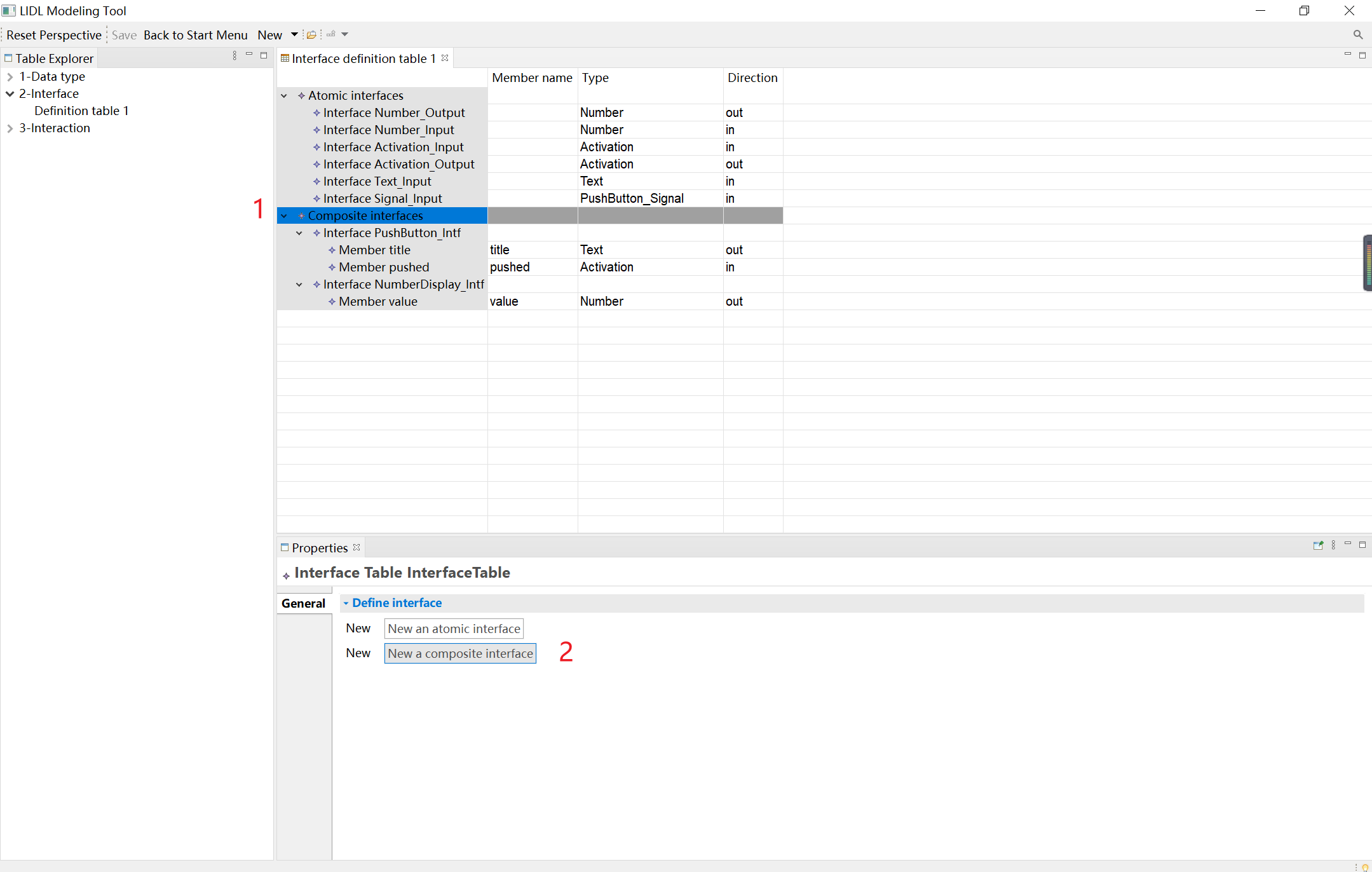
Create a new composite interface named NumberDisplay\_Intf, and then create a member atomic interface for NumberDisplay\_Intf. For the interface, name it value, set type to Number, and set direction as out.

You can see the final effect in the following figure.



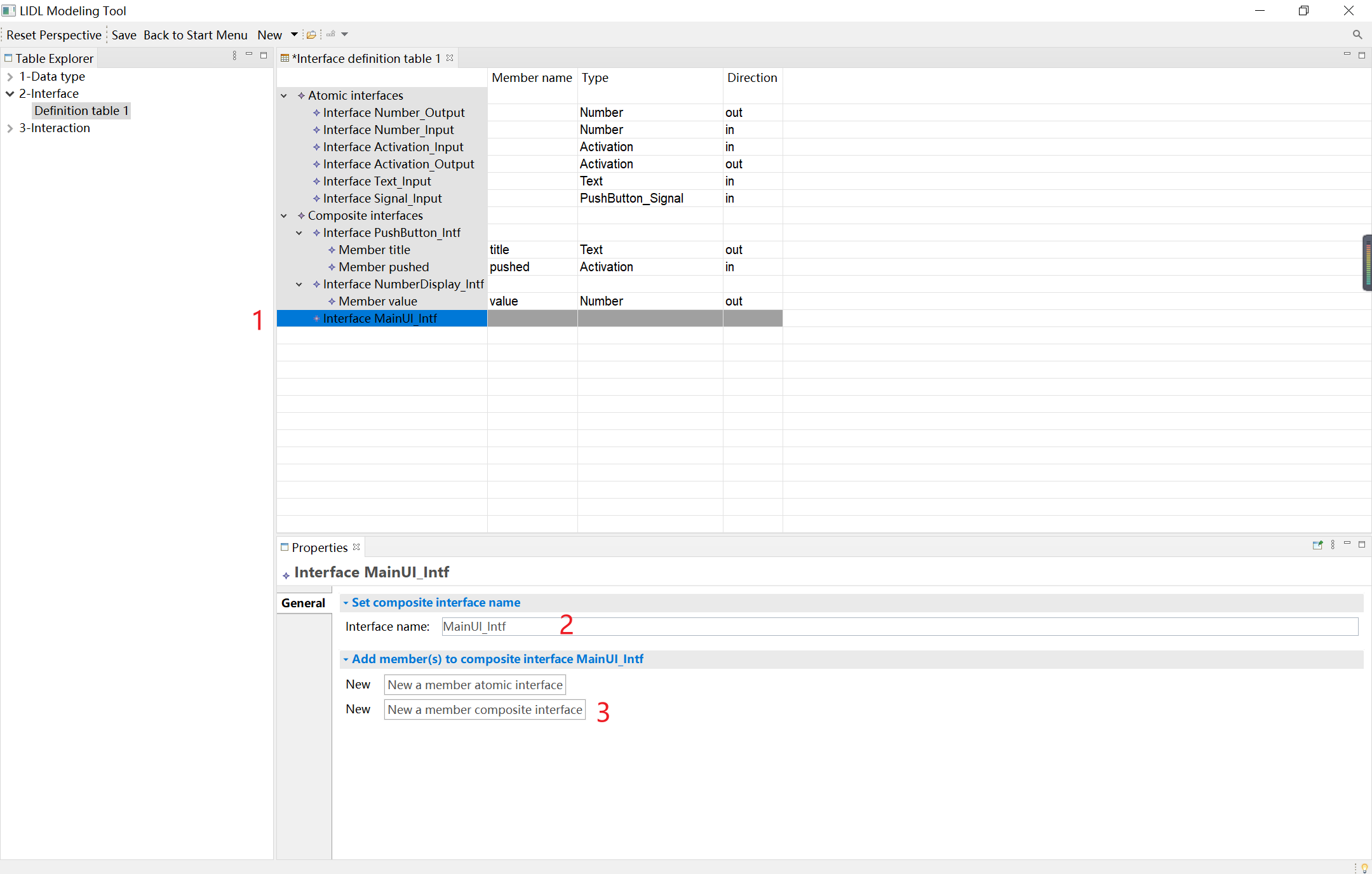
The following shows how to create a member composite interface in a new composite interface.

Click the Composite Interfaces cell, and click New a composite interface in the Properties view below.



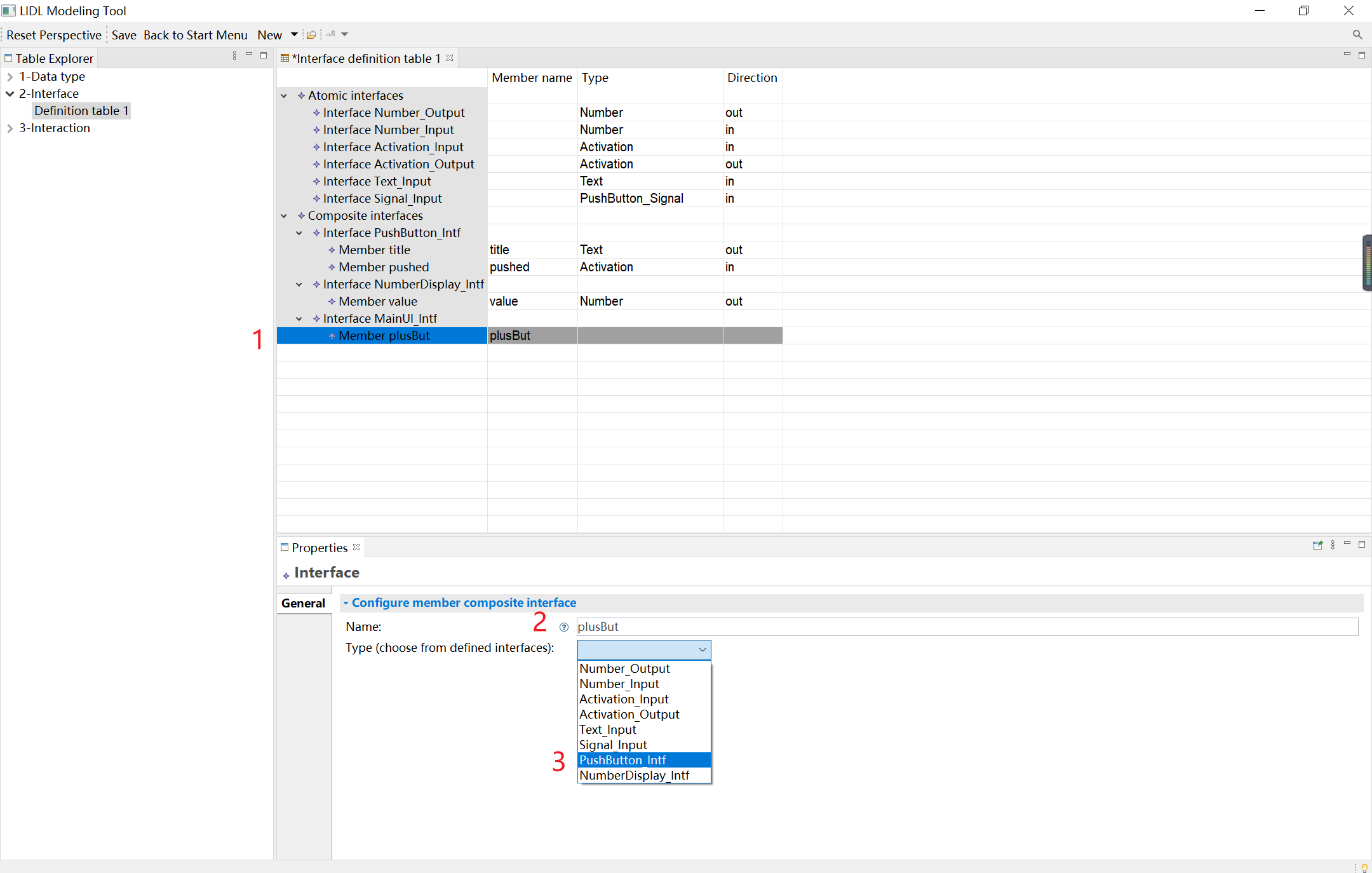
Click the newly created Composite Interface: Interface MyInterface@9.

In the Properties view below, edit the newly created interface MyInterface@9, name it MainUI\_Intf, and click New a member composite interface below to create a member composite interface for the composite interface.



Click the newly created Member Composite Interface: Member label@1.

In the Properties view below, name it plusBut, and select the configured composite interface PushButton\_Intf from the drop-down menu.



