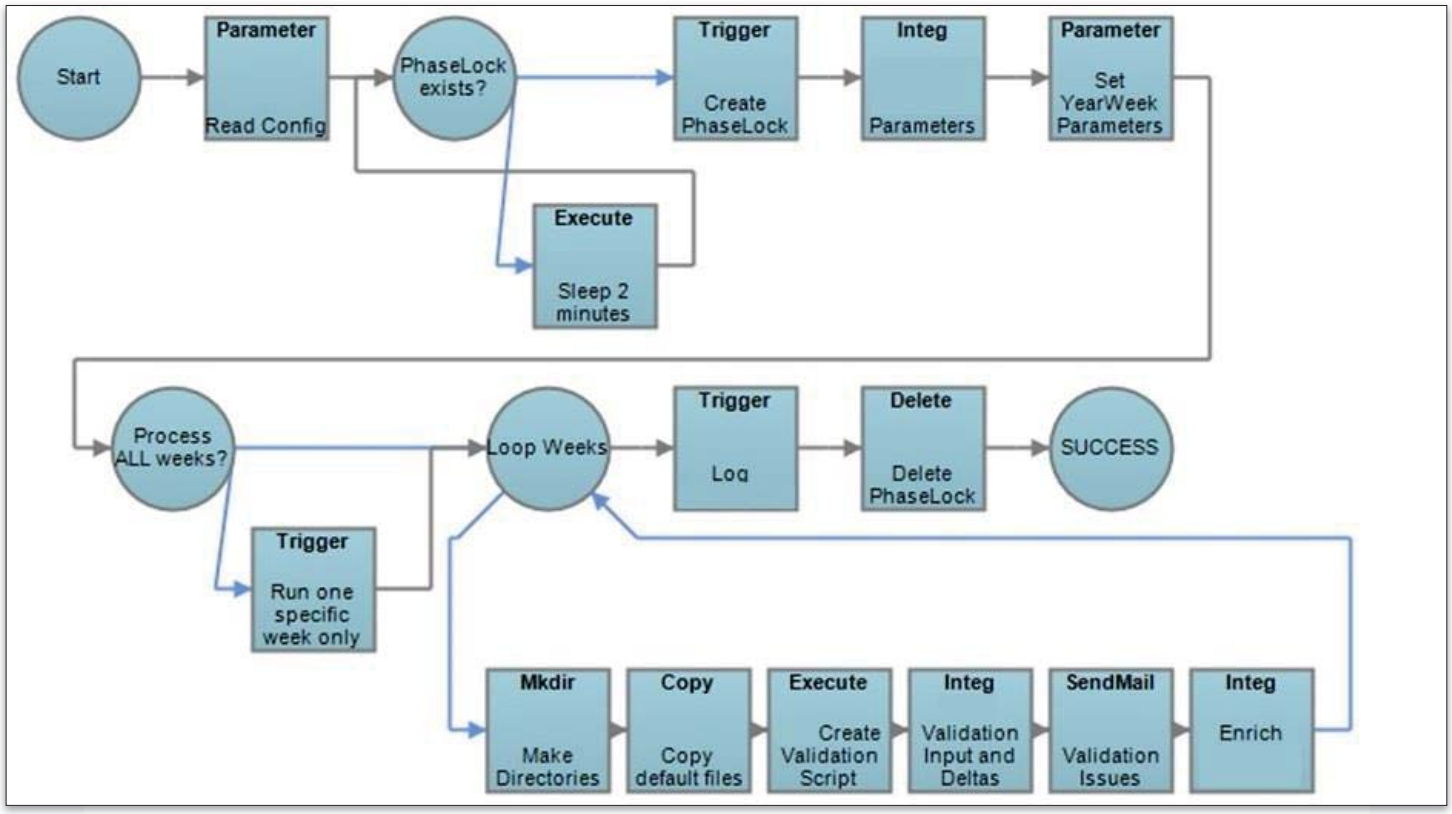
DI-Production

DI-Production is a process flow management tool part of the The Diver Solution. DI-Production provides various options to configure and maintain scripts, test nodes before placing them in a production job, schedule jobs to run at specified intervals, send email to individuals who need to know when a script has finished processing or has failed to finish, view the results of any script, and debug a failed script. It has built-in functionality for conditional and parallel computing and shares its look-and-feel with Visual Integrator using nodes on a canvas. Below shows an example of a DI-Production script.



