Derived Components

A *derived component* is a user-created component that can be used in workflows alongside standard platform components. The *base node* defines the data processing algorithm, input ports, and output ports of the derived component.

Base node

- Any node can serve as the base node, except for Loop, Node Execution, Reference Node, and reference or derived nodes for a connection.
- Typically, a supernode is used as the base node. The supernode allows for flexible data processing logic and customizable input/output ports. This enables users to create versatile derived components suitable for various analytical tasks.

This approach expands the platform's data processing capabilities beyond standard components, allowing users to implement structured designs for complex solutions (see Workflow Design).

Properties of derived components

- **Inheritance:** Changes to the base node automatically propagate to the derived component and all workflow nodes created from that component.
- **Overriding:** Derived nodes can be modified independently of the derived component from which they were created.

Derived nodes

Derived nodes are created from derived components and are descendants of the base node. Inheritance is managed by updating derived nodes when their base node changes. If a base node is modified, a ⚠ **Base node has been changed** notification will appear in all its derived nodes. These changes are applied to the derived nodes when:

- Selecting **Update node configuration** from the derived node's context menu.
- Activating or retraining the derived node, or attempting to reconfigure the node or one of its ports.

Modifying the properties and settings of derived nodes (i.e., *overriding*) does not affect the base node. If the base node's settings are changed, only the settings not overridden by the user will be updated in the derived node.

Overriding: Prohibited changes

The following table lists operations that are not allowed when overriding a derived node:

Base node	Object	List of prohibited operations
Any	Columns and variables in input/output ports	• Delete • Change data type • Rename
Supernode	Nodes within the supernode	• Set the derived node as the connection node
	Connections	• Delete
	Derived components	• Delete or change the base node
Nodes with the variable number of ports, including Supernode	Input and output ports	• Delete • Change type • Change name

To revert all changes made to a derived node and restore its original configuration, identical to the base node, select **Restore original node configuration** from the derived node's context menu.

> **Important note:** For example, consider a filtering condition such as `< 01.01.2020`. This condition can be broken down into two "atomic" parts: the operator `<` and the value `01.01.2020`. In this case, the override and inheritance mechanisms act on each of these parts independently, rather than on the filtering condition as a whole. This means if you override one part (like the operator `<`), the other part (the value `01.01.2020`) can still inherit changes if the corresponding part in the base node is modified.
> It's also worth noting that in different components, these distinct parts can be either individual settings or groups of settings.

Creating and configuring a derived component

To create a derived component:

1. Select the base node (a supernode).
2. Choose **Create derived component** from its context menu. This command is also available on the workflow area toolbar.

The **New derived component** window will open, where you can define the following:

- **Caption:** The name of the derived component.
- **Description:** Additional information about the component.
- **Visibility:** The availability of the derived component across different package levels:
  - **Private:** Available only in the current module.
  - **Protected:** Available only in the current module and its supernodes.
  - **Internal:** Available only in the current package.
  - **Public:** Available in all packages. To use the derived component from another package, a reference to the package where the derived component was created is required.

> **Note:** To avoid errors when using derived components that include connections, you should set up not only the derived component's visibility, but also the visibility of each connection.

- **Target node types**: The types of the nodes you can create based on the derived component. You must select at least one of the two options:
  - **Node Execution and Loop** (selected by default)
  - **Derived nodes**

> **Note:** If the derived component has its visibility set as **public**, and **Derived nodes** are selected as a target node type, the contents of the derived node will be available for view in dependent packages even if the derived component is located within an encrypted package.

After saving the settings, the derived component will be available in the **Derived components** pane. You can find them there as lists of derived components, grouped by the packages they were created in.

Right-clicking on the current module provides an option **Go to derived components**. Select it to open the **Components** window which will show the list of available derived components. This window also opens if you right-click a derived component in the list and select **Go to derived component**. In this case, the derived component you clicked will be highlighted in the list.

> **Note:** If the caption of the derived component starts with the underscore (_), this component will not be shown in the Components pane of the other packages even if it has **public** visibility. You can reference such derived components only within its package. You can still create other derived components within the same package. If a node is based on such a "hidden" component with **public** visibility, you can reference the node in other packages without errors. However, in case the hidden component has **private** visibility, errors would occur.

To create a workflow node based on a derived component, right-click the necessary component to open its context menu. You can also drag it to the workflow area:

- A simple drag creates a **Node Execution** node.
- Hold `Shift` to create a **Loop** node.
- Hold `Ctrl` to create a **derived node**.

Nodes created this way will display a **Show base node** button, which helps locate the base node if it's within the current workflow area.

In case a node selected to create a derived component already has one, a dialog window will open, asking you if you want to create another one.

If you create a derived node on the basis of an already existing derived node, the new node will have its caption indexed to show its derivate level (e.g., *#2*, *#3*, etc.)

Editing derived component settings

To edit the settings of derived components within a package:

1. Go to the **Navigation** pane.
2. Navigate to **Packages > Current package > Module > Components**.

Selecting this section opens a pane listing the derived components, with the following actions available:

Toolbar actions

- **Group (`Alt` + `G`):** Groups the list of derived components by base node type (e.g., *import, transformation, control*, or *exploration*).
- **Configure:** Opens the configuration window for the selected derived component.
- **Delete:** Deletes the selected derived component.

Context menu actions

- **Configure:** Opens the configuration window for the selected derived component.
- **Delete:** Deletes the selected derived component.
- **Go to base node:** Navigates to the base node's location in the workflow.
- **Refresh:** Updates the list of derived components.

Read on: Visibility Configuration