In the same way, for MainUI\_Intf you can configure the next three member composite interfaces.

minusBut: PushButton\_Intf

resetBut: PushButton\_Intf

display: NumberDisplay\_Intf

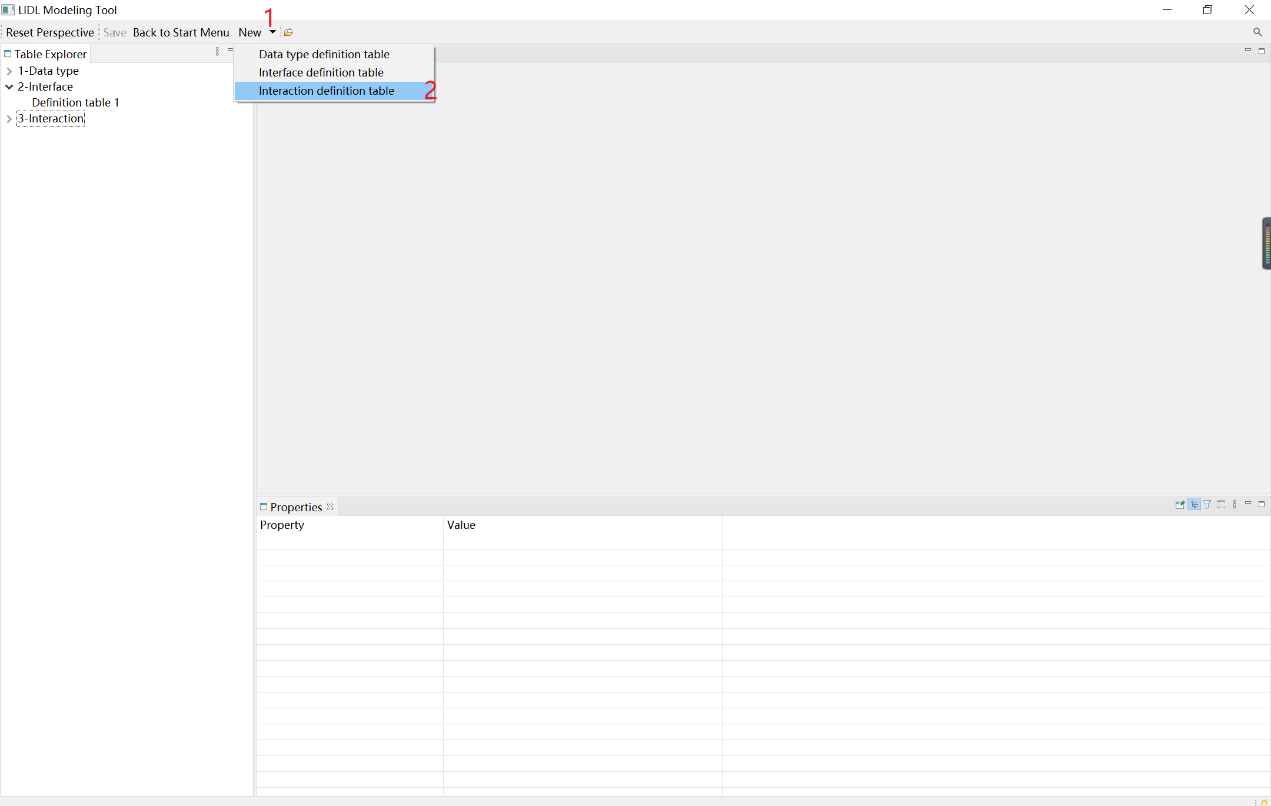
You can see the final effect in the following figure:



### Interaction definition table

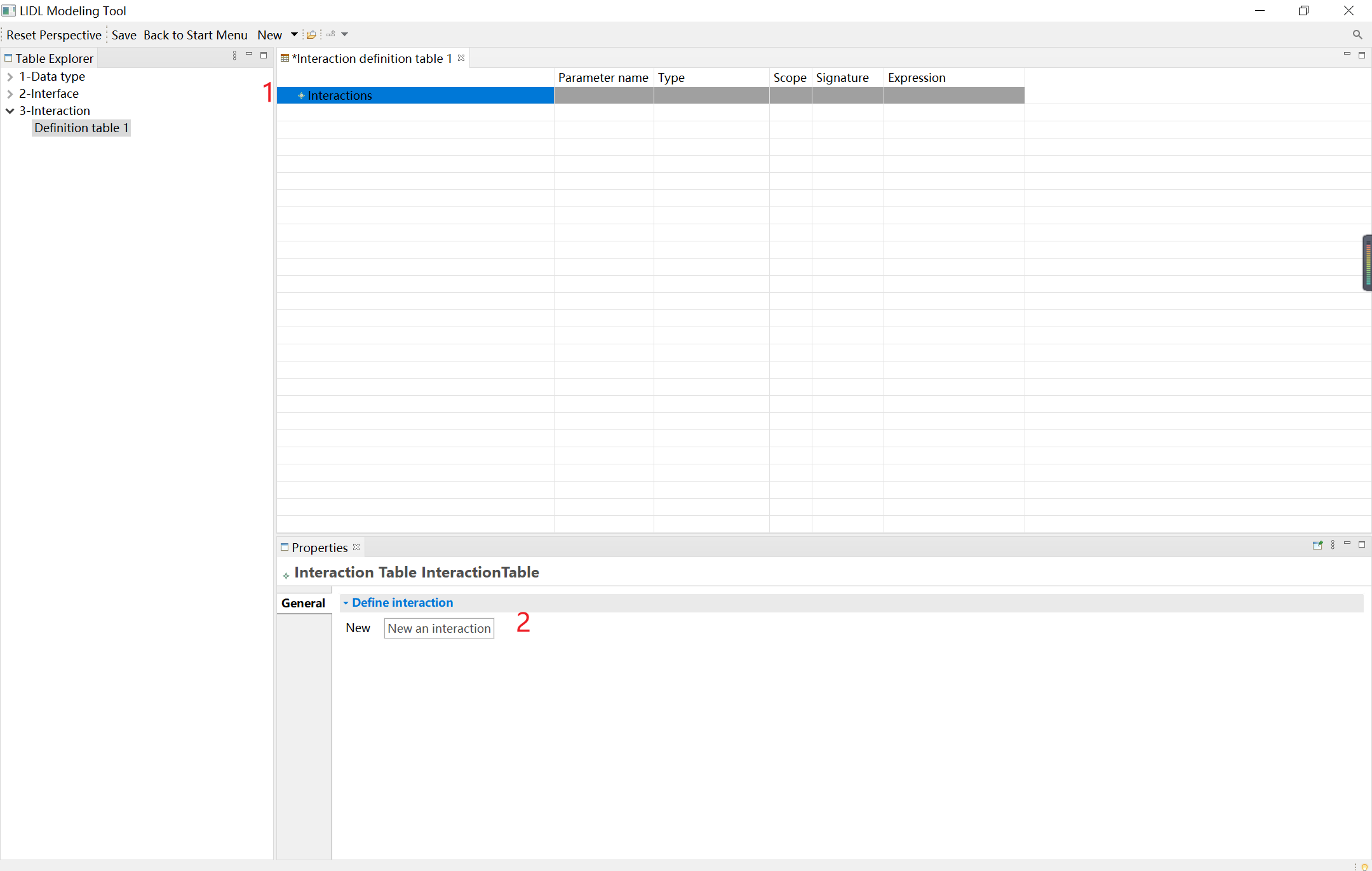
Next, we will demonstrate the definition process of the interactive definition table.

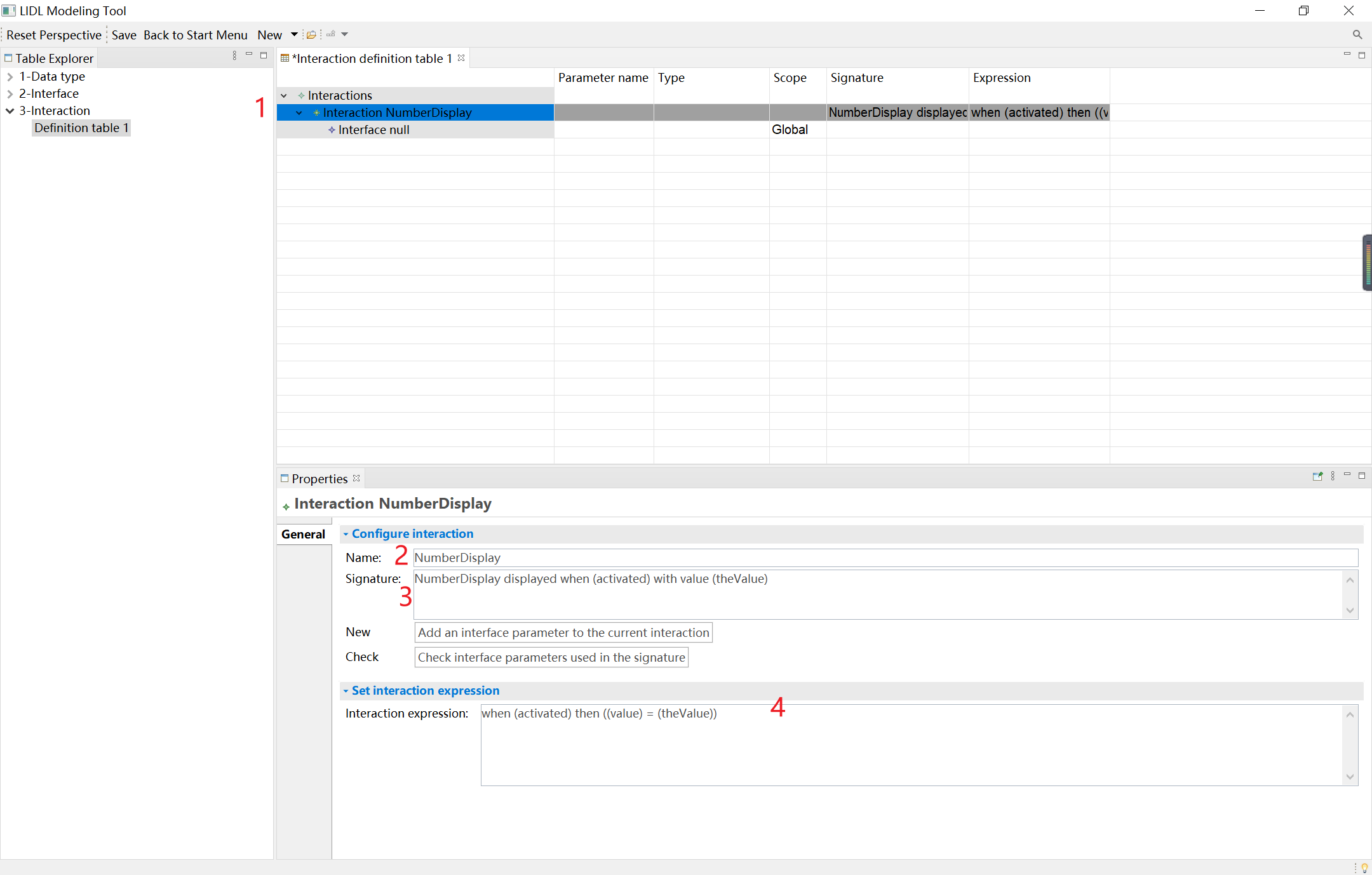
In the Definition table view, click New, and then click Interaction definition table.



Automatically create the Interaction definition table 1.

Click the Interactions cell, and click the New an interaction button in the Properties view below to create a new interaction.





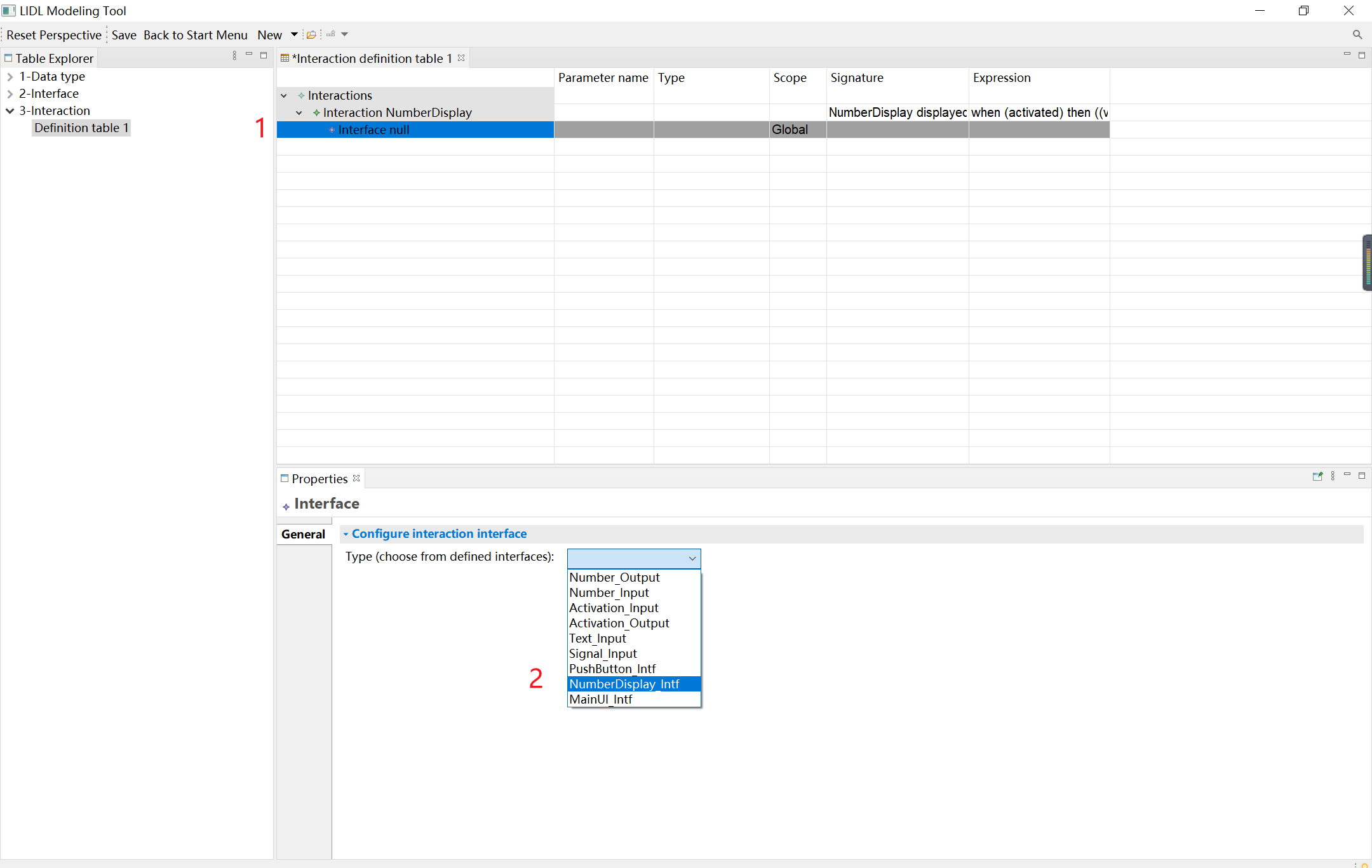
Click the newly created Interaction cell. You can configure the interaction information in the Properties view below and name it NumberDisplay,

