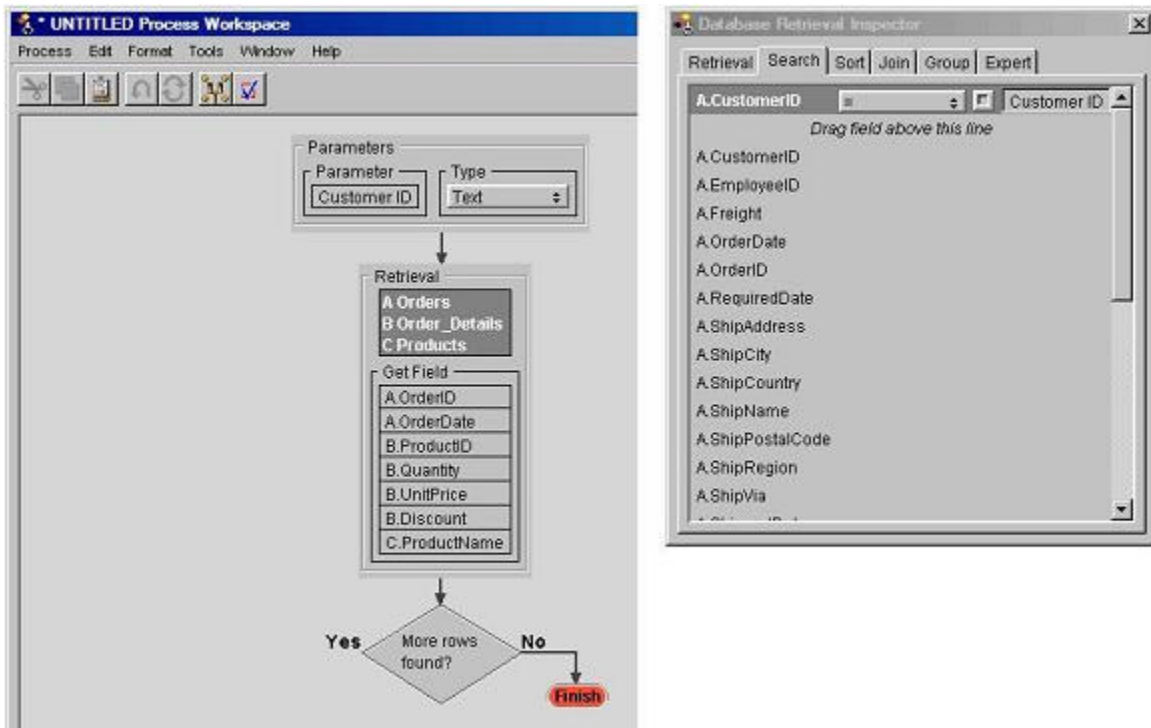


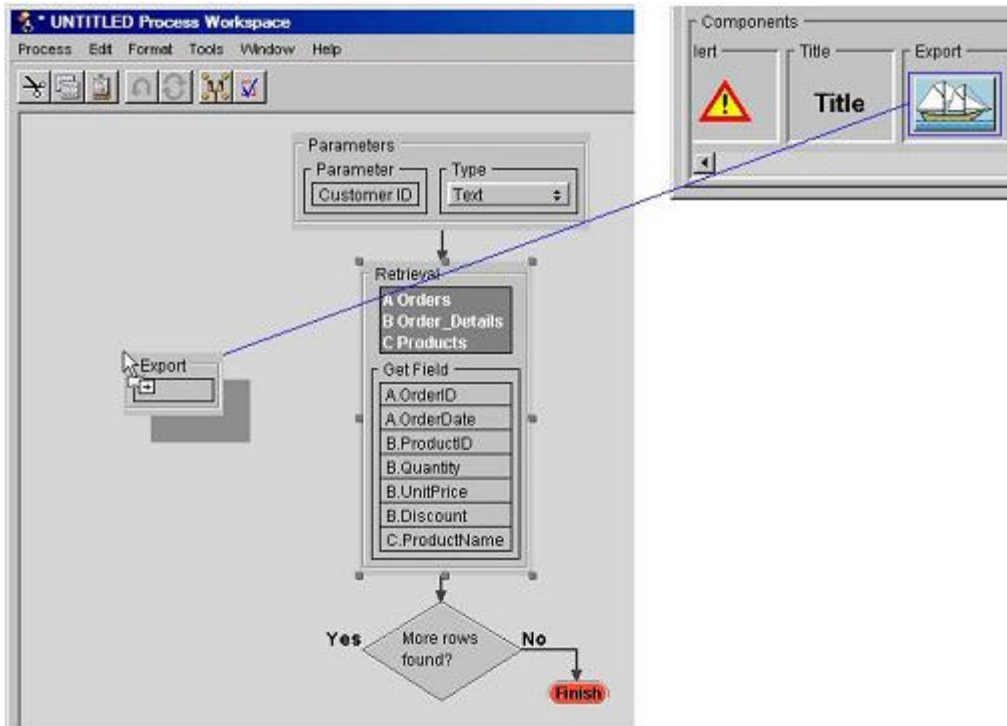
Tutorial 16 - Exporting data into a file

