



Using Replicas

RadExPro 2018.1

Replicas – are copies of one and the same flow executed with different sets of module parameters. Sets of parameters for each replica are taken from variables defined in a dedicated replica table.

Replica table – is a new type of database objects. Each column of such a table corresponds to a named variable. Each row corresponds to an individual replica (a copy of a flow). Each replica would use a set of variable values specified in a separate row of the table.

A flow with variables in module parameters is called a *Template Flow*. When a template flow is executed, variables are substituted by specific values taken from related replica table.

A typical use case is standard processing of a set of lines. Now you can create common template flows to be used for the processing, while individual parameters for each line (e.g. line name, SOL shot, EOL shot, etc.) are pre-defined in a replica table.

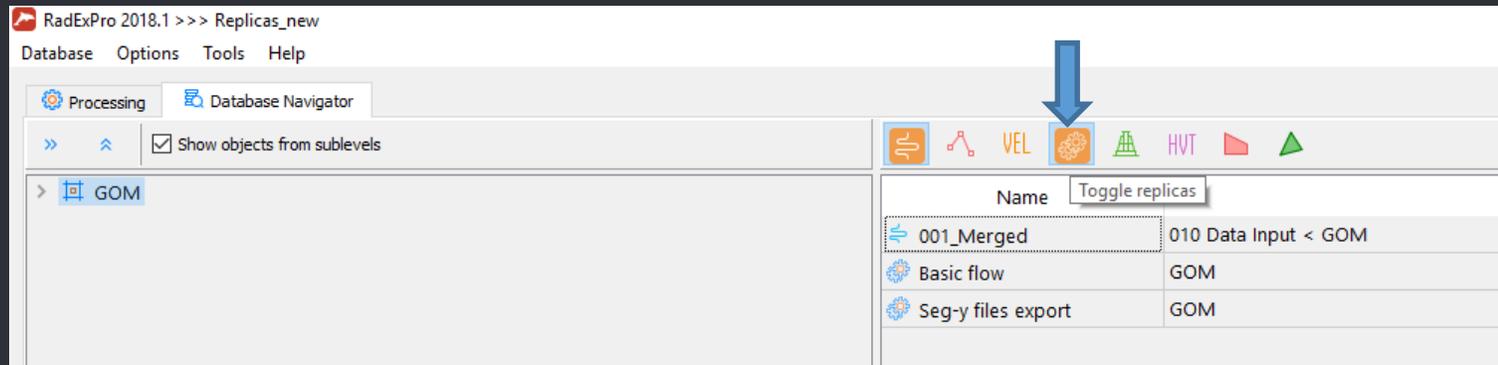
List of modules supporting replica variables in RadExPro 2018.1:

- Seg-d Input –input file list
- Seg-y Input –input file list
- Seg-y Output –output file name and EBCDIC editor
- Trace Input –input dataset list and Selection field
- Trace Output – input dataset list
- Import SPS – input file list
- Import P1-90 – input file name
- Trace Header Math
- Data Filter
- Header<->Dataset Transfer -- dataset name

The list is to be extended in the future.

1. Creating replica table

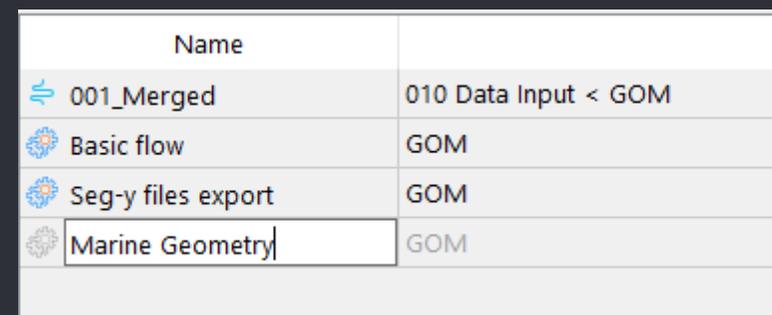
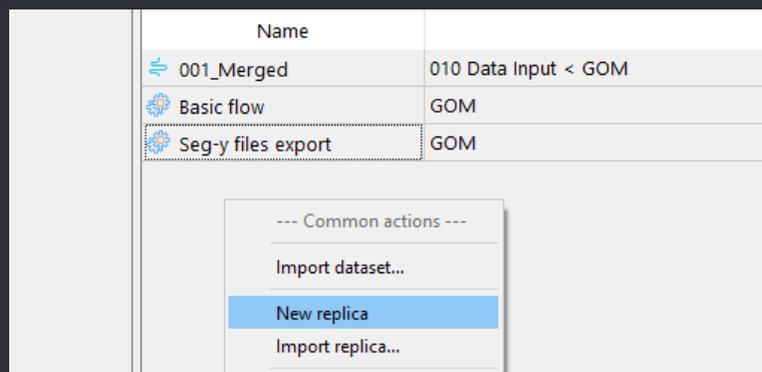
Switch to Database Navigator tab and switch one replica tables display – *Toggle replicas* toolbar button



Right mouse click in the list of objects and select New replica entry of the context menu.

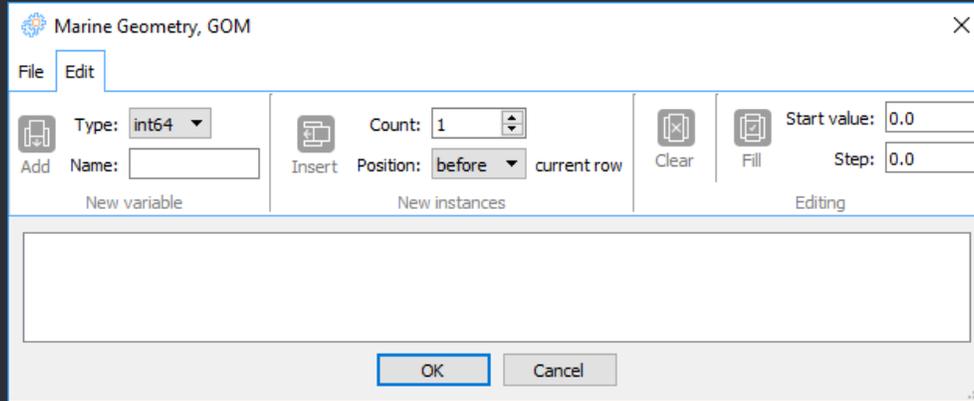
Specify a name of the new replica table – here we will call it Marine Geometry.

While the new table is empty its name is displayed in gray.

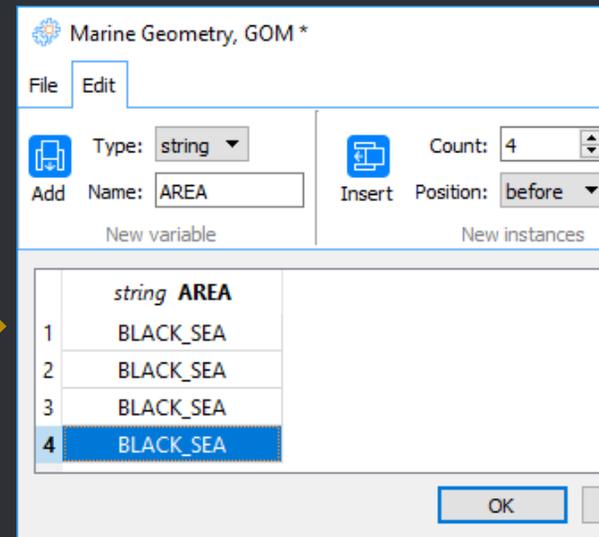
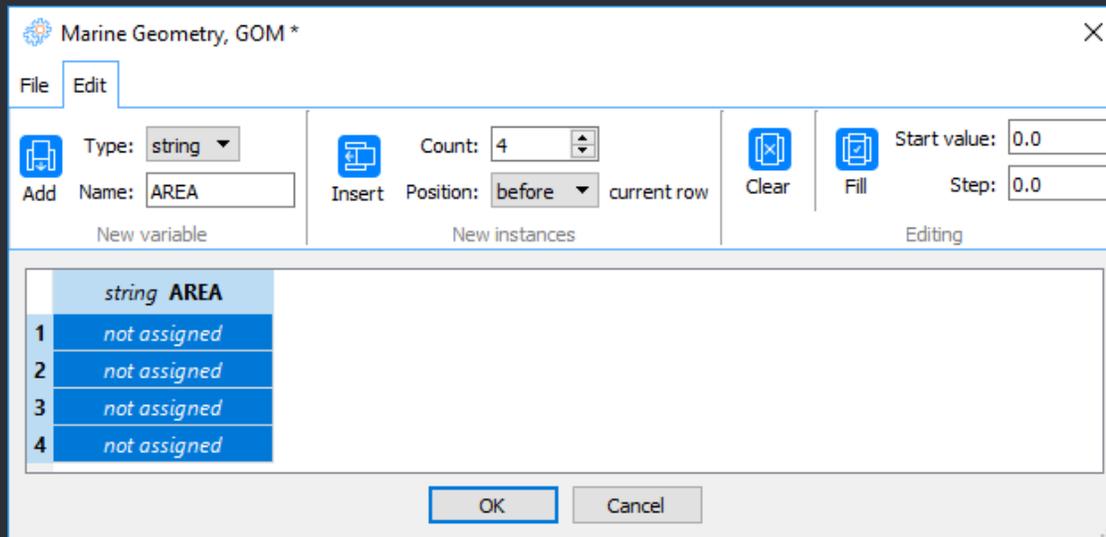