Signature: NumberDisplay displayed when (activated) with value (theValue)

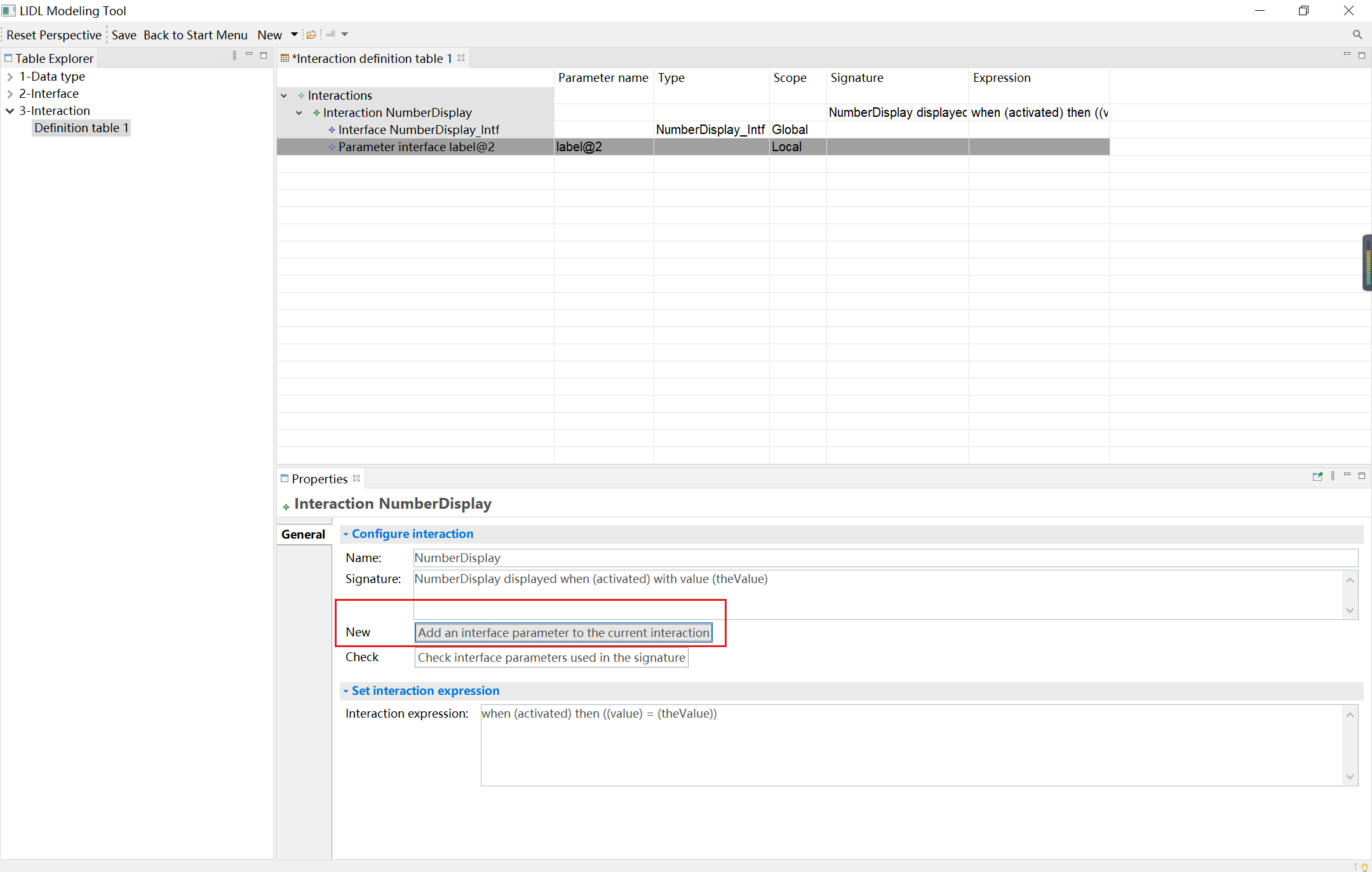
Expression: when (activated) then ((value)=(theValue))

Each interaction created naturally contains a Global interface, and you can configure its information.

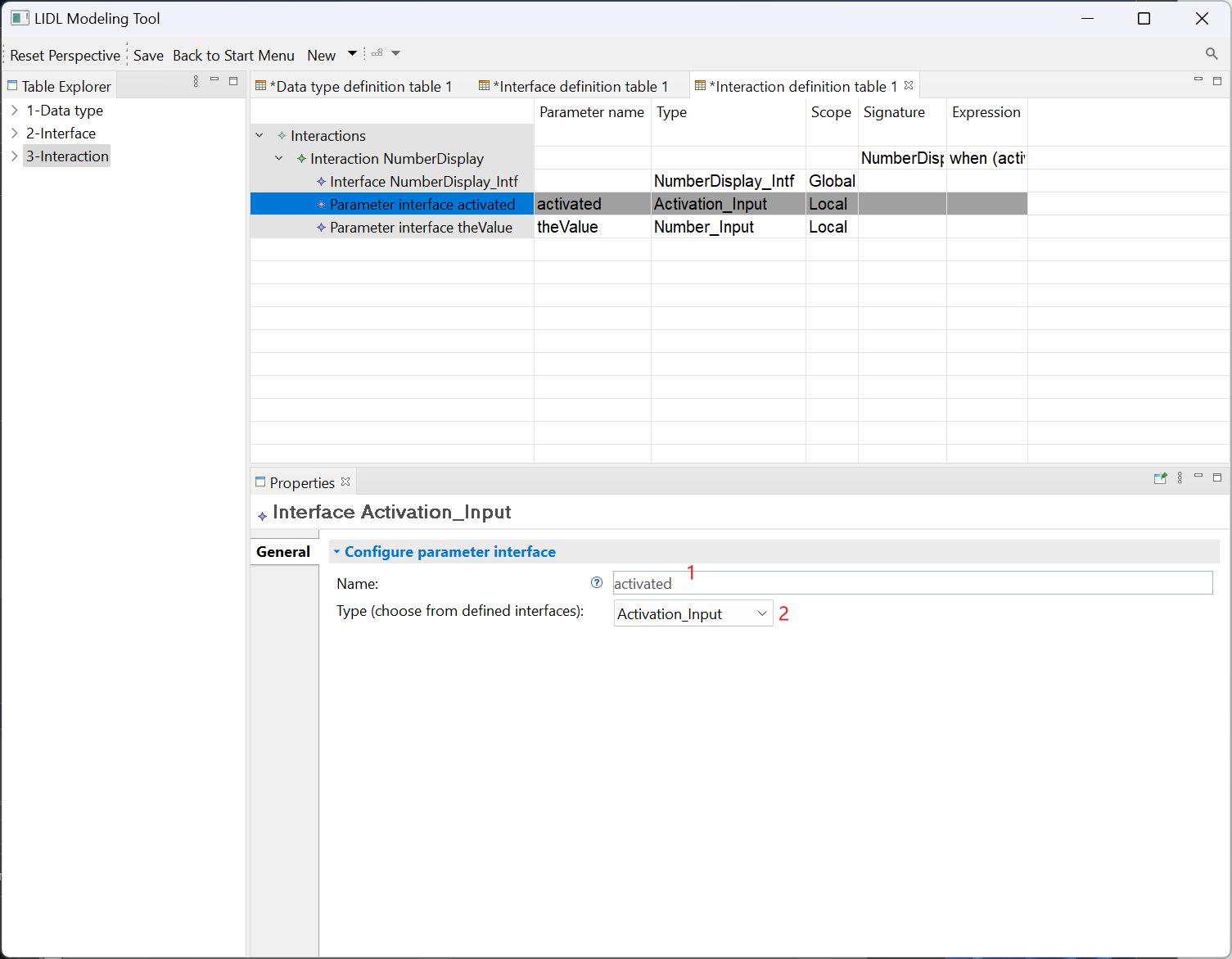
Click the interface null cell, and you can select a defined interface in the Properties view below



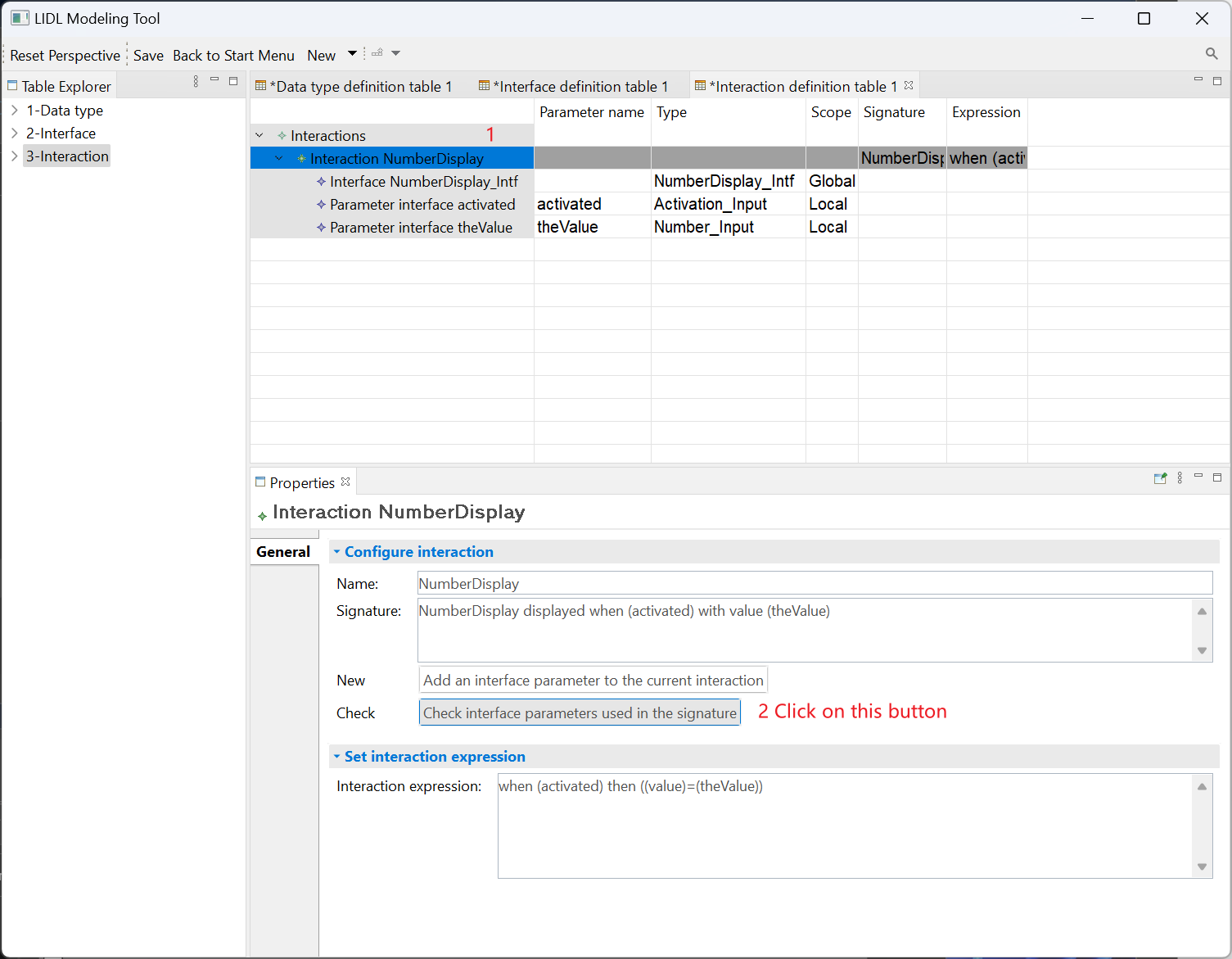
At the same time, you can add interfaces to the interaction, as shown in the figure below. Click the Add an interface parameter to the current interaction button,