Example of a DI-Production script

DI-Production Extensions

Introduction

DI-Production has many different built-in nodes. DI-Production also allows the creation of custom, user-made nodes. An example on how to create an extension is given in a different section. The use of extensions is the ability to re-use frequently used scripts or logic in a standardized and packaged way. An extension is installed onto a DiveLine-server, which means that its node is available for any DI-Production script running on that server. So maintenance or bug fixing only requires an update to the one server-wide installation, independent of how many scripts this extension is actually used in. Furthermore, extensions are packaged and can be shared among other users. This way, users which might not know PHP or DIAL can still use nodes written in those languages.

Extension types

DI-Production currently accepts five types of scripts for custom nodes:

Java, PHP, DI-Production, DI-Integrator and DI-Dial.

A script written in any of those languages is eligible for an extension. Note that the path to the Java and the PHP installation folder need to be specified in the DI-Production preferences tab.

Parameters

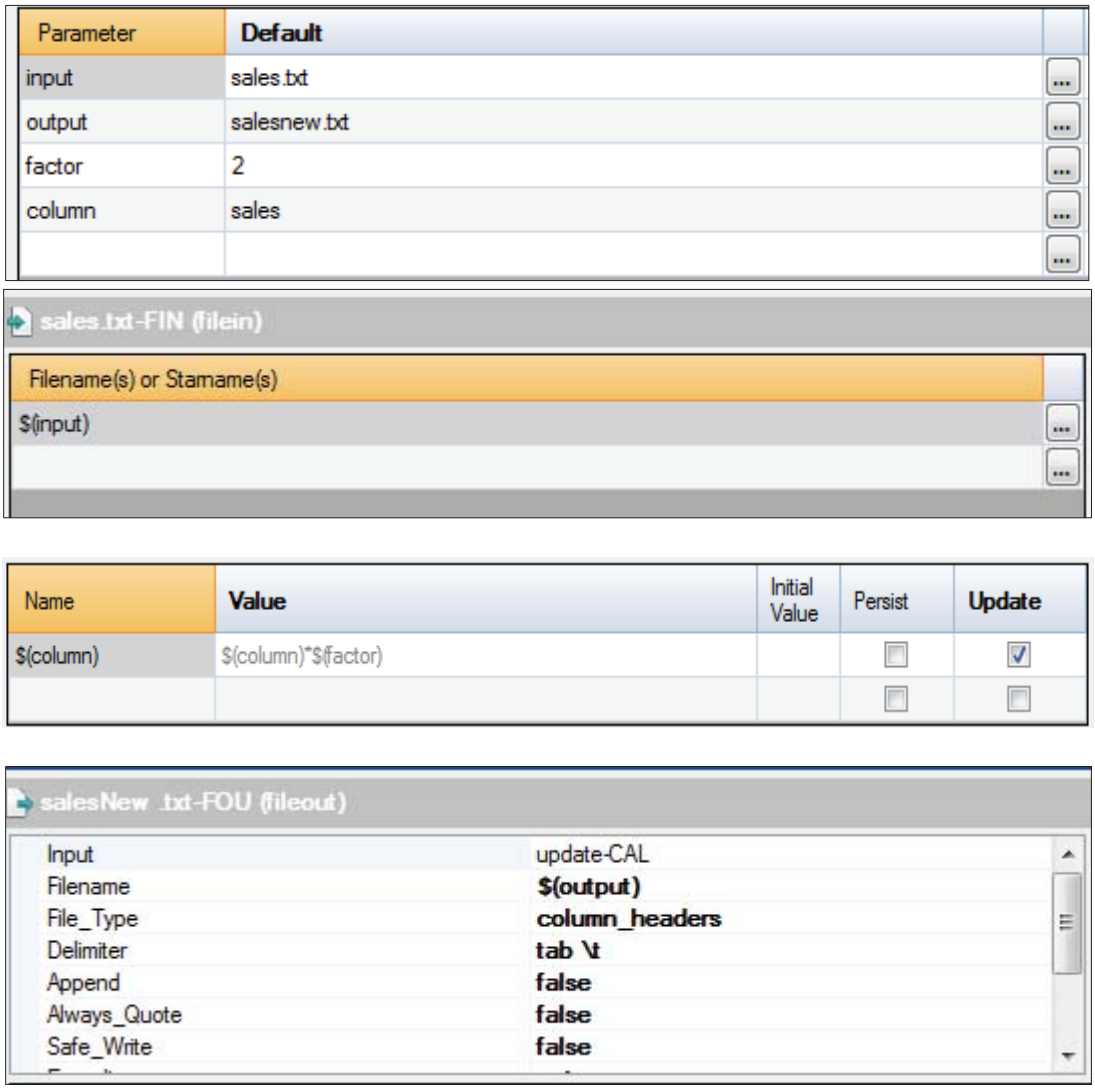
To optimize the use of an extension, the underlying script needs to be generic. To make a script generic any and all settings should be parameterized using the syntax for that specific language. The following section explains how to build an extension from an integrator script that runs a calculation on a specific column from a specific file.
This example Integrator script contains three nodes:

sales.txt is read and the result is written to **salesnew.txt**.
The calculation is the following:

Name	Value	Initial Value	Persist	Update
sales	sales*2		☐	☑
			☐	☐

When parameterizing this script, the first parameter that comes to mind would be the input-file. Alternatively, the output-location, the calculation and even the affected column could be parameters too.

A very generic version of this script would be:



Parameter	Default
input	sales.txt
output	salesnew.txt
factor	2
column	sales

sales.txt-FIN (filein)

Filename(s) or Stamame(s)
$(input)

Name	Value	Initial Value	Persist	Update
$(column)	$(column)*$(factor)		☐	☑

salesNew .txt-FOU (fileout)

Input	update-CAL
Filename	**$(output)**
File_Type	**column_headers**
Delimiter	**tab \t**
Append	**false**
Always_Quote	**false**
Safe_Write	**false**

Later on, a "multiply sales" extension will be created based on this Integrator script, which can read any column from any file, multiply that with any number and write the result into any file.

To keep in mind

There is no golden rule on how generic a script should be. The 'multiply sales' example above is a good example of something overly generic. Unless you're in the multiplication business, having such a generic script for something that trivial only makes scripts less understandable. Before making an extension the two following questions should be asked:

1. Is the script complex enough?
2. Will this logic be often re-used?

If either answer is no, then creating an extension will not be worth the effort. If both answers are yes, then it might very well be worth the one-time investment.

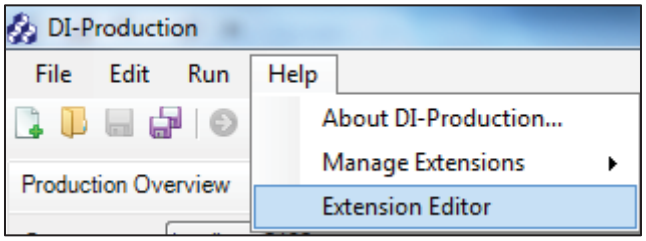
How to create a DI-Production extension

1) Open DI-Production and log onto the server.

2) Create (or select) any script to make an extension out of. In this example, the multiply sales Integrator script discussed in the previous section will be used.

3) Find a suitable icon. Make sure it is a “png”. It MUST be a png-file.

4) Create the Extension “.pre”-file
   a) In DI-Production: open the extension editor.

