FLOWX.AI

BUILDING BLOCKS / Node / subprocess-run-node

Contents

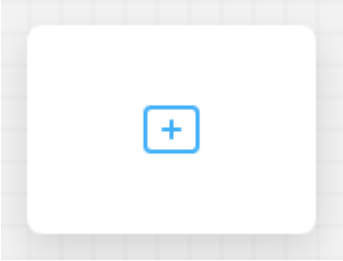
BUILDING BLOCKS / Node / Subprocess run node

> **INFO**
>
> There might be cases when extra functionality is needed on certain
>
> The fallback content to display on prerendering
>
> .

A node that provides advanced options for starting

The fallback content to display on prerendering

.



» Subprocess

It contains a default action for starting a subprocess.

A subprocess can be started in two modes:

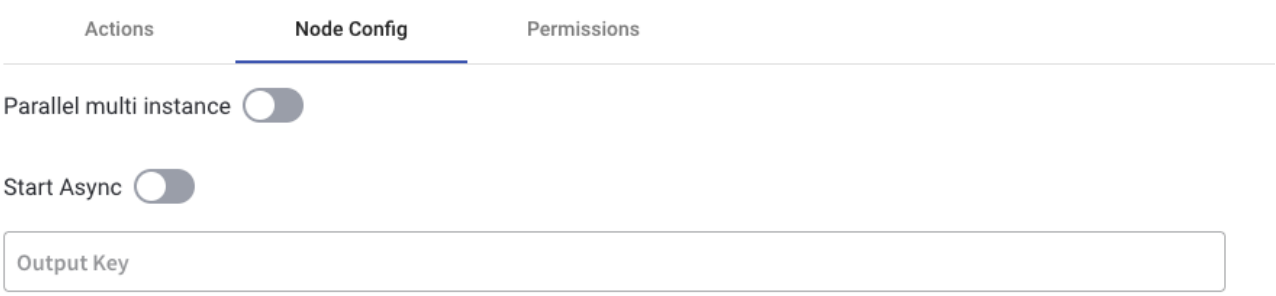
- **async mode** - the parent
  The fallback content to display on prerendering
  will continue without waiting for the sub-process to finish

> **INFO**
>
> Select if this task should be invoked asynchronously. Make tasks asynchronous if they cannot be executed instantaneously, for example, a task performed by an outside service.

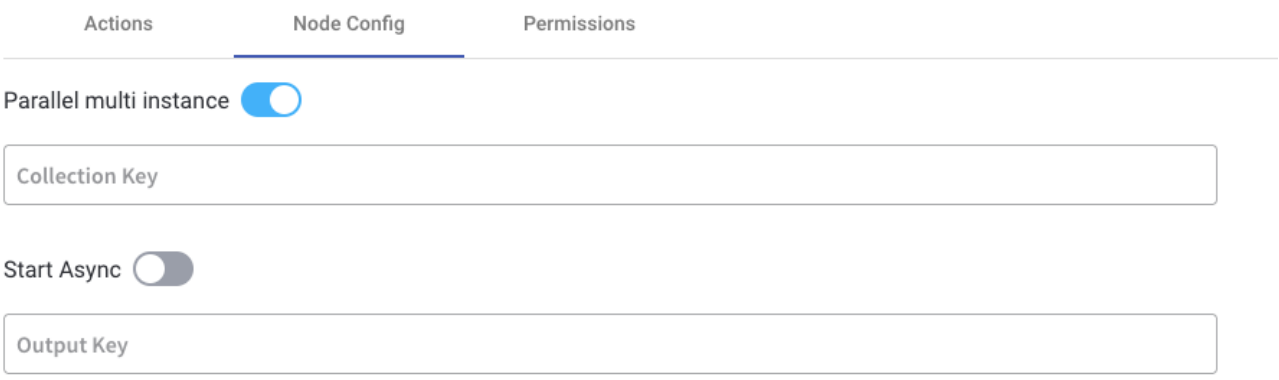
- **sync mode** - the parent process must wait for the subprocess to finish before advancing

The start mode can be chosen when configuring the subprocess run node.

In case the parent process needs to wait for the subprocess to finish and retrieve some results from it, the parent process key that will hold the results must be defined using the *output key* node config value.



This node type can also be used for starting a set of subprocesses that will be started and run at the same time. This will prove useful in case we have an array of values in the parent process parameters and we want to start a subprocess for each of the elements in that array.



In order to do this, we need to select the parallel multi instance option. The *collection key* name from the parent process also needs to be specified.

> **INFO**
>
> When designing such a subprocess that will be started in a loop, you need to keep in mind that the input value for the subprocess (that is, one of the values from the array in the parent process) will be stored in the subprocess parameter values under they key named *item*. This will have to be used inside the subprocess. If this subprocess produces any results, they should be stored under a key named *result* in order to be sent back to the parent process.

Was this page helpful?