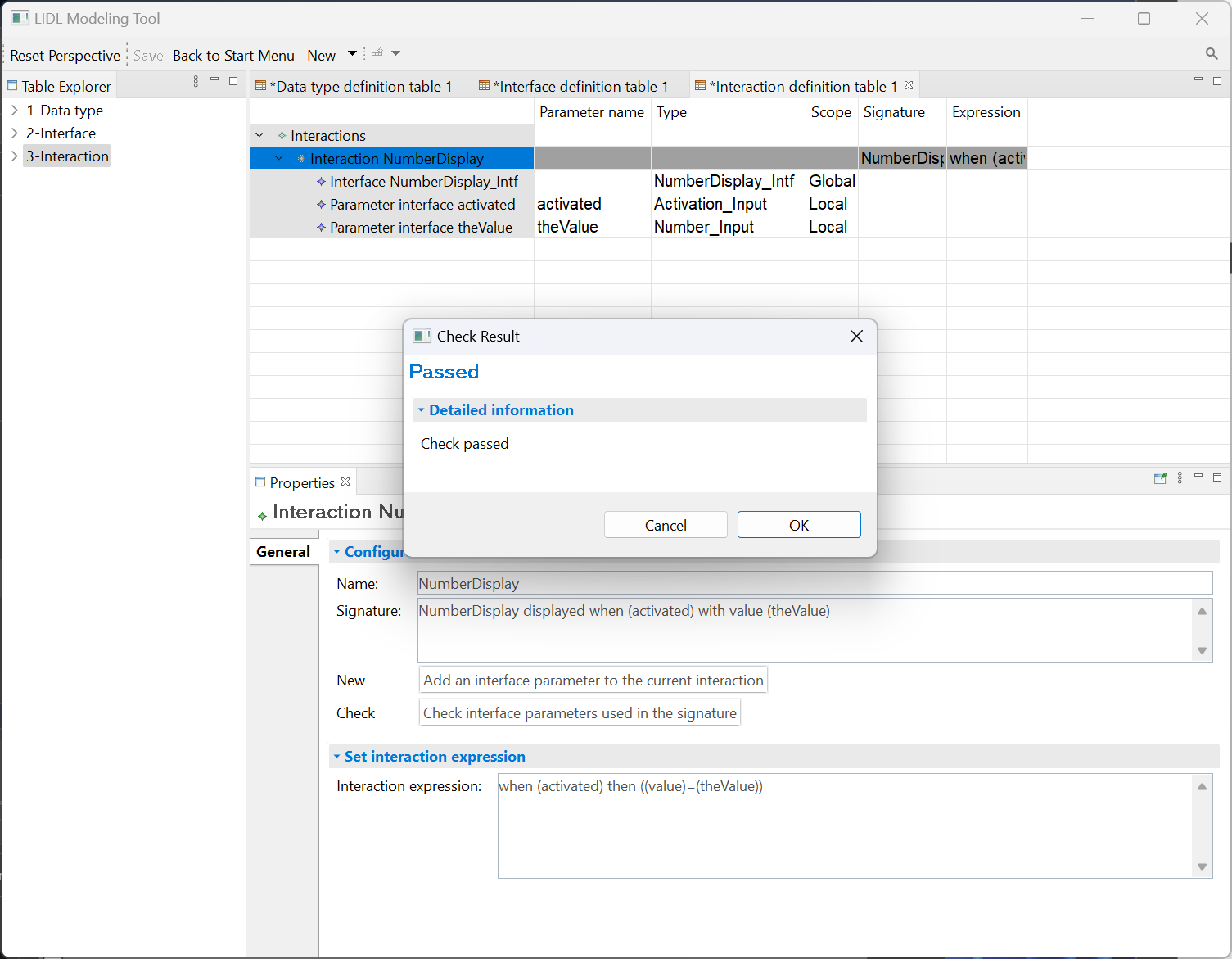
Click the newly created Parameter interface cell, and configure its properties in the Properties view below. The name is activated, and the type is Activation\_ Input.



After adding all the parameter interfaces to the NumberDisplay interaction, you can check if all the parameter interfaces used in the interaction signature have been defined.



The two interface parameters in the NumberDisplay interaction signature have been defined, i.e., activated and theValue, which are content in parentheses. The software will indicate that the check passed.

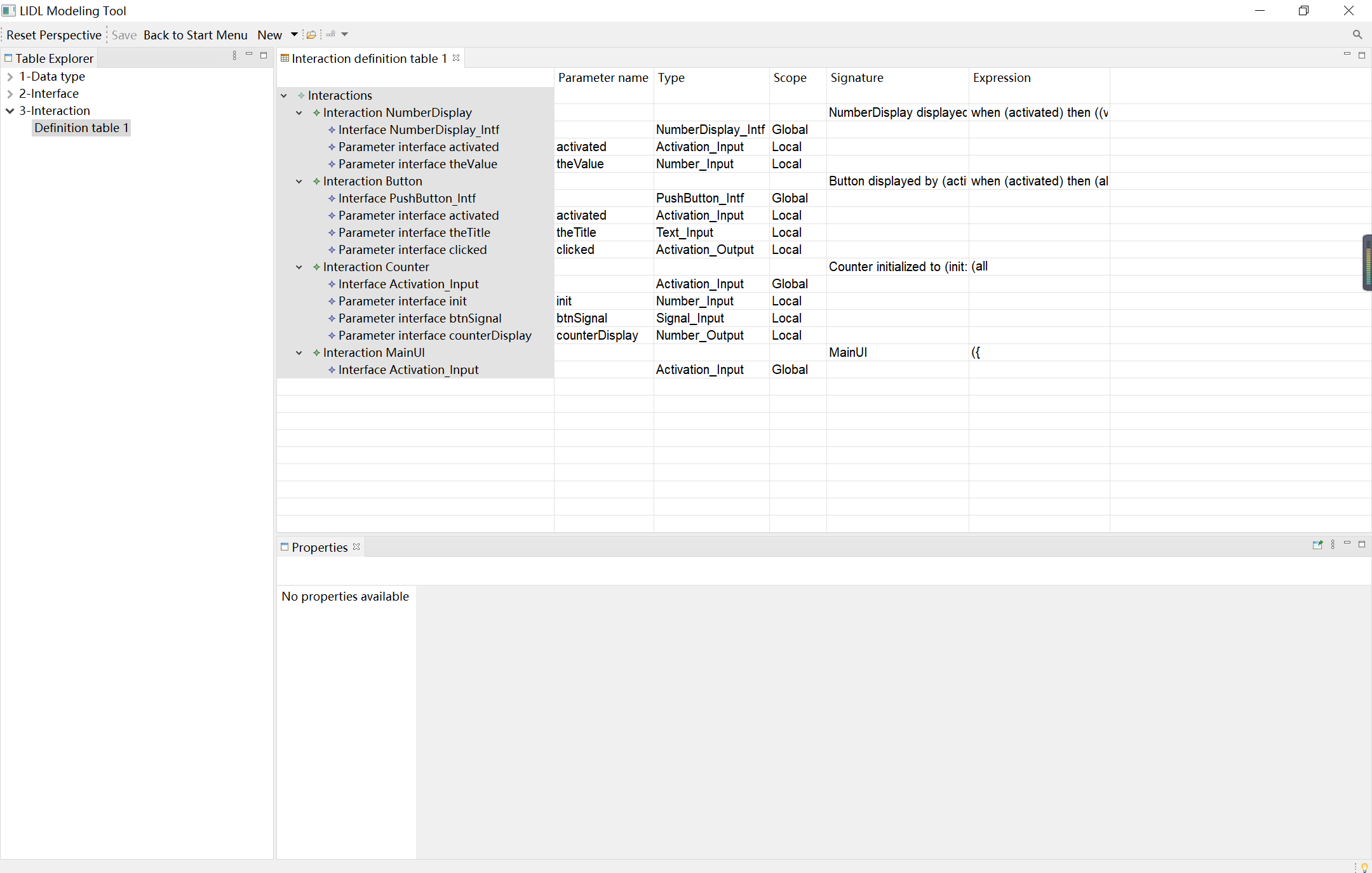


We have demonstrated all the basic operations in the interaction definition table. Now you can define the interactions required by the Counter case according to the provided attributes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NumberDisplay** | **Interface** | **Parameter Name** | **Type** | **Scope** |
|  | NumberDisplay | Global |
| activated | Activation\_Input | Local |
| theValue | Number\_Input | Local |
| **Signature** | NumberDisplay displayed when (activated) with value (theValue) | | |
| **Expression** | when (activated) then ((value) = (theValue)) | | |
| **Button** | **Interface** | **Parameter Name** | **Type** | **Scope** |
|  | PushButton\_Intf | Global |
| activated | Activation\_Input | Local |
| theTitle | Text\_Input | Local |
| clicked | Activation\_Output | Local |
| **Signature** | Button displayed by (activated) with title (theTitle) triggers (clicked) | | |
| **Expression** | when (activated) then (all ((title) = (theTitle)) ((clicked) = (pushed))) | | |
| **Counter** | **Interface** | **Parameter Name** | **Type** | **Scope** |
|  | Activation\_Input | Global |
| init | Number\_Input | Local |
| btnSignal | Signal\_Input | Local |
| counterDisplay | Number\_Output | Local |
| **Signature** | Counter initialized to (init) answer click on (btnSignal) buttons and displays (counterDisplay) | | |
| **Expression** | (all  ((counterDisplay) = (counter))  (if (((btnSignal).pActive) is active)  then ((counter) = ((previous (counter))+(1)))  else  (if (((btnSignal).mActive) is active)  then ((counter) = ((previous (counter))-(1)))  else  (if (((btnSignal).rActive) is active)  then ((counter) = (0))  else ((counter) = (previous (counter))))))  )  With Behaviour  (make (counter) is a flow initialy (init)) | | |

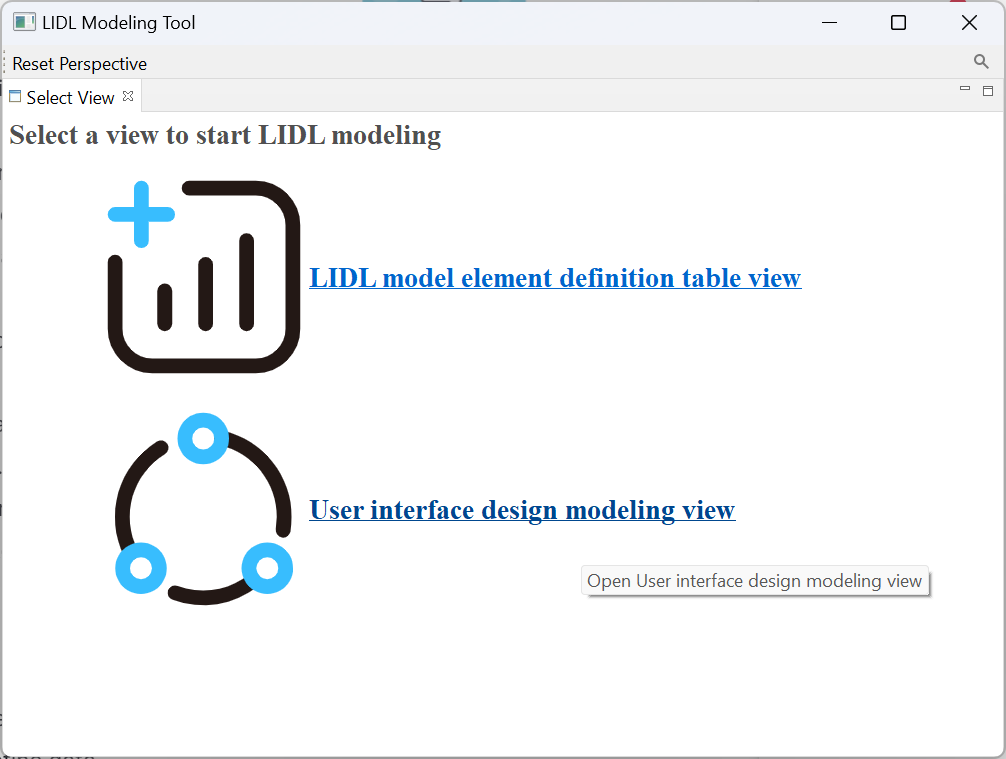
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MainUI** | **Interface** | **Parameter Name** | **Type** | **Scope** |
|  | MainUI\_Intf | Global |
| **Signature** | MainUI | | |
| **Expression** | ({  plusBut: (Button displayed by (active) with title ("+") triggers (plusClicked)),  minusBut: (Button displayed by (active) with title ("-") triggers (minusClicked)),  resetBut: (Button displayed by (active) with title ("reset") triggers (resetClicked)),  display: (NumberDisplay displayed when (active) with title (counterDisplay))  })  With Behaviour  (Counter initialized to (0) answer click on (plusClicked) or (minusClicked) or (resetClicked) buttons and displays (counterDisplay)) | | |

You can see the final effect in the following figure.