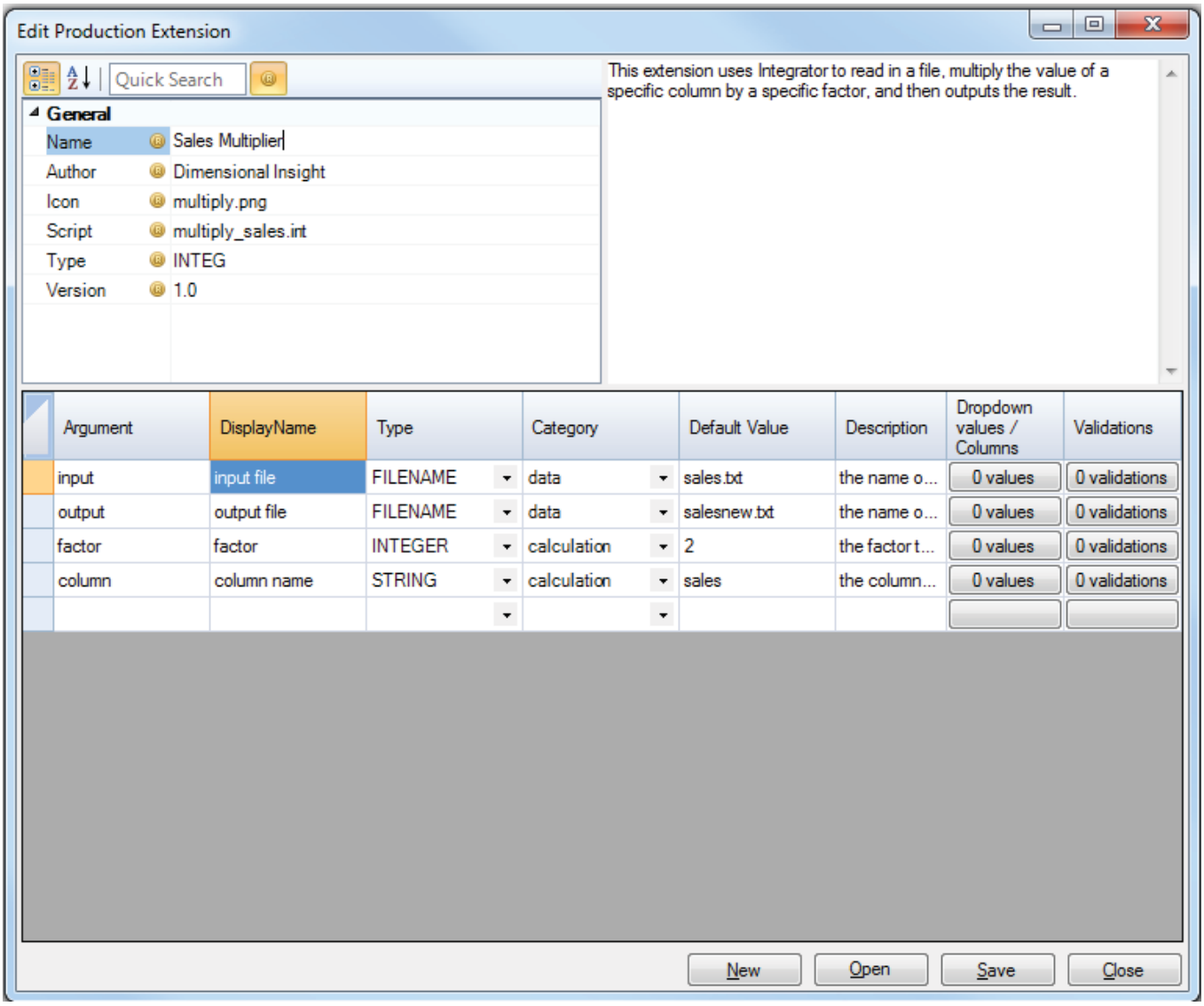


   b) Set the script Type to INTEG and select the multiply_sales.int script which was created earlier. If any global parameters are used in the scripts, then these will be automatically listed and filled with their default values.

      There are several very useful options for parameters. For instance, parameters can be grouped by typing a group name into the category option. Parameters can be made ‘required’ by adding the validation option, and the display name can be changed. Also, by setting dropdown values the user can be restricted to only use specific values. The most useful tool is the parameter type column. The extension automatically alerts if the filled in value does not match the parameter type. The default value is a STRING which comes down to plain text. INTEGER ensures that the parameter is a whole number. BOOLEAN changes the parameter input to a check box stating true or false. FILENAME allows the user to browse to a specific file and DIRECTORY allows the user to browse to a specific folder. The LIST option allows for even further parameter options.

      The end-result of the extension editor is shown on the next page.