2. Filling replica table in

Double click the just created table to open it for editing:



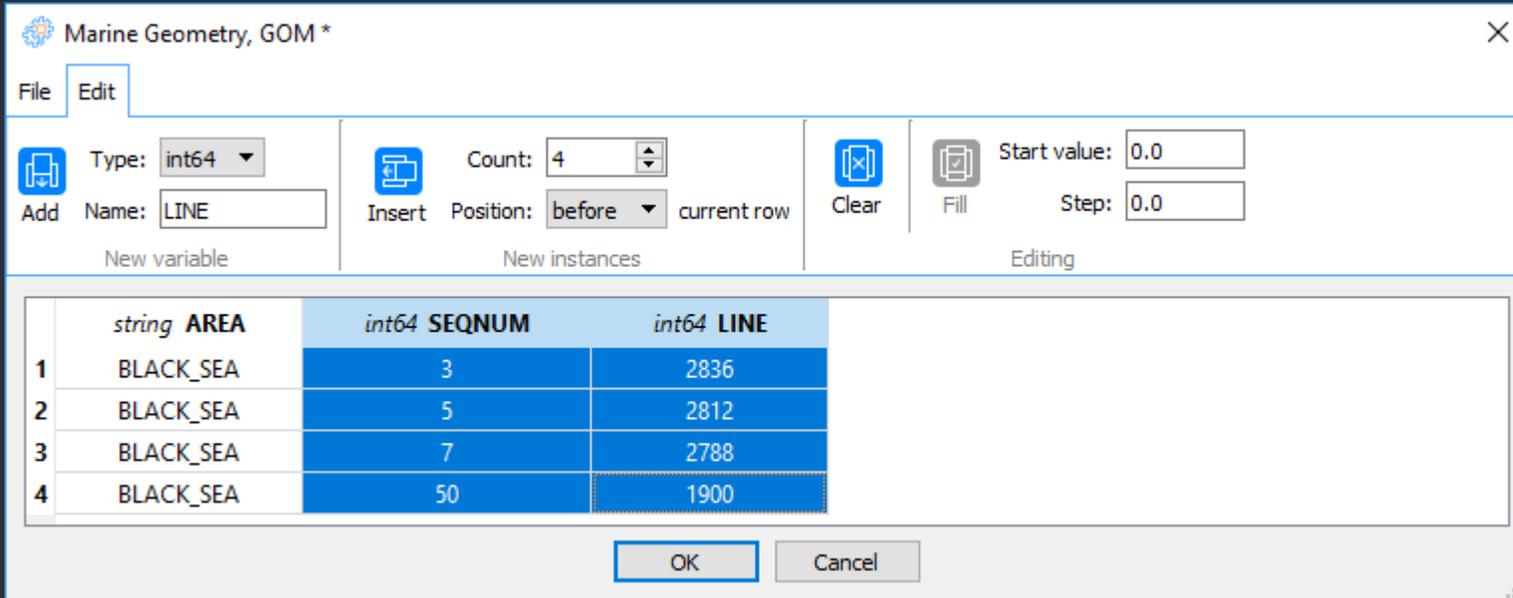
Let's add a variable with an area name. Specify variable's name -- AREA, and type -- *string*, indicate the number of rows required (4) and click the *Add* button. Assign a specific area name to all cells of the AREA column. In this example, the name will be the same -- BLACK SEA.



2. Filling replica table in

Now fill in sequence and line numbers – they are commonly used for naming of SEG-D and P1-90 files and we are going to use them as variables in I/O module parameters.

For that, we create 2 new variables – SEQNUM and LINE – and fill their cells in as shown on the figure. In this examples we use 4 lines numbered as 3, 5, 7, and 50.



The screenshot shows the 'Marine Geometry, GOM' dialog box with the 'Edit' tab selected. The 'Add' section shows a new variable named 'LINE' of type 'int64'. The 'Insert' section shows 4 instances to be added before the current row. The 'Editing' section shows start and step values of 0.0. Below the controls is a table with 4 rows and 3 columns: 'AREA' (string), 'SEQNUM' (int64), and 'LINE' (int64). The table contains the following data:

	<i>string</i> AREA	<i>int64</i> SEQNUM	<i>int64</i> LINE
1	BLACK_SEA	3	2836
2	BLACK_SEA	5	2812
3	BLACK_SEA	7	2788
4	BLACK_SEA	50	1900

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

2. Filling replica table in

Add the following variables to the table and fill them with values:

SOL_SHOT – first good shot of a line

EOL_SHOT – last good shot of a line

STATUS – line status (Primary, Infill)

DAY, MONTH, YEAR

BAD_SHOT

Save the table using File/Save command. Now we can use the variables of this table in template flows.

Basic flow, GOM

File Edit

Add Type: int64 Name:

Insert Count: 1 Position: before current row

Clear Fill Start value: 0.0 Step: 0.0

	string AREA	int64 SEQNUM	int64 LINE	int64 SOL_SHOT	int64 EOL_SHOT	string STATUS	int64 DAY	int64 MONTH	int64 YEAR	int64 BAD_SHOT
1	GOM	3	2836	1	101	P1	21	3	2018	50
2	GOM	5	2812	15	97	I1	21	3	2018	80
3	GOM	7	2788	3	105	P2	22	3	2018	90
4	GOM	50	1900	3	104	I1	23	4	2018	45

OK Cancel

3. Using variables in module parameters

General variable syntax:

`{@name}`, where name — is the name of a column from a replica table.

Examples of Trace Header Math formulas:

`S_LINE = {@LINE}`

`offset = (chan — 1) * 25.0 + {@first_channel_offset}`

When a number is converted to a string you may wish to specify the number format. Use extended variable syntax with format specifier for that.

Examples of using format specifiers:

`{@file_no, 06d}` — 6-digit integer number, missed higher number positions are filled with zeroes (resulting strings look like “000001”, “000002”, ..., “000123”, ... etc.)

`{@first_channel_offset, 6.2f}` — 6-digin real number with 2 decimal places, missed higher number positions are filled with spaces (resulting strings look like “ 1.00”, “ 2.50”, “ 123.32”, ... etc.)

Format specifiers are discussed in more detail in the Appendix on the last slide of this presentation.

4. Template flow example

We start from input of SEG-D files. The paths to folders with the files look as following:

Data\SEGD\Seq003_BLACK_SEA5102836\2836

Data\SEGD\Seq005_BLACK_SEA15102812\2812

...etc.

We will use SEG-D Input module to read the files. In order to read all of them at once, instead of a list of specific file names, we are going to use a *selection mask* (use Mask button to add one)



A mask may contain plain text, replica variables, wildcard characters -- *, ?, and *intervals*.

An *interval* $\langle a,b \rangle$ includes all integer numbers starting from a and up to b .

You can also use extended interval syntax $\langle a,b/d \rangle$ -- here d is a format specifier (only integer formats are allowed here, see Appendix at the last slide).

For instance, interval $\langle 1,3|03d \rangle$ will be converted to sequence of the following strings: "001", "002", "003".