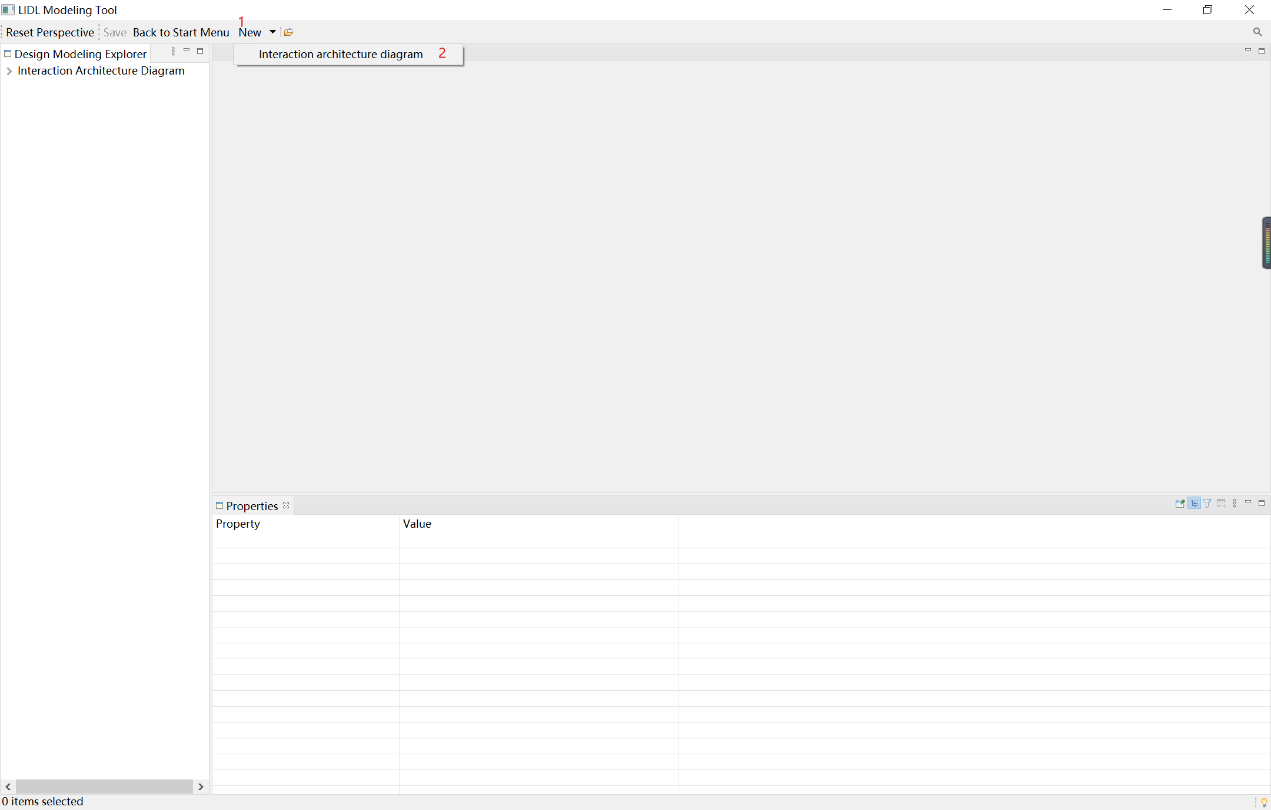


## Design modeling view

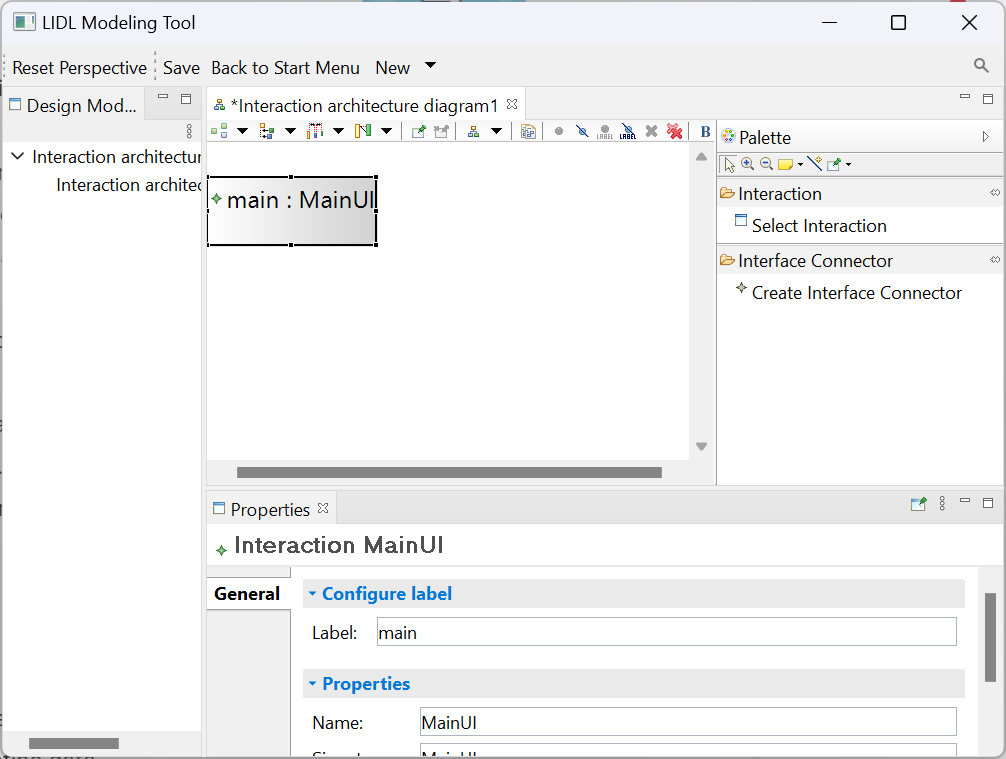
Click on the button Back to Start Menu, then click on the User interface design modeling view.



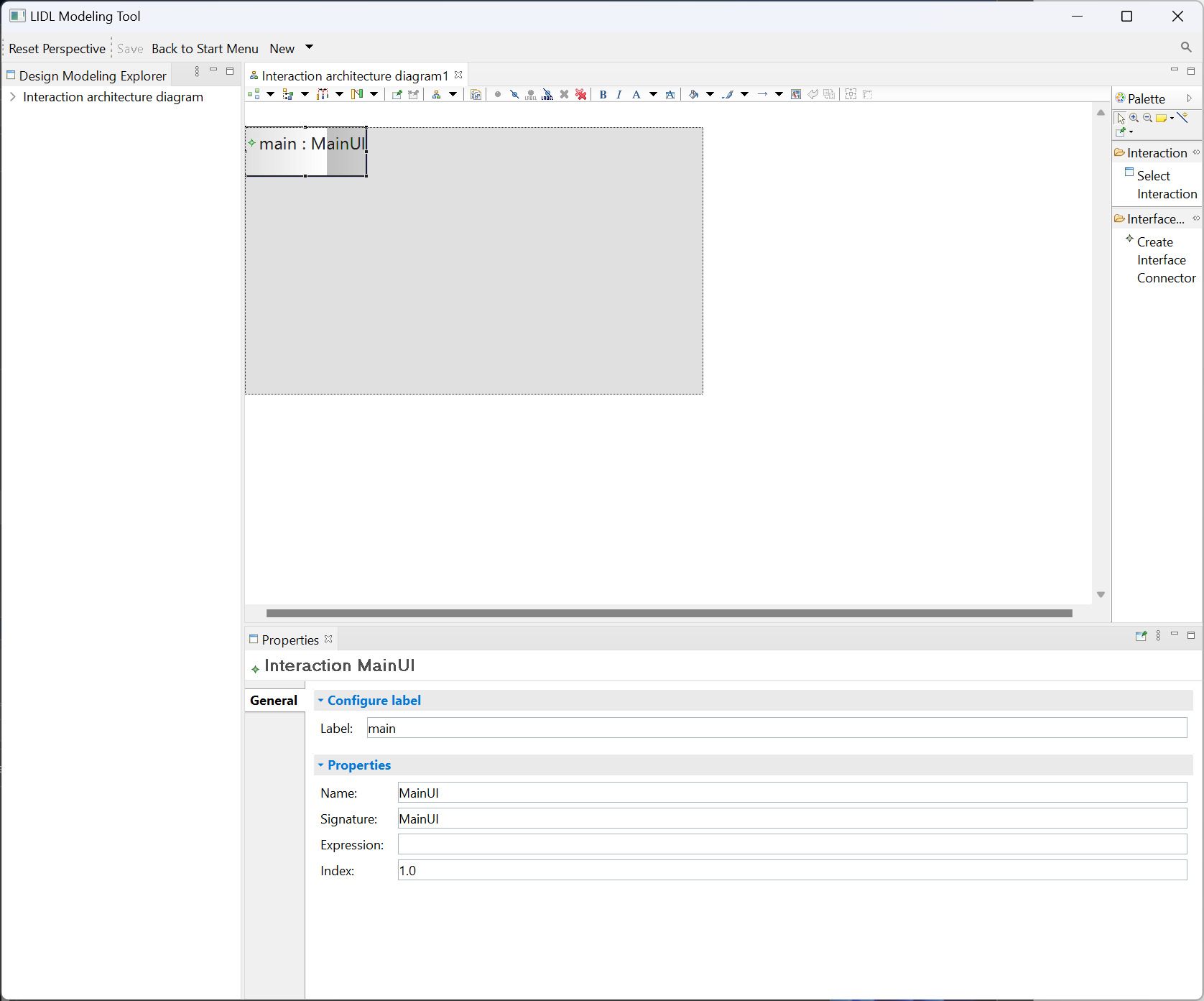
After entering the user interface design modeling view, click New, and then click the Interaction architecture diagram.



The new Interaction Architecture diagram comes with a Main Interaction.



Place the mouse on the edge of the Main Interaction rectangle to adjust the size of the Main Interaction.

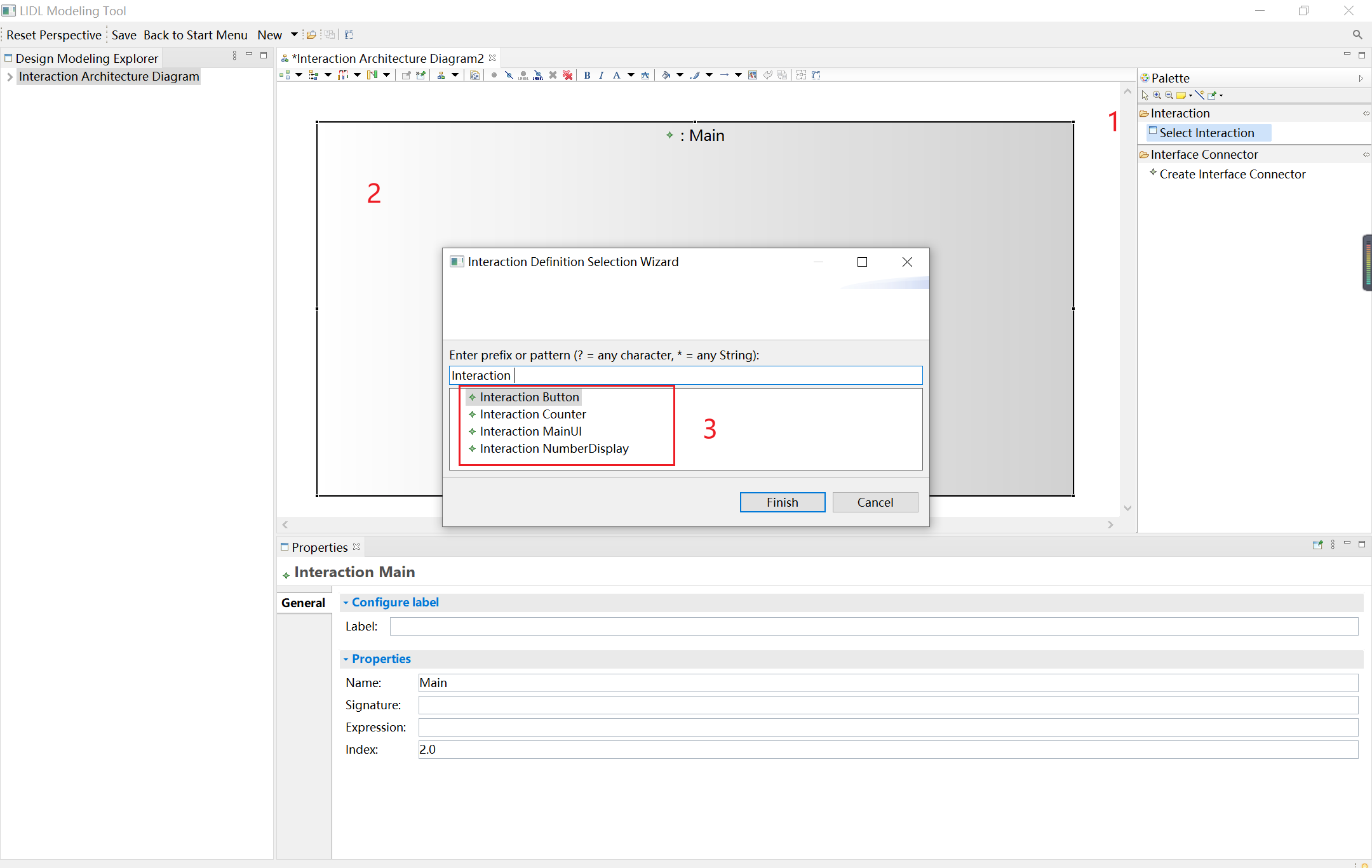


Click Select Interaction in the right toolbar, and then click on the Main Interaction to bring up the selection wizard as shown below.