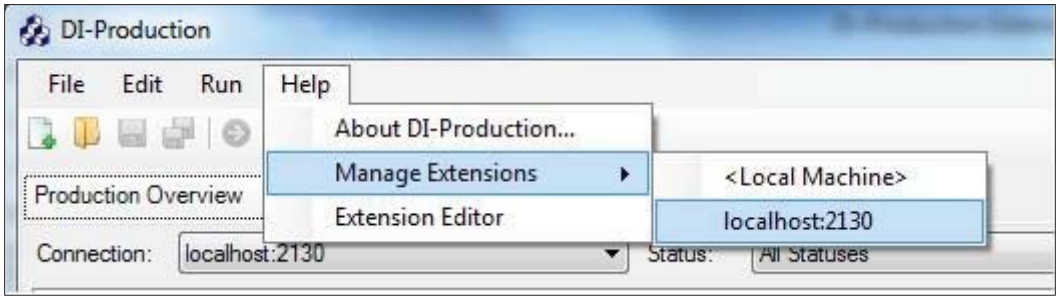
   c) Pressing the save-button will save a ‘.PRE’-file to the desired location. Having created this ‘.PRE-‘file, the extension can now be installed on any DiveLine-server.



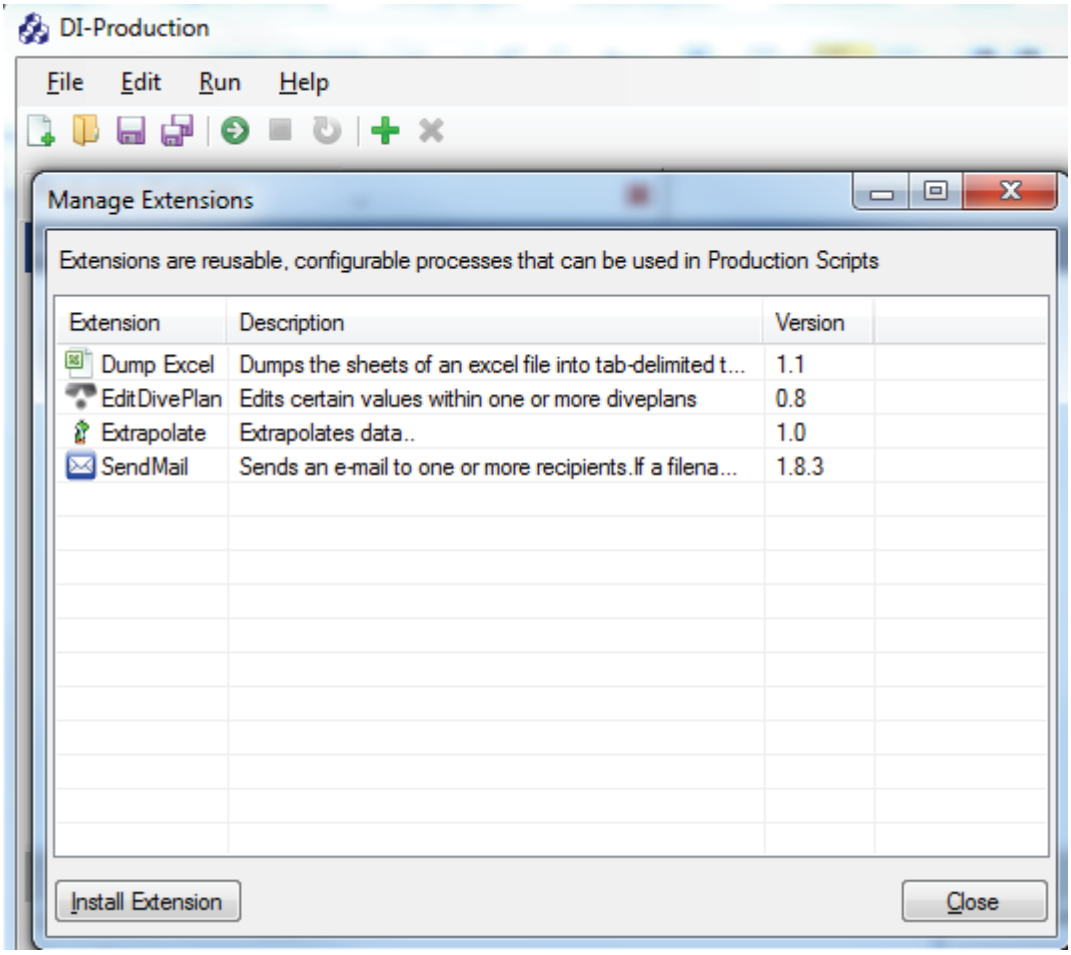
This is what the Extension Editor would show

How to install DI-Production extensions

1) Open DI-Production
2) Open the 'Manage Extensions' dialog and select your server.