4. Template flow example

So, we were going to read all data files from the folders like these:

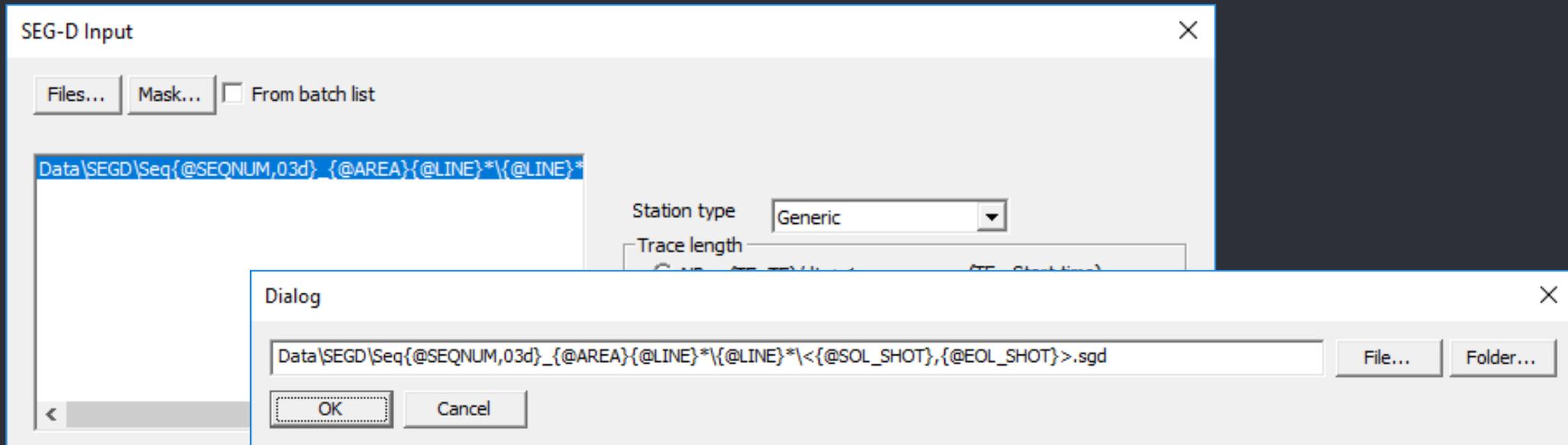
Data\SEGD\Seq003_BLACK_SEA5102836\2836

Data\SEGD\Seq005_BLACK_SEA15102812\2812

... etc.

In this case, we may use the following selection mask:

Data\SEGD\Seq{@SEQNUM,03d}_{@AREA}{@LINE}*{@LINE}*\<{@SOL_SHOT},{@EOL_SHOT}>.sgd



Let us discuss selection mask in more detail:

Data\SEGD\Seq{@SEQNUM,03d}_{@AREA}{@LINE}* \{@LINE}* \<{@SOL_SHOT},{@EOL_SHOT}>.sgd

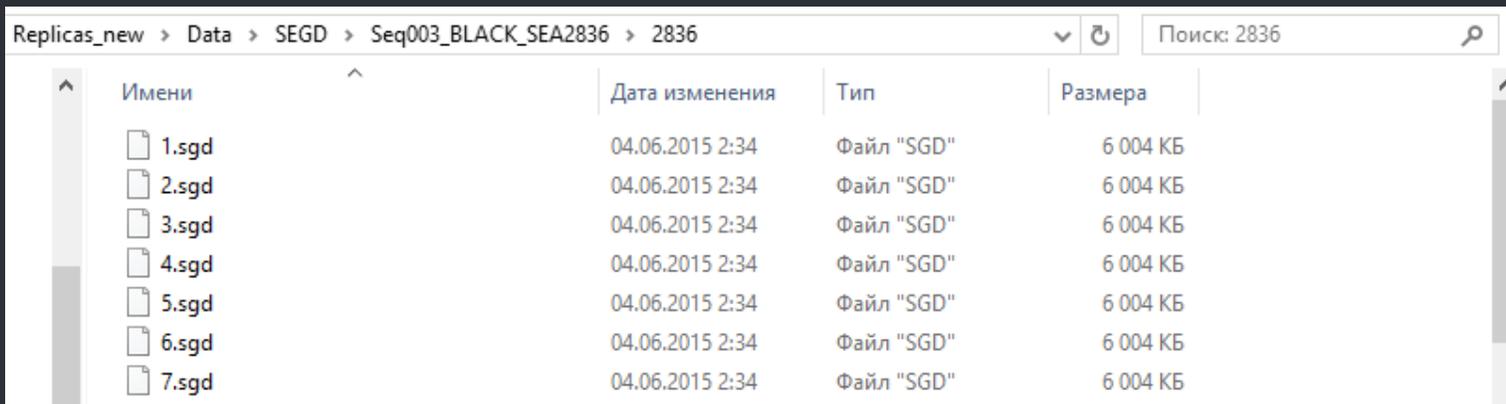
Seq{@SEQNUM,03d}_{@AREA}{@LINE}* -- defines folder names:

Seq003_BLACK_SEA2836	01.03.2018 11:25	Папка с файлами
Seq005_BLACK_SEA2812	01.03.2018 11:25	Папка с файлами
Seq007_BLACK_SEA2788	01.03.2018 11:25	Папка с файлами
Seq050_BLACK_SEA1900I	05.03.2018 14:03	Папка с файлами

	string AREA	int64 SEQNUM	int64 LINE
1	BLACK_SEA	3	2836
2	BLACK_SEA	5	2812
3	BLACK_SEA	7	2788
4	BLACK_SEA	50	1900

{@LINE}* - defines a subfolder with the line name, here * is a wildcard character that allows any additional characters at the end of the line name, to accommodate cases like "1900I".

<{@SOL_SHOT},{@EOL_SHOT}>.sgd – define files names based on shot interval from the replica table



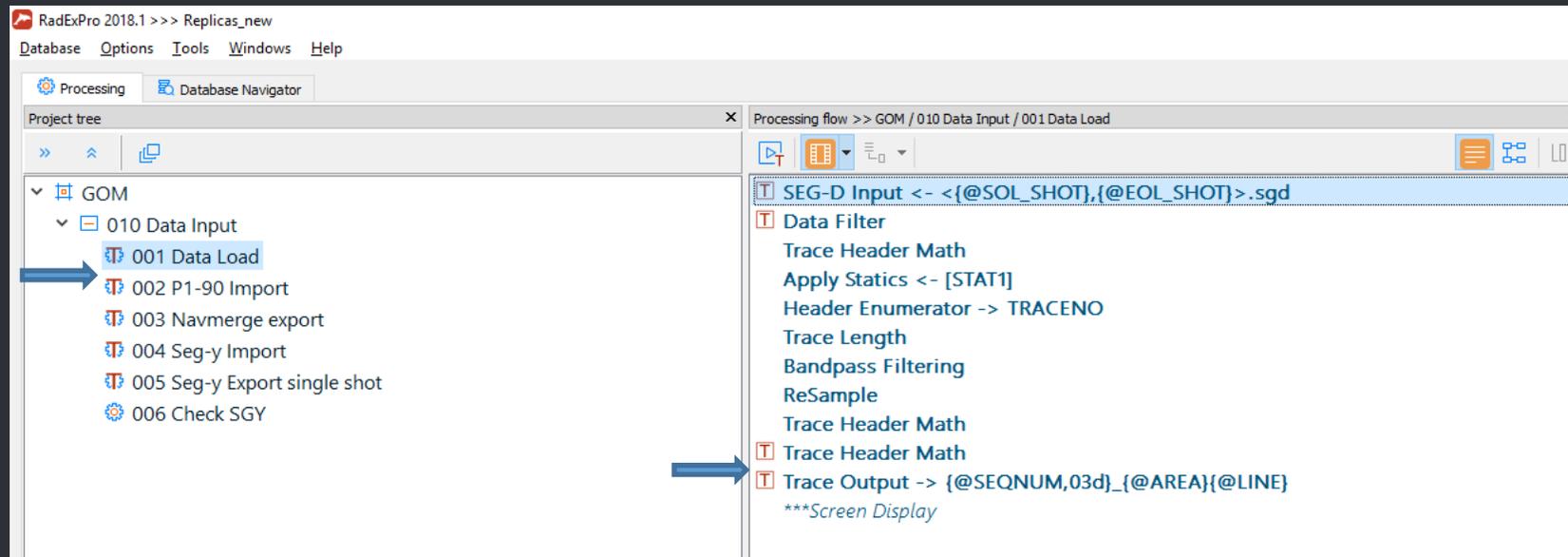
Имени	Дата изменения	Тип	Размера
1.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
2.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
3.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
4.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
5.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
6.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ
7.sgd	04.06.2015 2:34	Файл "SGD"	6 004 КБ

int64 SOL_SHOT	int64 EOL_SHOT
1	101
15	97
3	105
3	104

4. Template flow example

When a module in the flow uses variables in its parameters, it is considered as a *template* and is marked in the flow editor with T icon.