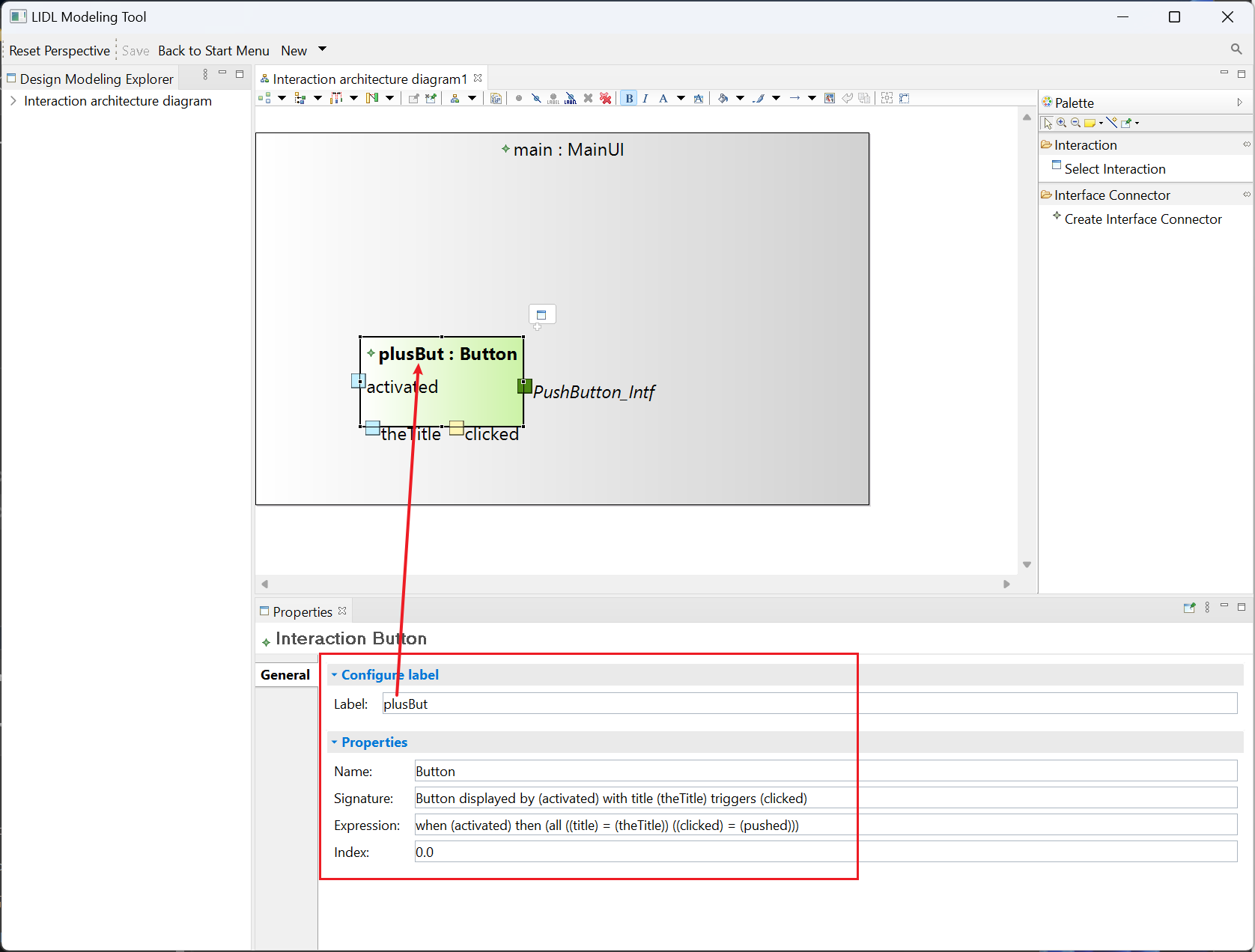
You can select interactions to add to the Main Interaction in the selection wizard.

Specifically, click on an interaction and then click the Finish button to complete the addition.

Note: Only interactions that have been defined in the Interaction definition table in the Definition table view will be displayed in the selection wizard list here.

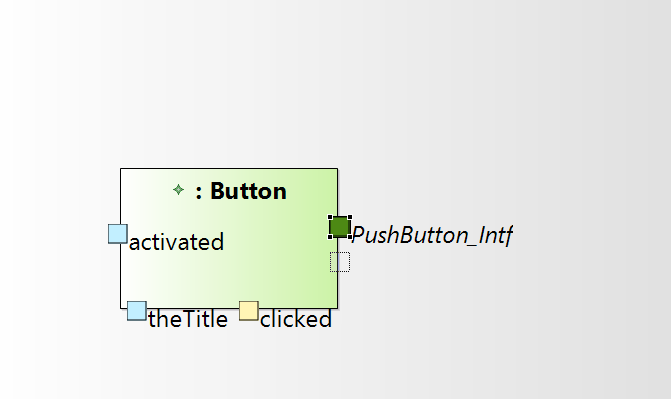


Click the interaction in the diagram, and you can see the properties of the interaction in the Properties view below. It contains configurable interactive labels in the diagram after configuration.



You can see the interface of the interaction on the edge of the interaction.

You can click on any interface with the left mouse button and hold down the left mouse button to move the mouse and move the interface anywhere on the edge of the interaction.



The tool uses different colors to distinguish between different interface directions, specifically, green for io-type interfaces, blue for in-type interfaces, and yellow for out-type interfaces.

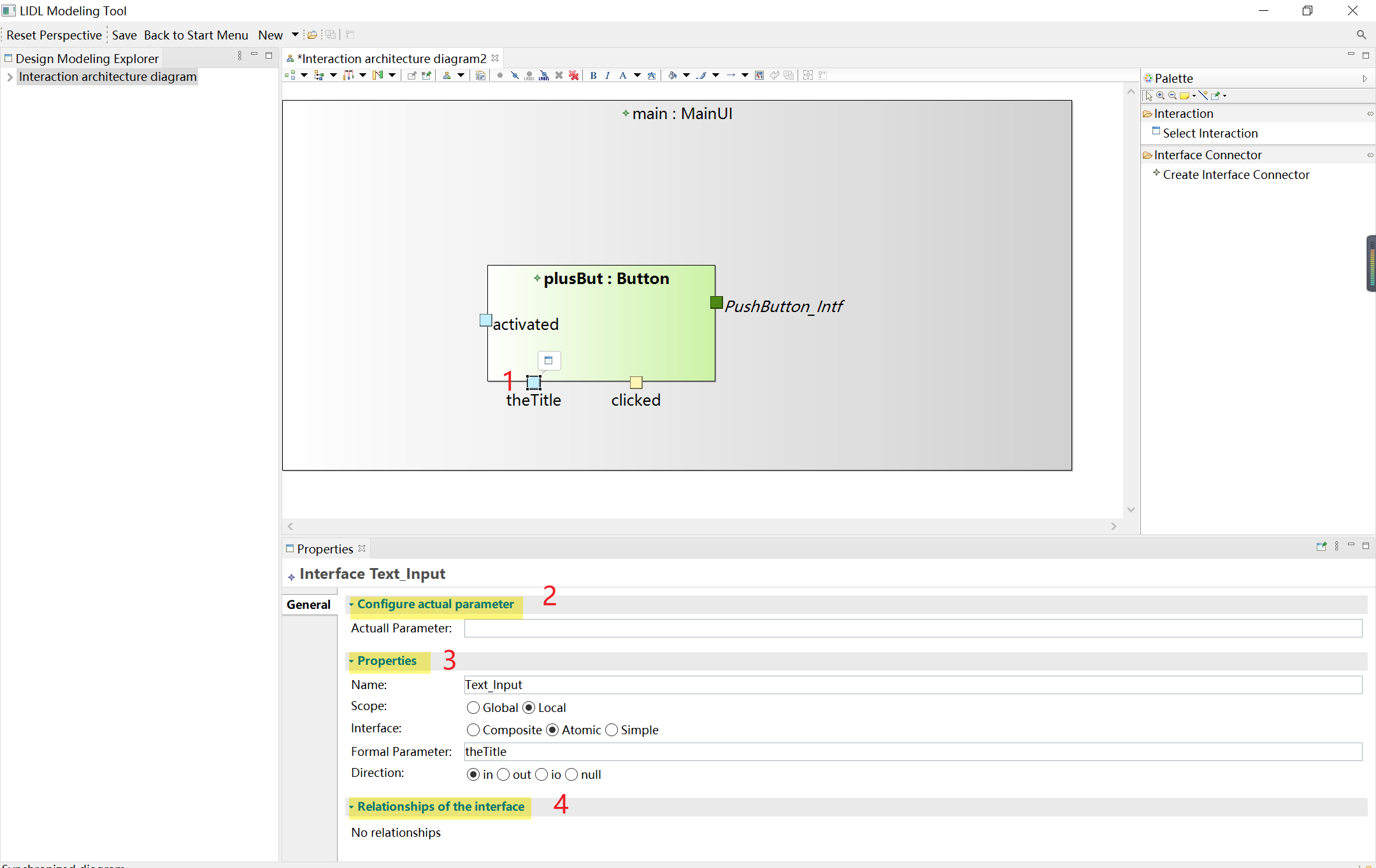
Clicking on an interface node will display the interface properties in the Properties View section below

The interface properties include three parts.

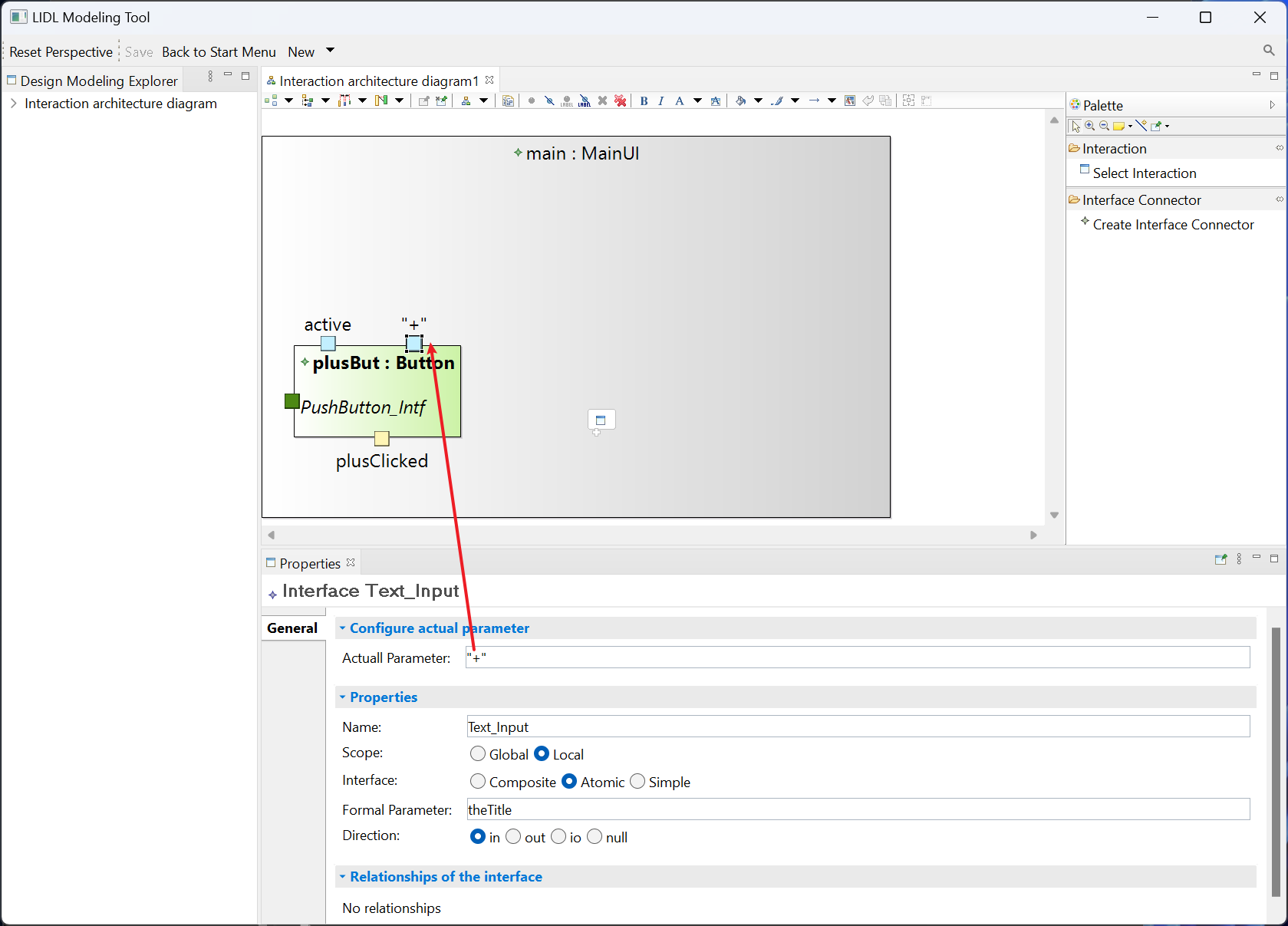
2 in the following figures include the configurable actual parameters.

3 in the figure below shows the read-only properties of the interface.