Scribe can save extracted data into an ASCII file. In this Tutorial we will create a Process that retrieves the Order data for the selected Customer and saves it as comma-delimited fields in a file that could be used for further processing outside of Scribe, e.g. for importing data into a spreadsheet. In our Retrieval component, we join three tables: Orders, Order_Details and Products to get the required data into the Process Procedure, as in the following figure:

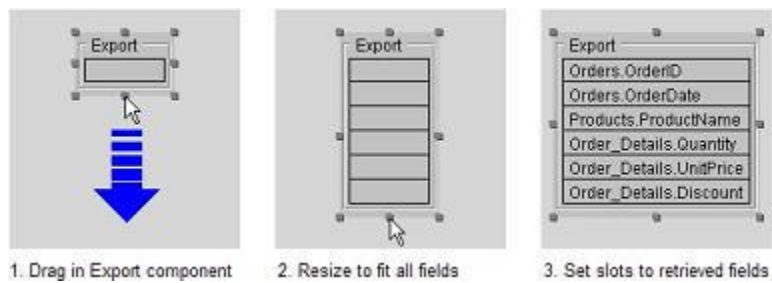


This Retrieval is no different from the one created in the Tutorial 6, except that here we are joining not four, but three tables. The join conditions in this case are created automatically, and do not need any manual refinement.

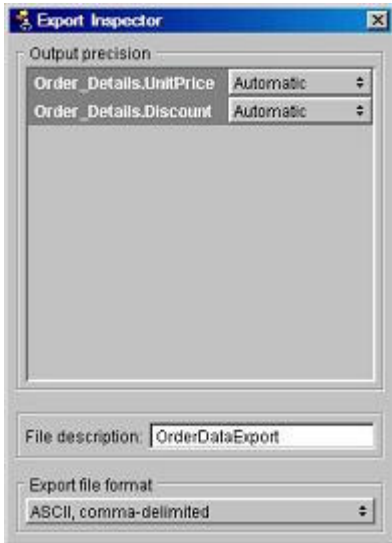
In order to export the retrieved data into a file, we drag in the [Export](#) component from the Palette:



Next, we will resize the Export component to the same number of slots as the number of Retrieval component fields, and populate it with the retrieved fields by either clicking on them in the Palette, or dragging them into the Export slots:



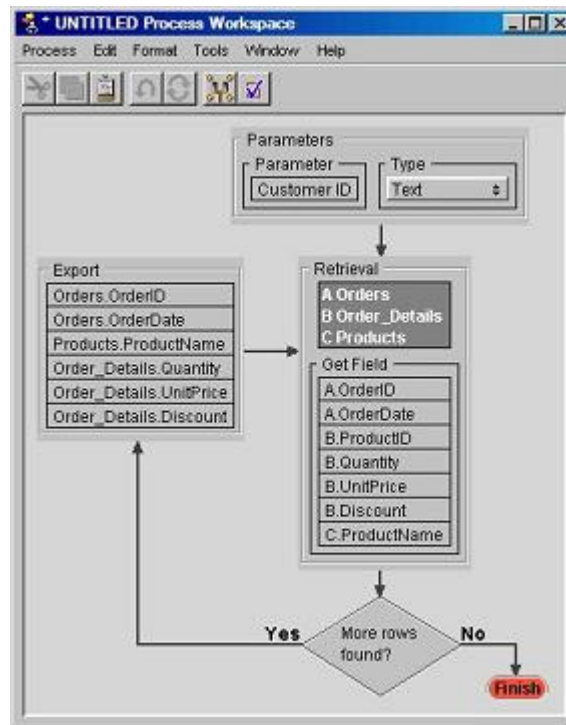
The Export component has a number of parameters controlled by the Inspector. These parameters include:



| Parameter | Description |
|--------------------|--|
| Output precision | Number of decimal places to show in the floating-point values. Two settings are possible: <ul style="list-style-type: none"> Automatic (the number of decimal places is derived from the database table, or from the procedure context) Fixed (the number of decimal places is established in the Inspector) |
| File description | Name that identifies the exported data after running a background job. This name is shown in the Saved Reports (Processes) window, and can be used in saving the data to the Client-side file. |
| Export file format | Choice of the delimited character for field separation: blank space or comma. |