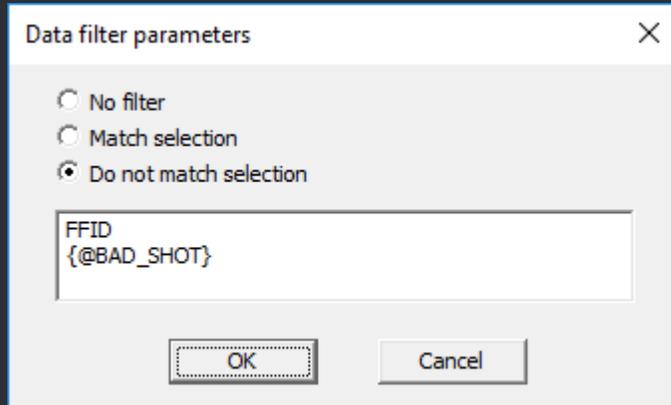
The whole flow is considered as a *template* when it contains at list one template module. Template flows are marked with T icon in the project tree.



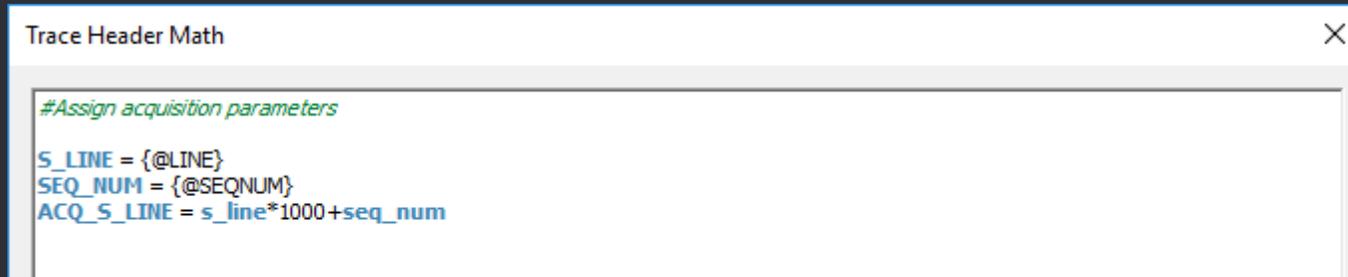
4. Template flow example

Here are parameters of other template modules using replica variables in the example template flow:

Data Filter – does not let the bad shot into the flow

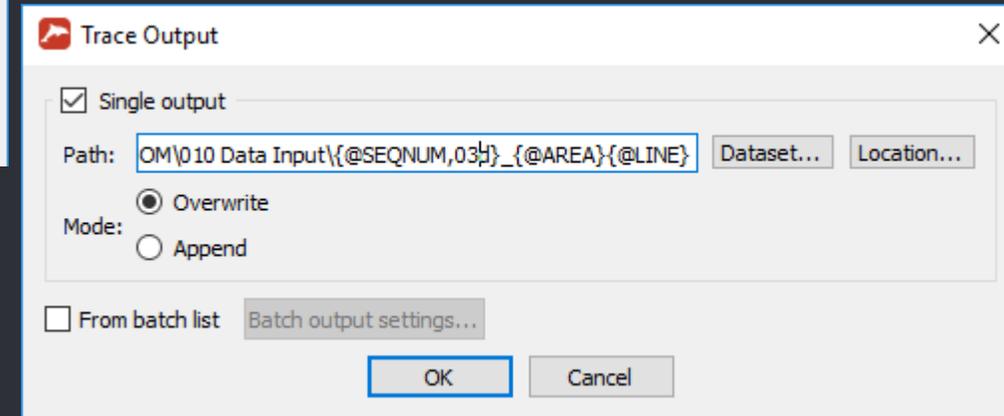


Trace Header Math – assigns S_LINE, SEQ_NUM headers



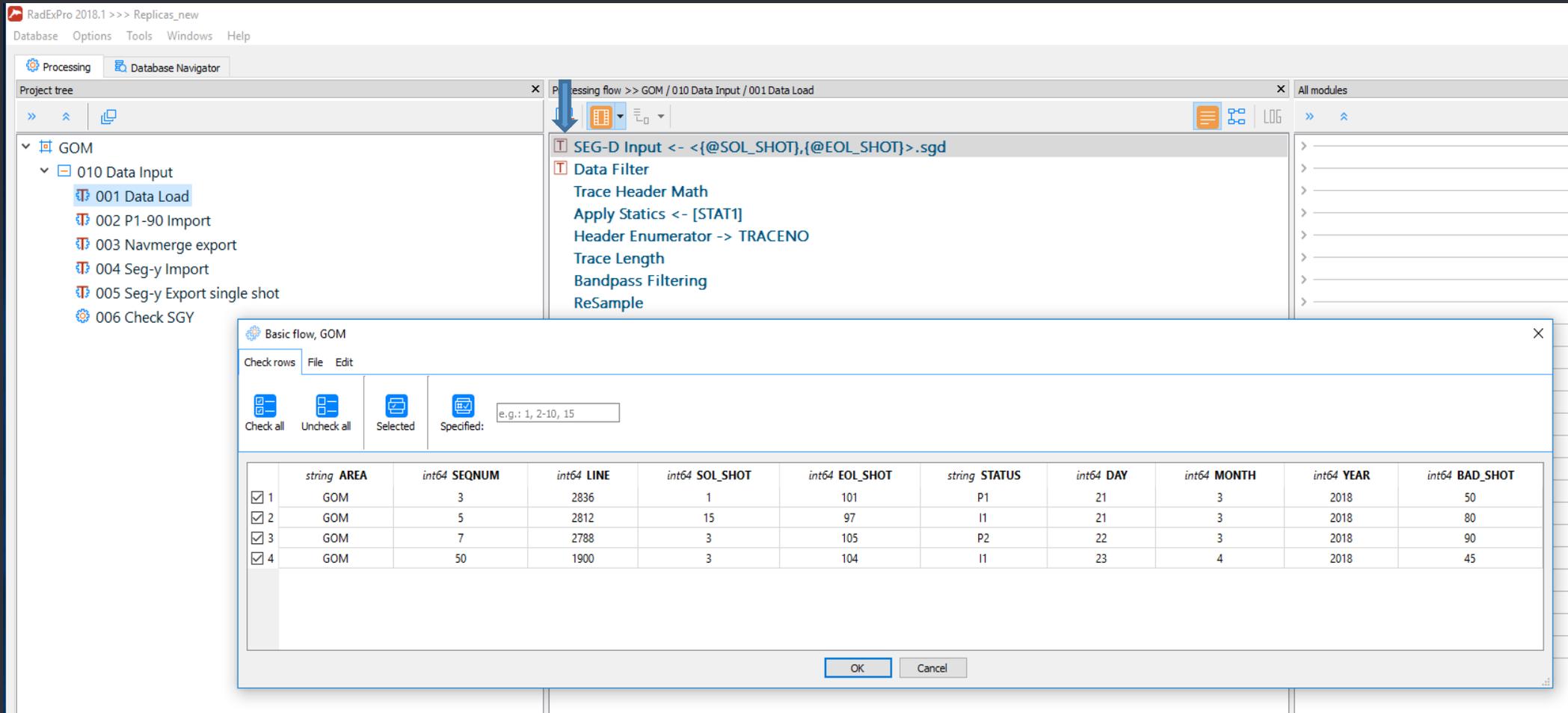
Trace Output – the output dataset name defined as following:

GOM\010 Data Input\{@SEQNUM,03d}\{@AREA}\{@LINE}



5. Executing a template flow

As our flow is a template, it cannot be executed without a reference to a replica table. When you click the Run button, you are prompted to select a table from the project database and check specific rows within the table to be used for flow replicas:

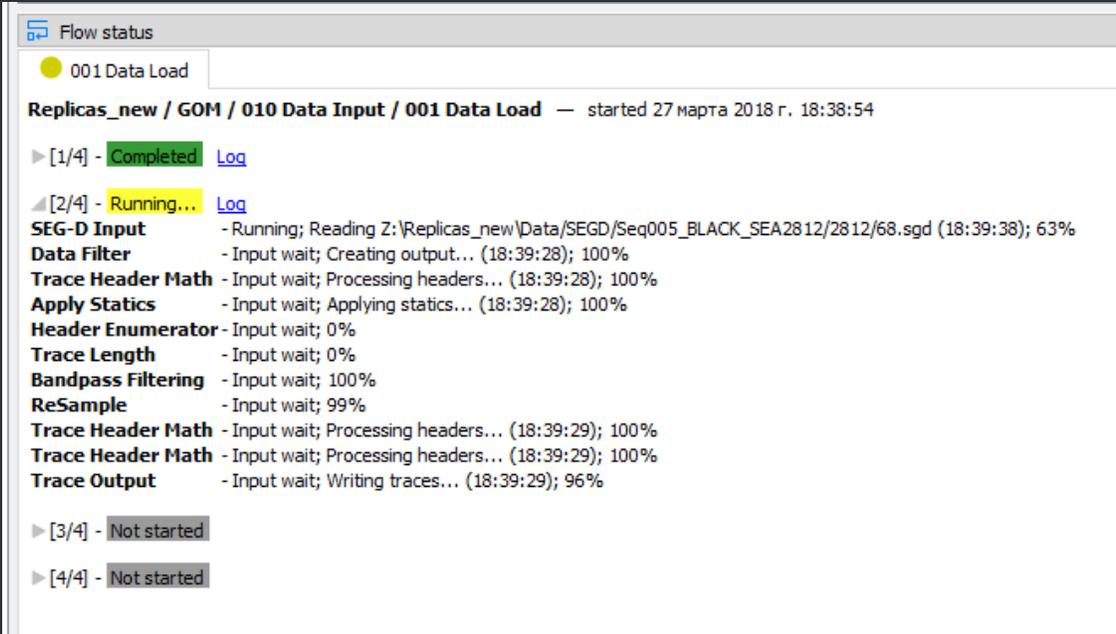


The screenshot shows the RadExPro 2018.1 interface with a 'Basic flow, GOM' dialog box open. The dialog box has a 'Check rows' tab and a table of data. The table has columns for AREA, SEQNUM, LINE, SOL_SHOT, EOL_SHOT, STATUS, DAY, MONTH, YEAR, and BAD_SHOT. The first four rows are checked, indicating they are selected for execution.

	string AREA	int64 SEQNUM	int64 LINE	int64 SOL_SHOT	int64 EOL_SHOT	string STATUS	int64 DAY	int64 MONTH	int64 YEAR	int64 BAD_SHOT	
<input checked="" type="checkbox"/>	1	GOM	3	2836	1	101	P1	21	3	2018	50
<input checked="" type="checkbox"/>	2	GOM	5	2812	15	97	I1	21	3	2018	80
<input checked="" type="checkbox"/>	3	GOM	7	2788	3	105	P2	22	3	2018	90
<input checked="" type="checkbox"/>	4	GOM	50	1900	3	104	I1	23	4	2018	45

5. Executing a template flow

After a replica table is selected and the rows are checked, an individual replica of the template flow will run for each of the rows. Status of replica flows execution is shown as an extendable list:



Flow status

● 001 Data Load

Replicas_new / GOM / 010 Data Input / 001 Data Load — started 27 марта 2018 г. 18:38:54

▶ [1/4] - Completed [Log](#)

▲ [2/4] - Running... [Log](#)

SEG-D Input - Running; Reading Z:\Replicas_new\Data\SEGD\Seq005_BLACK_SEA2812\2812\68.sgd (18:39:38); 63%

Data Filter - Input wait; Creating output... (18:39:28); 100%

Trace Header Math - Input wait; Processing headers... (18:39:28); 100%

Apply Statics - Input wait; Applying statics... (18:39:28); 100%

Header Enumerator - Input wait; 0%

Trace Length - Input wait; 0%

Bandpass Filtering - Input wait; 100%

ReSample - Input wait; 99%

Trace Header Math - Input wait; Processing headers... (18:39:29); 100%

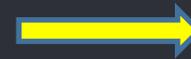
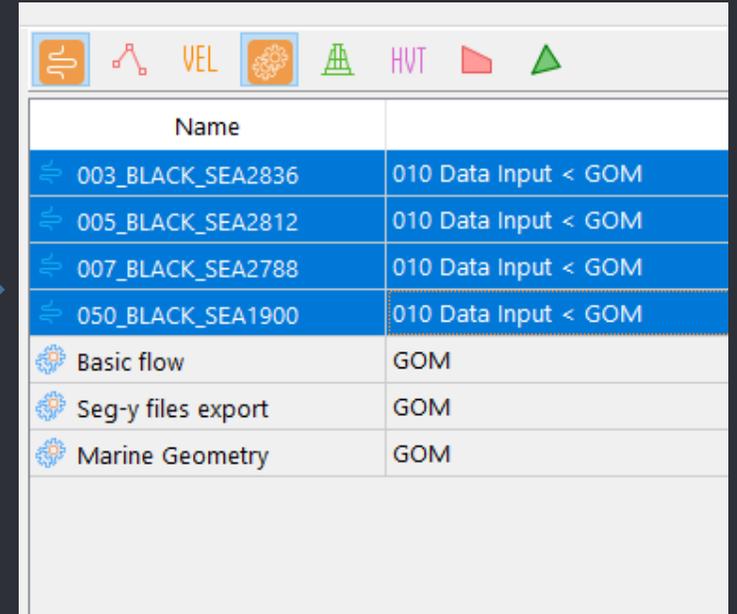
Trace Header Math - Input wait; Processing headers... (18:39:29); 100%

Trace Output - Input wait; Writing traces... (18:39:29); 96%

▶ [3/4] - Not started

▶ [4/4] - Not started

Result of execution of replicas of this template flow – 4 output datasets

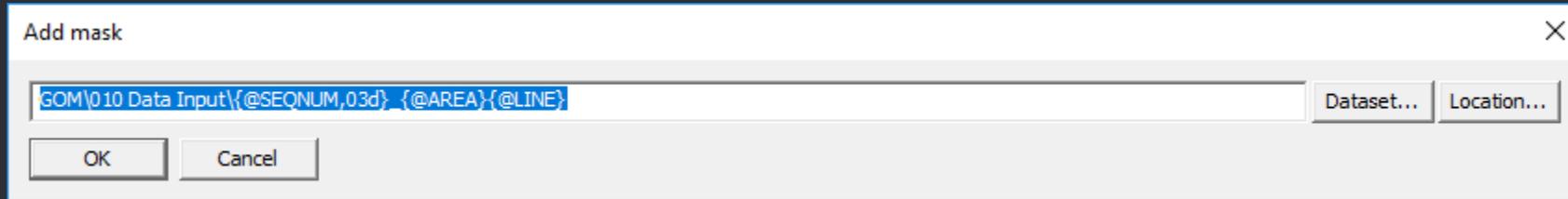



Name	
003_BLACK_SEA2836	010 Data Input < GOM
005_BLACK_SEA2812	010 Data Input < GOM
007_BLACK_SEA2788	010 Data Input < GOM
050_BLACK_SEA1900	010 Data Input < GOM
Basic flow	GOM
Seg-y files export	GOM
Marine Geometry	GOM