4 in the figure below shows the relationship of an interface to other interfaces. Currently, there are no relationships, so No relationships are shown.



Enter "+" in the Actual Parameter input box and save (you can press Ctrl and S to save), as you can see in the interactive architecture diagram the interface shows the actual parameters entered. When the actual parameter is not configured, the parameter interface (the interface with the range Local) displays the formal parameter.



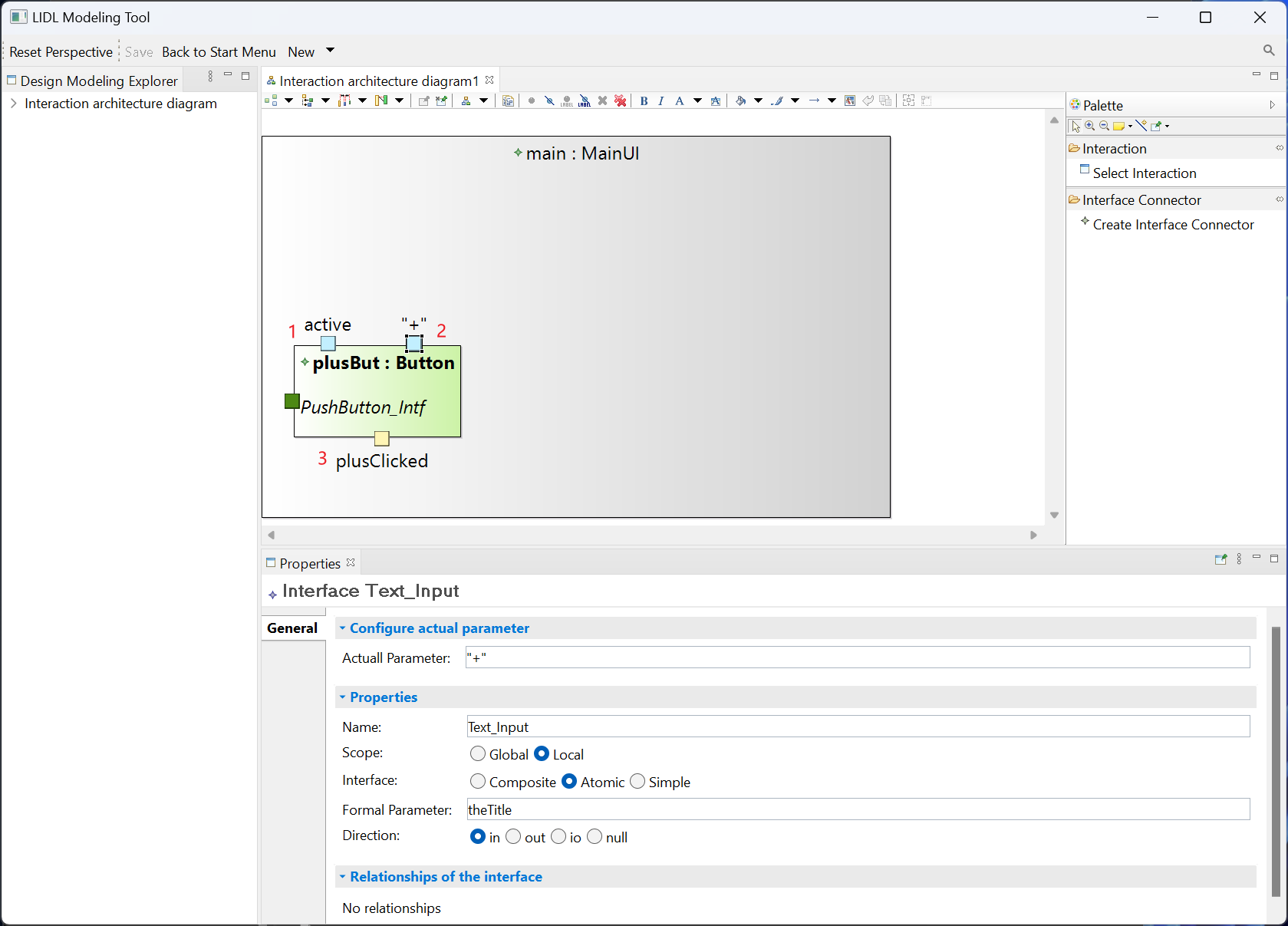
Based on the above demonstration, we configure the actual parameters for the plusBut interface.

Activated: active

theTitle: “+”

clicked: plusClicked

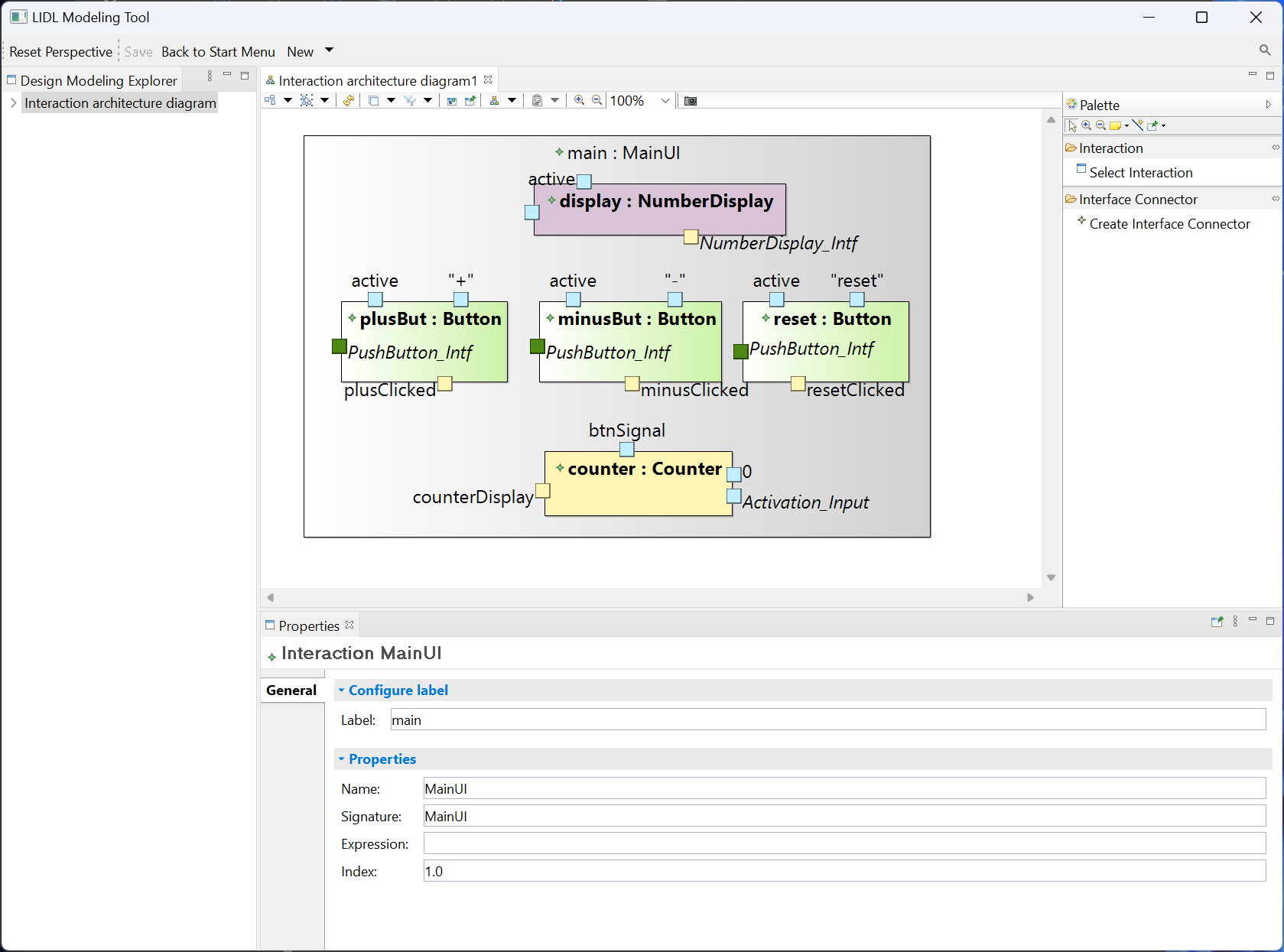
You can see the final effect in the following figure:



Similarly, following the above process, you can create the following interactions required by the Counter case and configure the actual parameters of the interface.

|  |  |  |  |
| --- | --- | --- | --- |
| **Interaction** | **Label** | **Formal Parameter** | **Actual Parameter** |
| Button | minusBut | activated | active |
| TheTitle | “-” |
| clicked | minusClicked |
| Button | resetBut | activated | active |
| TheTitle | “reset” |
| clicked | resetClicked |
| NumberDisplay | display | activated | active |
| theValue |  |
| Counter | counter | counterDisplay | counterDisplay |
| btnSignal |  |

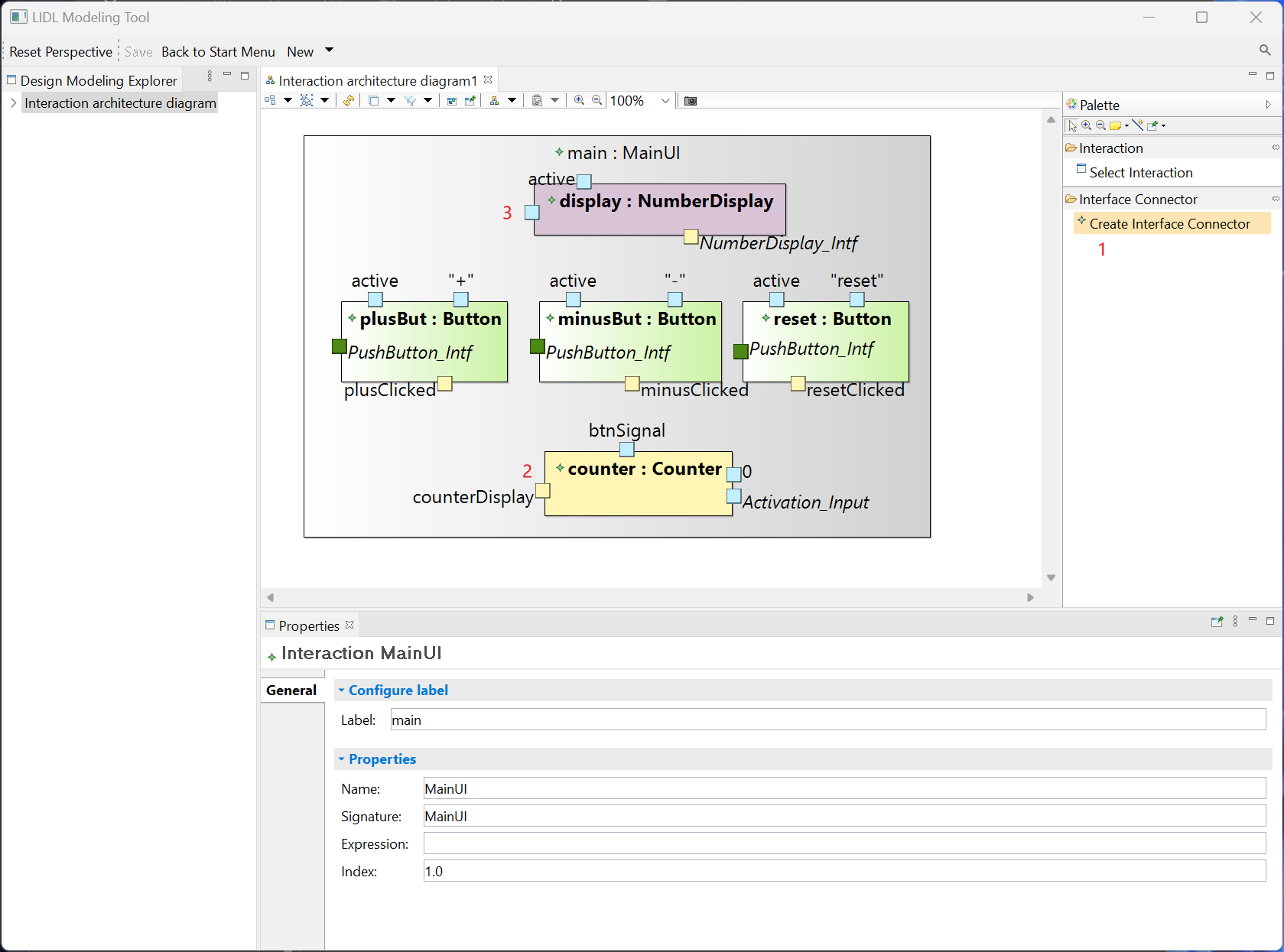
You can see the final effect in the figure.