Note: the "Output precision" view of the Inspector is populated by the fields of floating point type (i.e. the ones that have a decimal part). Initially, while the Export component is being constructed, the data type of the fields contained in it may not be known to the Scribe syntax checking mechanism. Consequently, the "Output precision" view remains empty until the first *successful* syntax check is run on the procedure.

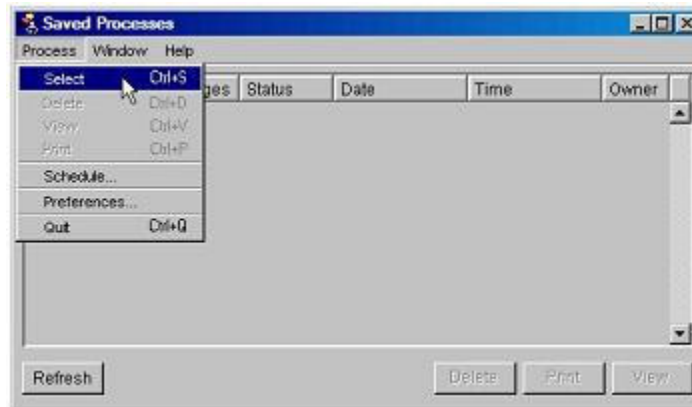
The complete procedure that retrieves Order data and exports it into the file is shown below:



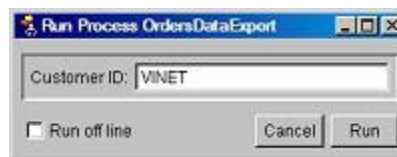
We are now ready to save and run the Process. The Process is saved as "OrderDataExport" (do not forget to check "Selectable as a Process" box in the Legend Inspector); we will try running it first in the foreground, and then in the background as a stand-alone job.

1. Running OrderDataExport process in the foreground

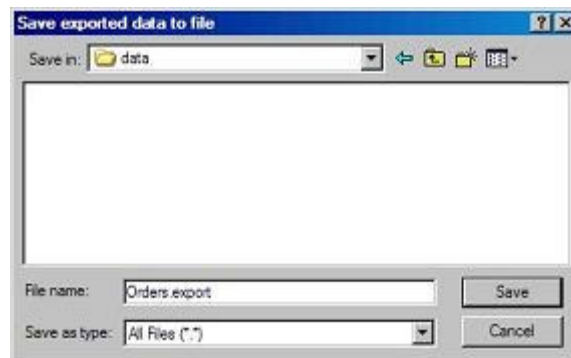
Select Run Process entry in the Application Launchpad window, and press the Launch button; this will bring up the Saved Processes window:



After selecting the OrderDataExport entry from the Process Selector panel, the Process submission window appears, with one input field for the Customer ID parameter:

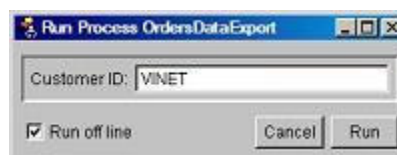


Once the Process completes, Scribe asks the user to specify the name and location of the exported file in a File Dialog panel:

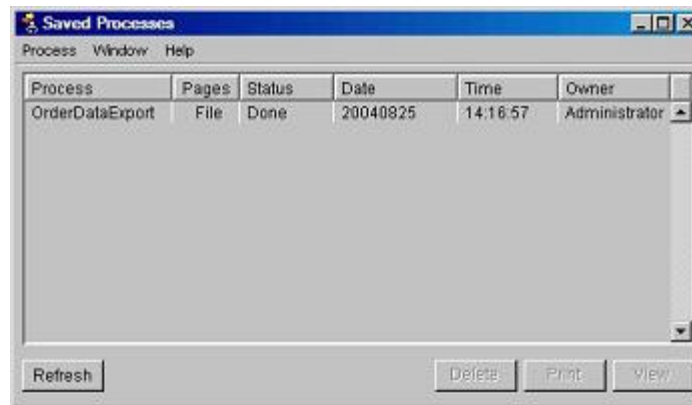


2. Running OrderDataExport process in the background

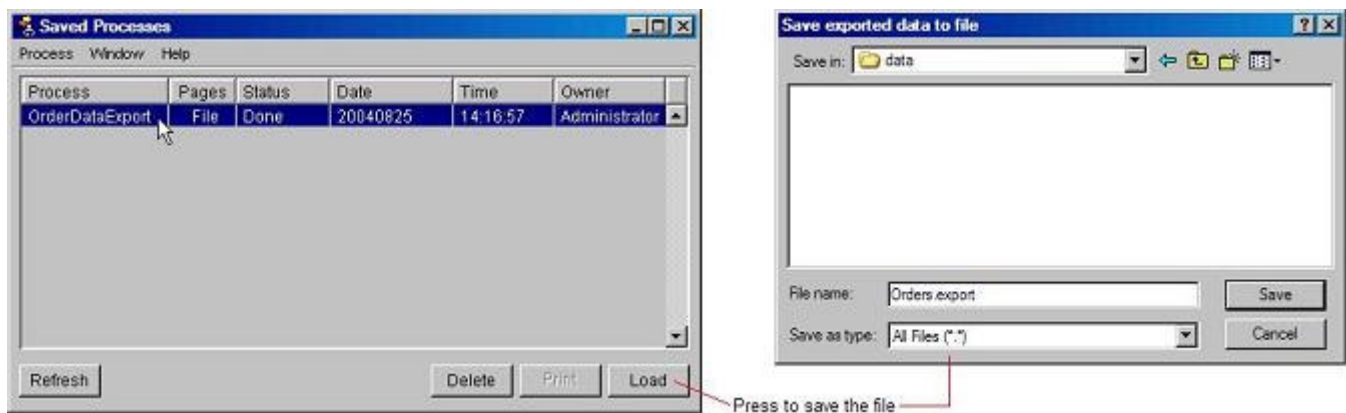
To run the process in the background, we repeat the same steps as in the previous section, but this time we check the "Run off line" box in the Process submission window:



The process will be launched in the background, and after completion the Saved Processes window will have this entry:



The window shows the "OrderDataExport" item as a file, but in reality, at this point, the data exists as a BLOB in the SF_SYSReports table, in Scribe's configuration schema (repository). To load the file on the Client's computer, click anywhere on the "OrderDataExport" row, press the Load button and select the name and location of the file, as in the previous section:



Output data formats

When the OrderDataExport process is run against Northwind sample database hosted by McKoi, the following file is produced:

```
10248,1996-07-04 00:00:00.0,"Queso Cabrales",12,14.000000,0.0000000000000000
10737,1997-11-11 00:00:00.0,"Konbu",4,6.000000,0.0000000000000000
10739,1997-11-12 00:00:00.0,"Inlagd Sill",6,19.000000,0.0000000000000000
10737,1997-11-11 00:00:00.0,"Jack's New England Clam
Chowder",12,9.650000,0.0000000000000000
10248,1996-07-04 00:00:00.0,"Singaporean Hokkien Fried
Mee",10,9.800000,0.0000000000000000
10739,1997-11-12 00:00:00.0,"Filo Mix",18,7.000000,0.0000000000000000
10295,1996-09-02 00:00:00.0,"Gnocchi di nonna Alice",4,30.400000,0.0000000000000000
10274,1996-08-06 00:00:00.0,"Flotemysost",20,17.200000,0.0000000000000000
10248,1996-07-04 00:00:00.0,"Mozzarella di Giovanni",5,34.800000,0.0000000000000000
10274,1996-08-06 00:00:00.0,"Mozzarella di Giovanni",7,27.800000,0.0000000000000000
```

A number of observations can be made on the content of the exported data.

1. The Order.OrderDate field (second from the left in the record) is formatted as YYYY-MM-DD HH:MM:SS.s, which is a standard format for DATE data types returned by the Retrieval component. It may be desirable to have it as YYYYMMDD, or some other date-only format, free of the time component.

2. The Order_Details.UnitPrice field (second from the right in the record) has six decimal places. This happens because the original UnitPrice field in Order_Details table has the data type of DECIMAL(2) - in other words, the decimal part is not set, and Scribe has to use the default precision of 6. This precision flows through to the Export component fields unchanged, according to the "Automatic" setting in the Export Inspector.