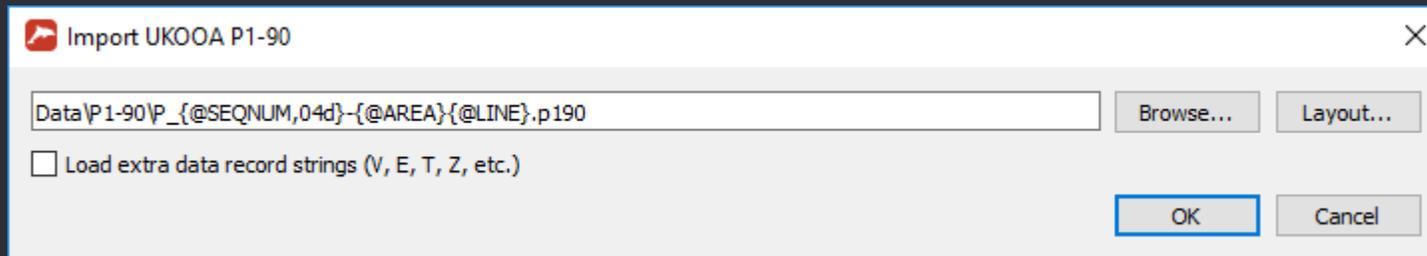
5. Executing several template flows

Let us make one more template flow, here we are going to assign geometry from P1-90 files to the data

In the Trace Input module:



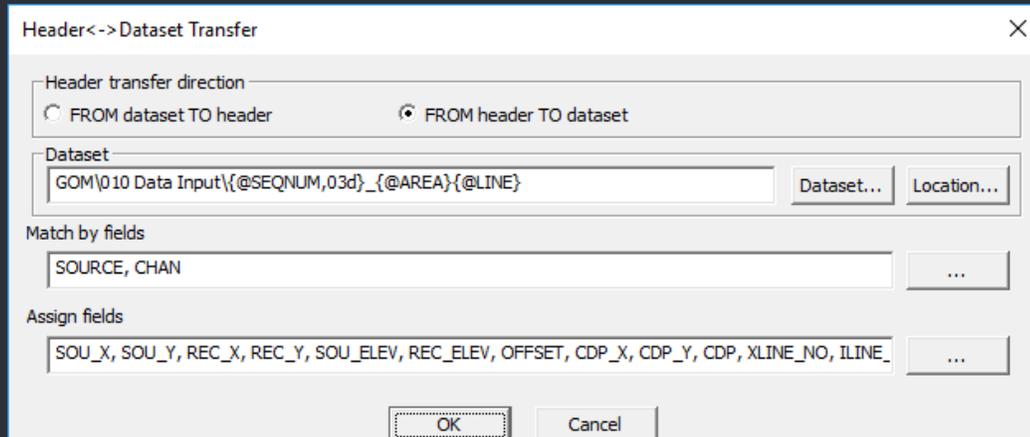
In the Import P1-90 module:



Actual path to the files:

Имени	Дата изменения	Тип
P_0003-MGL15102836.p190	03.09.2015 23:41	Файл "P190"
P_0005-MGL15102812.p190	03.09.2015 23:41	Файл "P190"
P_0007-MGL15102788.p190	03.09.2015 23:41	Файл "P190"
P_0050-MGL15101900.p190	03.09.2015 23:40	Файл "P190"

In the Header<->Dataset Transfer module:



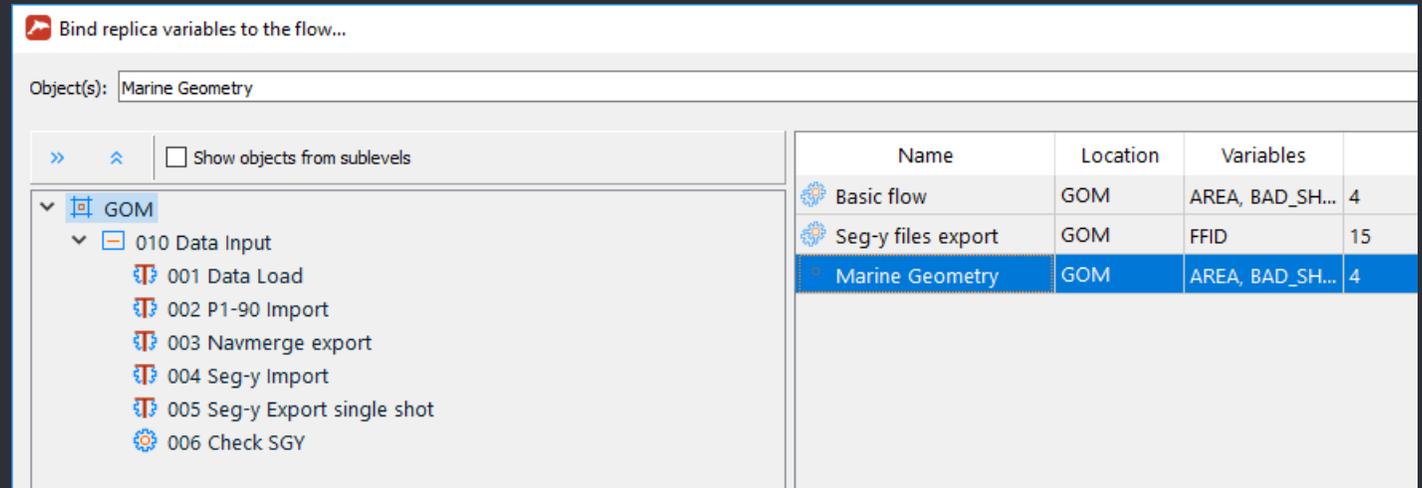
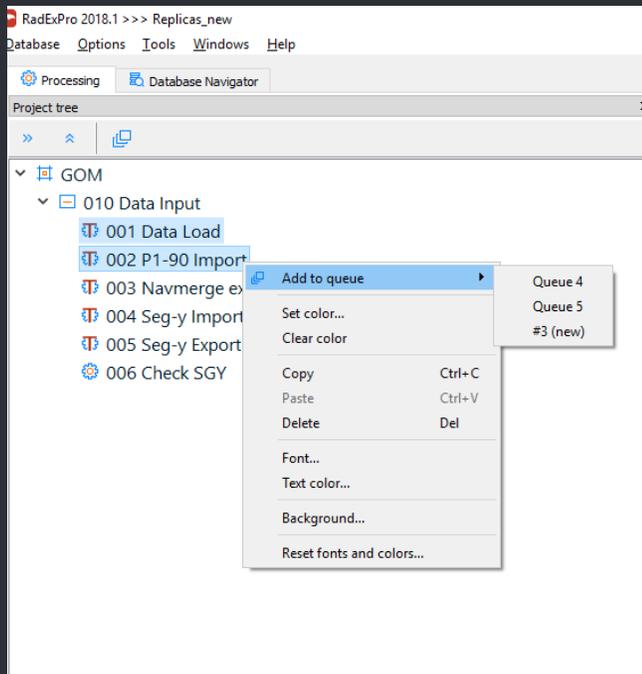
5. Executing several template flows

After a template flow for geometry assignment is created and fine tuned, let us run both template flows at the same time (assume, that we have not yet executed the 1st flow):

001 Data Load

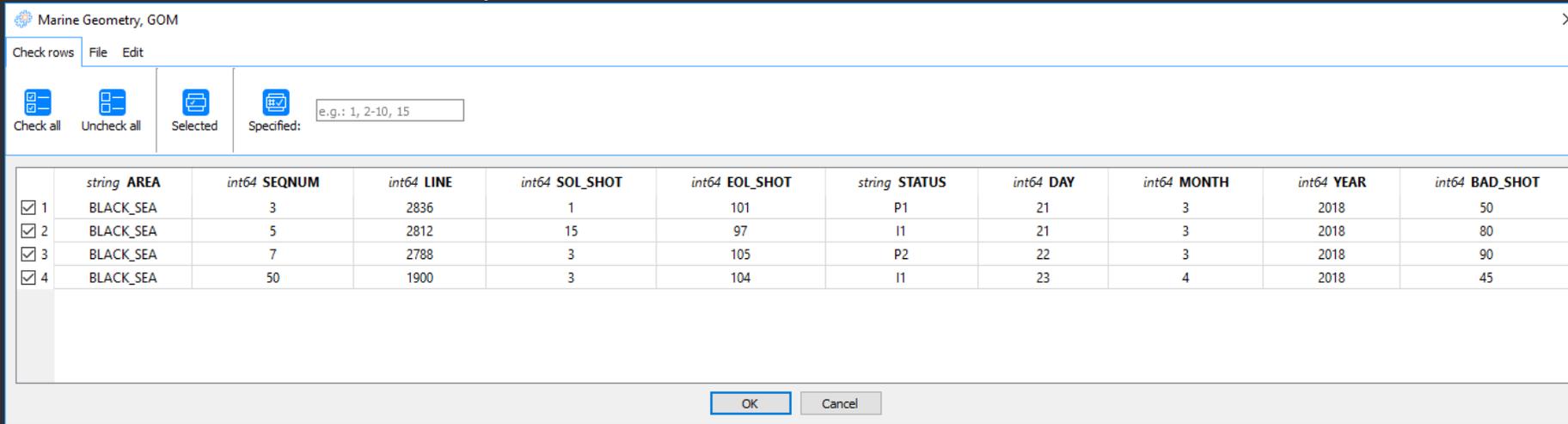
002 P1-90 Import

Select both template flows (e.g. with Ctrl+left mouse click). Right-click to see the pop-up menu and add them to a new queue. Similarly to executing one template flow, you will be prompted to select a replica table.