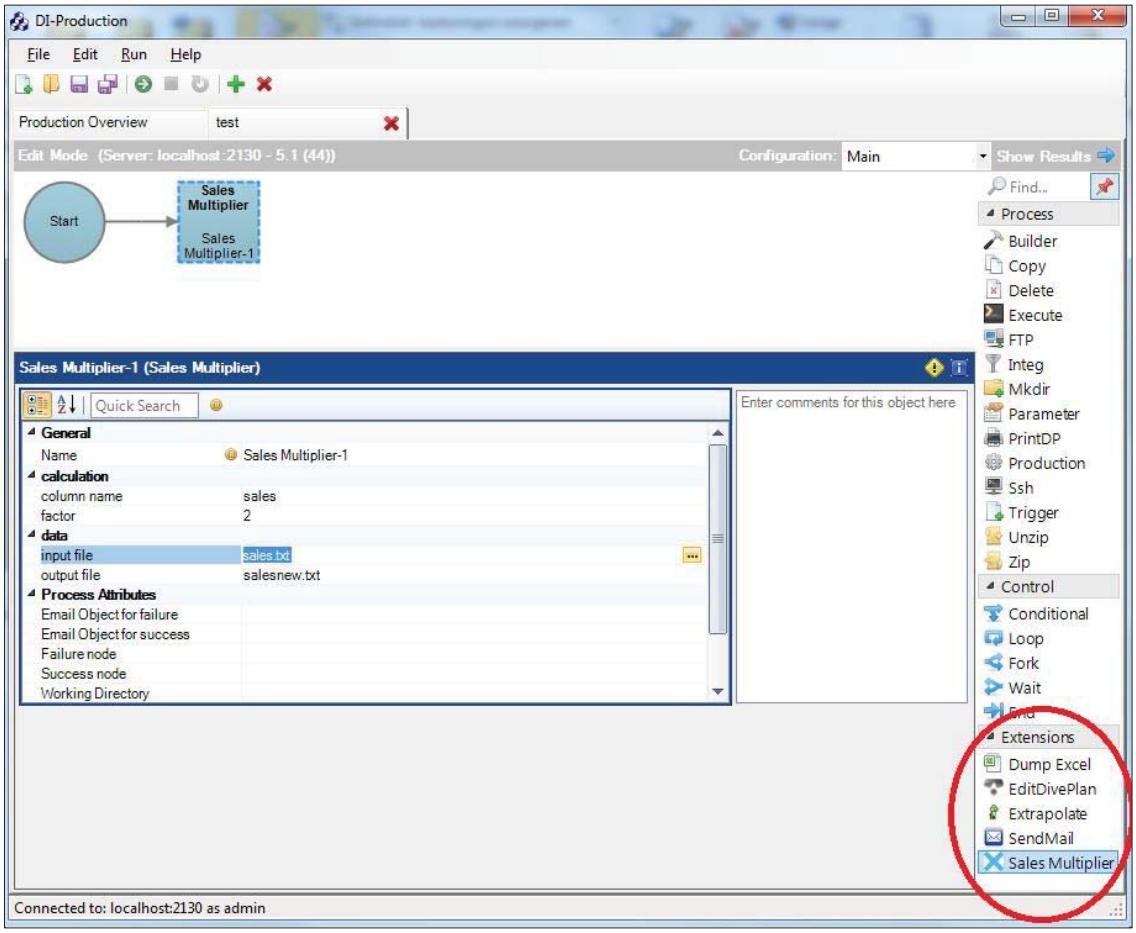
3) The list will show the currently installed extensions and their descriptions. This list will most likely be empty on your installation. Press 'install extension' to add a new extension. Right-clicking an extension allows the removal of an extension or giving additional information about it. Note that after having removed an extension from the server, any script referencing such a node will fail. So remove with caution.



The manage extensions window

After clicking on "Install Extension", browse to the .PRE-file to install and press "open".

4) The Extension has now been installed on the selected server. If the server already knows this extension, then it will try to update the extension. When installing an older version of the same extension, the system will warn you about this.
5) The installed extension nodes will now be available in the bottom right for any user logged onto that DiveLine server.



Installed extensions appear in the bottom-right corner and are available to all users.