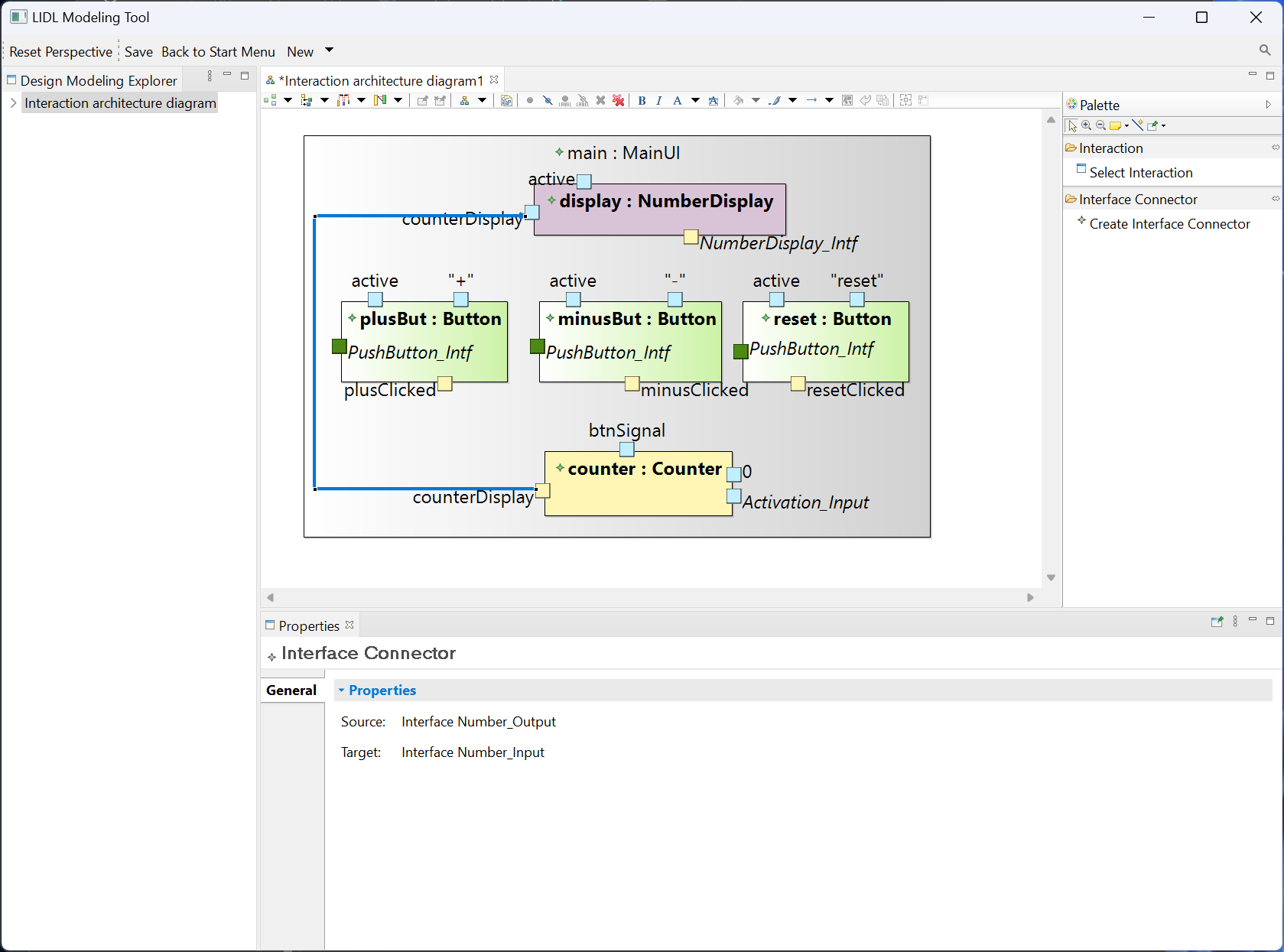


Create the interface connector.

Click Create Interface Connector, then click the counterDisplay interface of counter and finally click the counterDisplay interface of display to connect these two interfaces.

Click on the Interface Connector and the Properties View below will show its Source and Target.

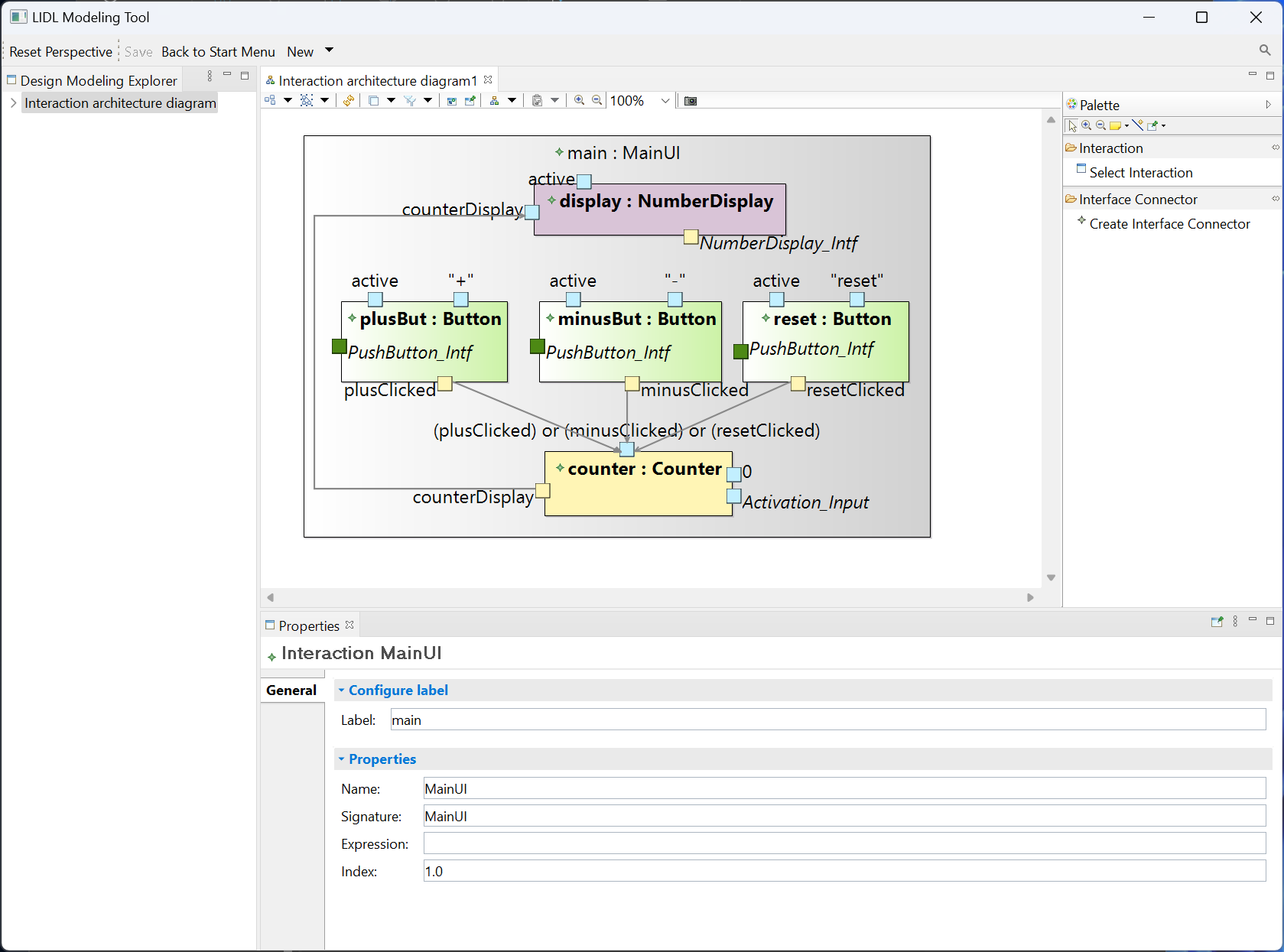




Follow the above procedure to create the interface connector in the following table for Counter case.

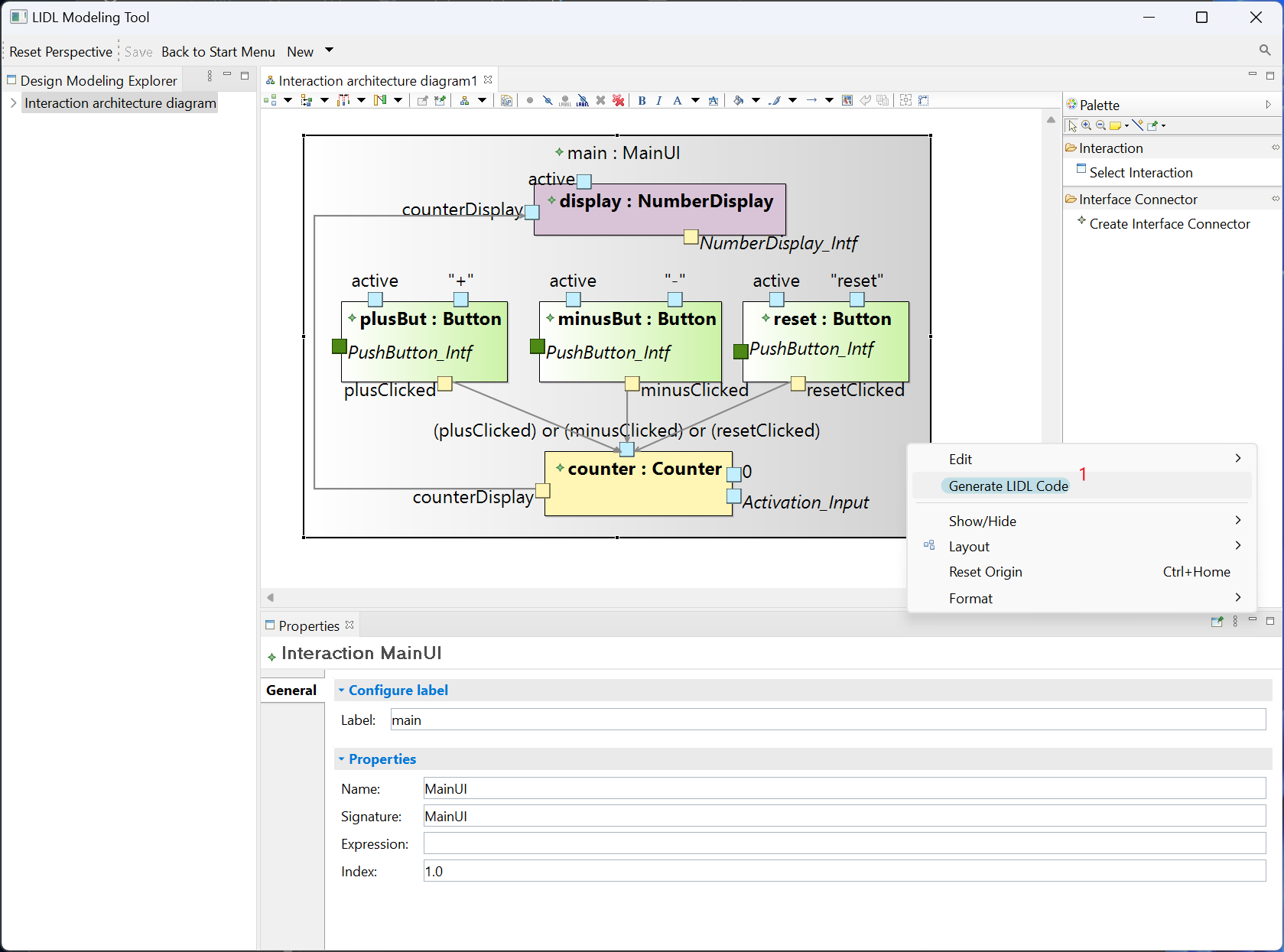
|  |  |
| --- | --- |
| **Source Interface** | **Target Interface** |
| counter: counterDisplay | display: counterDisplay |
| plusBut: plusClicked | counter: btnSignal |
| minusBut: minusClicked | counter: btnSignal |
| resetBut: resetClicked | counter: btnSignal |

You can see the final effect in the figure.

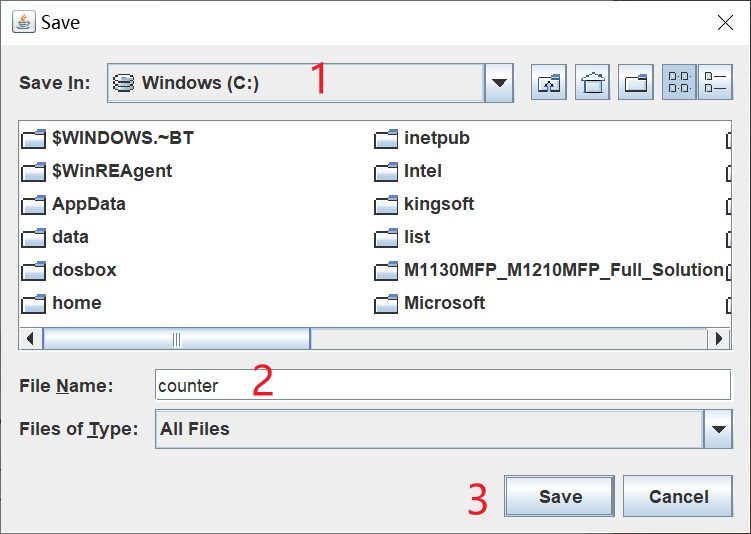


So far, you have completed the design and modeling of Counter's interactive architecture diagram.

In the Interactive architecture diagram, right-click and click Generate LIDL Code to bring up the Save File window.



In the Save File window, you can configure the file save path and name the file, here we name the file counter. Click Save and you will find the counter.lidl file in the corresponding path.



counter.lidl file