5. Executing several template flows

When a table is selected it will open on the screen:



Marine Geometry, GOM

Check rows | File | Edit

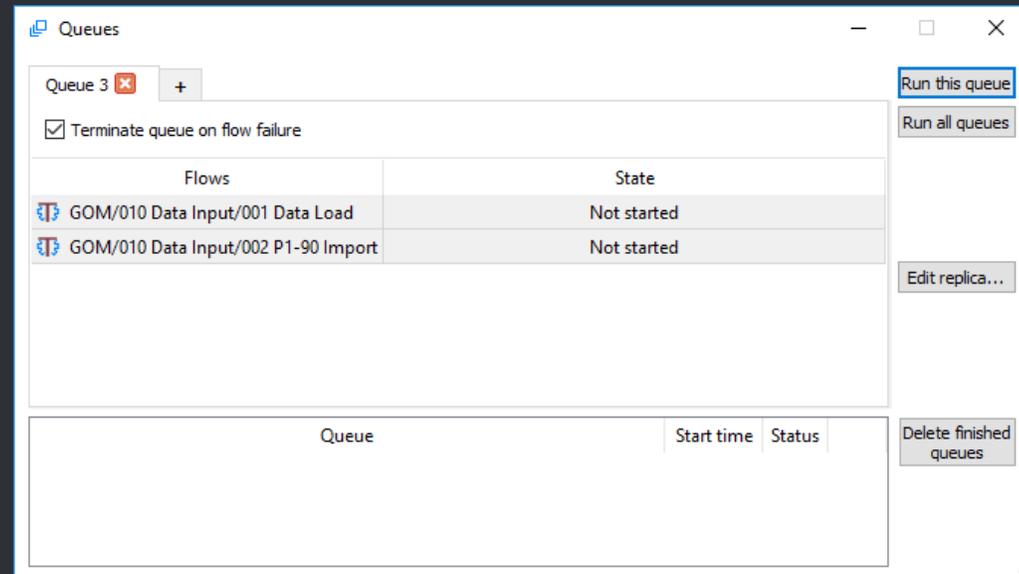
Check all
 Uncheck all
 Selected
 Specified:

	string AREA	int64 SEQNUM	int64 LINE	int64 SOL_SHOT	int64 EOL_SHOT	string STATUS	int64 DAY	int64 MONTH	int64 YEAR	int64 BAD_SHOT
<input checked="" type="checkbox"/> 1	BLACK_SEA	3	2836	1	101	P1	21	3	2018	50
<input checked="" type="checkbox"/> 2	BLACK_SEA	5	2812	15	97	I1	21	3	2018	80
<input checked="" type="checkbox"/> 3	BLACK_SEA	7	2788	3	105	P2	22	3	2018	90
<input checked="" type="checkbox"/> 4	BLACK_SEA	50	1900	3	104	I1	23	4	2018	45

OK Cancel

Check the rows for which you are going to run flow replicas – use checkboxes to the left of the rows

Click OK button, then a standard RadExPro Queues dialog will open. Now you can execute all the replicas using *Run this queue* button



Queues

Queue 3 +

Terminate queue on flow failure

Flows	State
GOM/010 Data Input/001 Data Load	Not started
GOM/010 Data Input/002 P1-90 Import	Not started

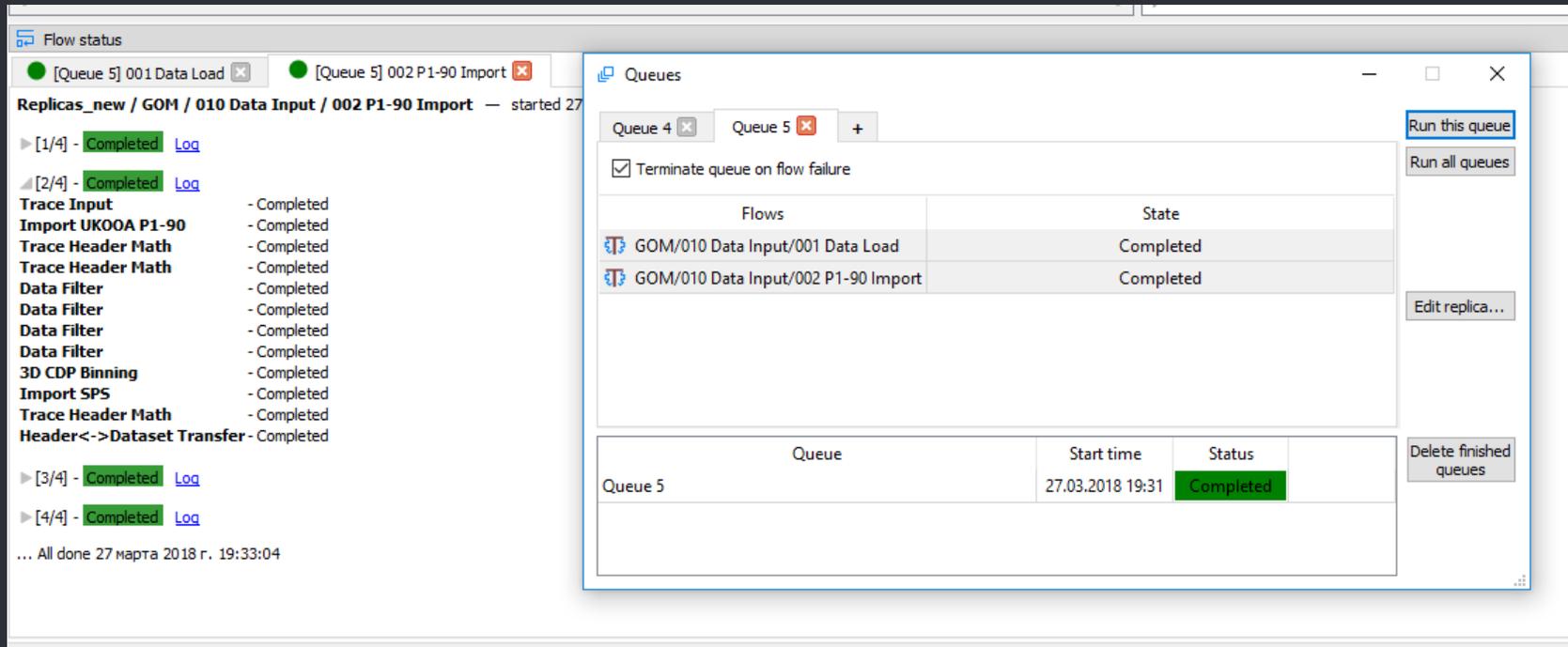
Queue Start time Status

Run this queue
Run all queues
Edit replica...
Delete finished queues

5. Executing several template flows

As a result, 4 replicas of each of the 2 template flows (**001 Data Load** and **002 P1-90 Import**) will be executed: each flow will process 4 lines as indicated in the replica table.

Replicas execution status will be displayed in the *Flow status* window as an extendable list.



The screenshot shows two windows from the software interface. The 'Flow status' window on the left displays a list of tasks for 'Replicas_new / GOM / 010 Data Input / 002 P1-90 Import'. The tasks are grouped into four replicas, each with a 'Completed' status and a 'Log' link. The tasks include Trace Input, Import UKOOA P1-90, Trace Header Math, Data Filter, 3D CDP Binning, Import SPS, and Header Dataset Transfer. The status at the bottom indicates 'All done 27 марта 2018 г. 19:33:04'.