3. The Order_Details.Discount field (the rightmost field in the record) has 15 decimal places. The original Discount field in Order_Details table has the data type of REAL, with no decimal places set explicitly, similar to the UnitPrice field. In this case, Scribe uses the default precision of 15, and it too is passed to the corresponding Export component field "as is", for the same reason.

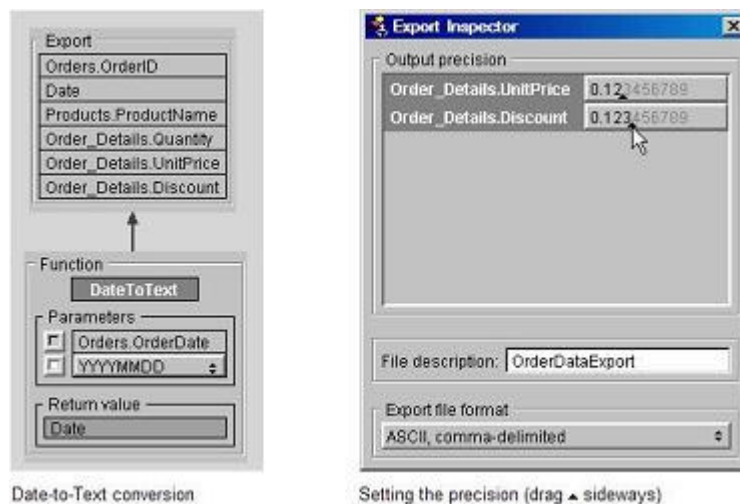
A differently configured source database, with different field data types, could have produced better-formatted results, but Scribe can handle this situation using its own means:

Date formatting

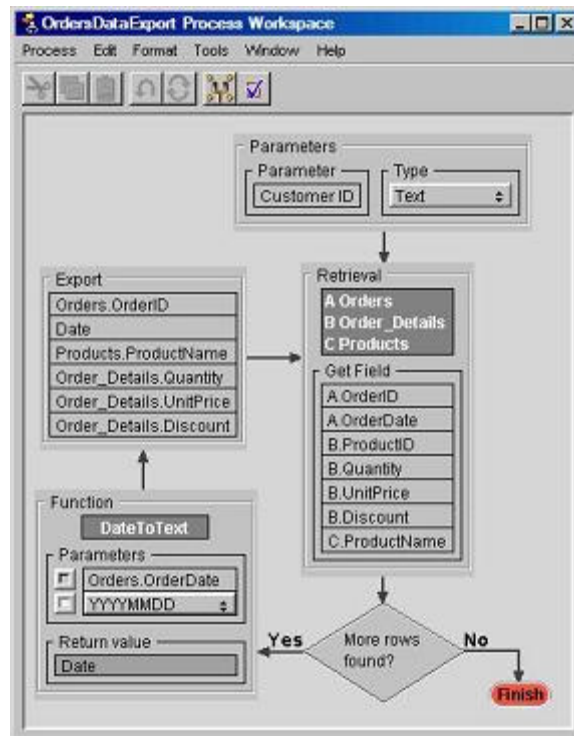
This can be done using the system function DateToText - the same function that was used in the Tutorial 8. This function can take the date value in the standard database format, and convert it into a text string with the formatting of the user's choice.

Numeric data precision

The Export Inspector provides for an explicit, manual setting of the number of decimal places to be shown in the exported data. To do this, click on the precision pop-up list, and change the "Automatic" option to the one that reads "0.123456789".



The complete procedure for the modified Process is shown below:



The new file contents is formatted like this:

```

10248,"19960704","Queso Cabrales",12,14.00,0.000
10737,"19971111","Konbu",4,6.00,0.000
10739,"19971112","Inlagd Sill",6,19.00,0.000
10737,"19971111","Jack's New England Clam Chowder",12,9.65,0.000
10248,"19960704","Singaporean Hokkien Fried Mee",10,9.80,0.000
10739,"19971112","Filo Mix",18,7.00,0.000
10295,"19960902","Gnocchi di nonna Alice",4,30.40,0.000
10274,"19960806","Flotemysost",20,17.20,0.000
10248,"19960704","Mozzarella di Giovanni",5,34.80,0.000
10274,"19960806","Mozzarella di Giovanni",7,27.80,0.000

```