The 'Queues' window on the right shows the execution status of the queues. It includes a table of flows and a table of queues.

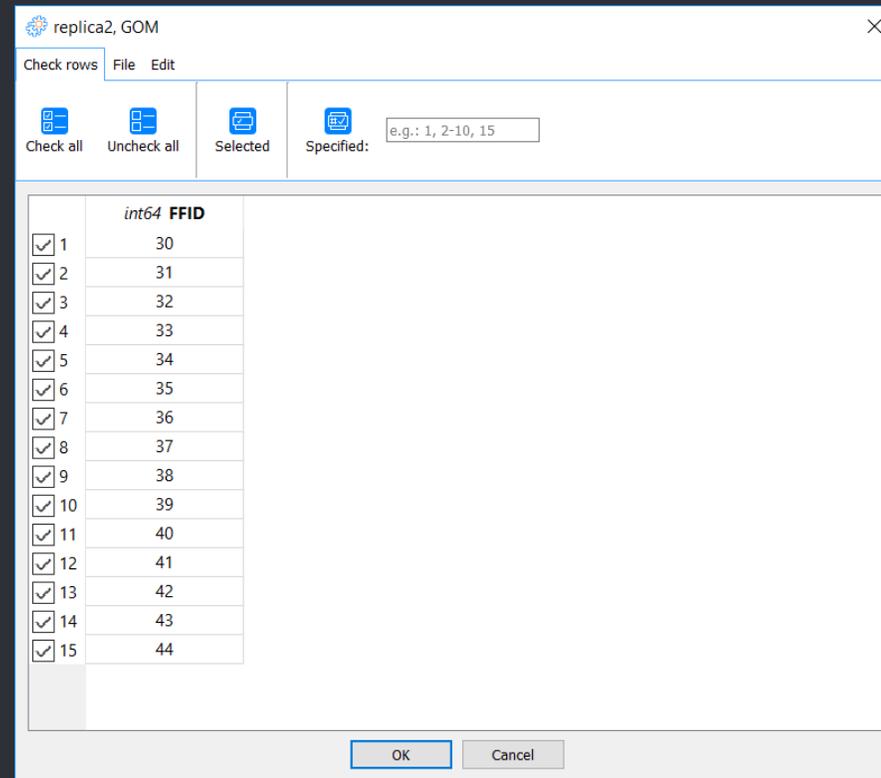
Flows	State
GOM/010 Data Input/001 Data Load	Completed
GOM/010 Data Input/002 P1-90 Import	Completed

Queue	Start time	Status
Queue 5	27.03.2018 19:31	Completed

6. One more example of replica use

The aim is to export an existing dataset to SEG-Y so that each shot is saved to a separate SEG-Y file.

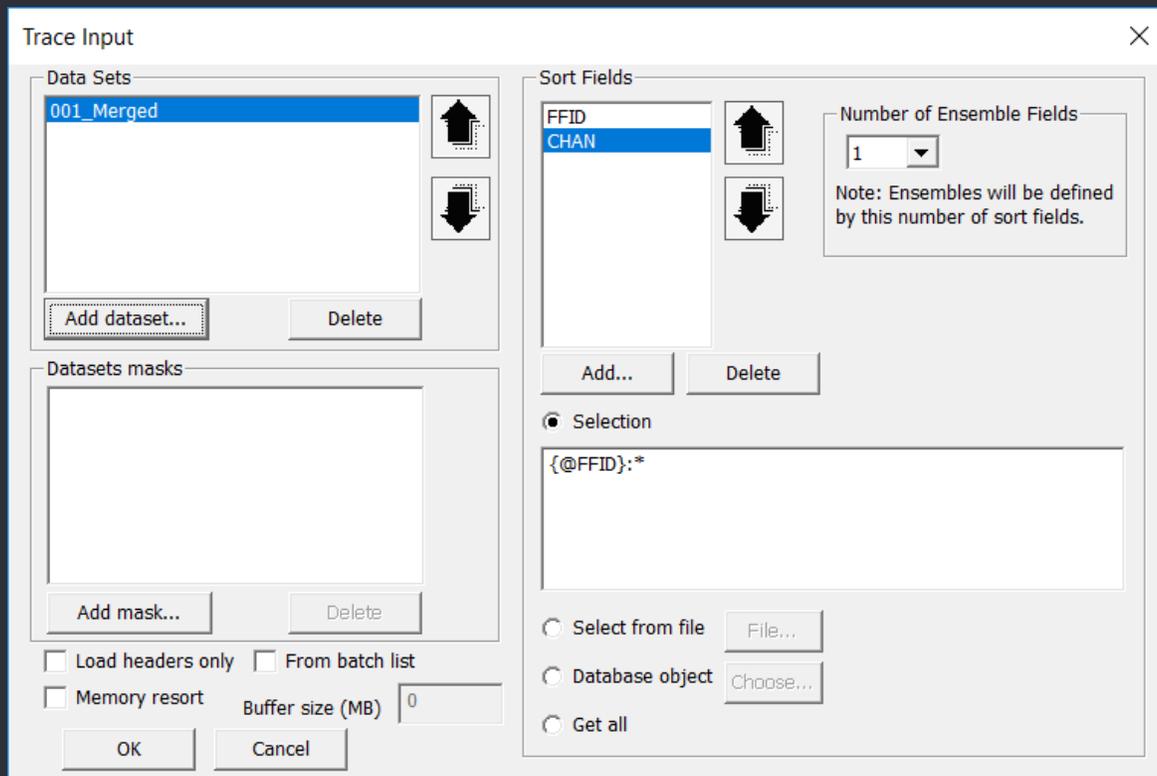
1) Create a replica table. Assumer, our dataset contains of 15 shots with different FFIDs:



6. One more example of replica use

2) Create a flow with Trace Input and Seg-Y Output modules

In the Selection field of the Trace Input module, we will use a variable called FFID that we have defined in the replica table:



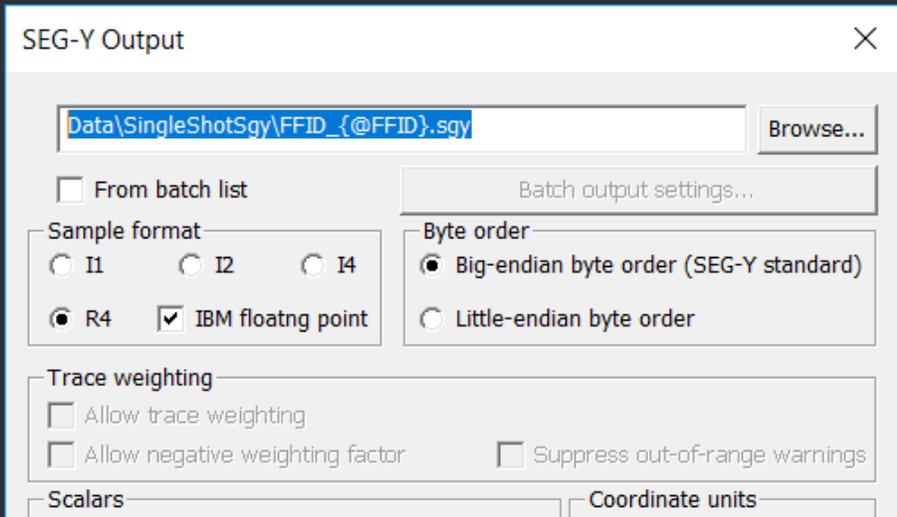
The screenshot shows the 'Trace Input' dialog box with the following configuration:

- Data Sets:** A list containing '001_Merged'.
- Sort Fields:** A list containing 'FFID' and 'CHAN'.
- Number of Ensemble Fields:** A dropdown menu set to '1'. A note below it states: 'Note: Ensembles will be defined by this number of sort fields.'
- Selection:** A radio button is selected, and the text field contains '{@FFID}:*'. Below this are three options: 'Select from file' (with a 'File...' button), 'Database object' (with a 'Choose...' button), and 'Get all'.
- Options:** 'Load headers only' and 'From batch list' are unchecked. 'Memory resort' is unchecked. 'Buffer size (MB)' is set to '0'.
- Buttons:** 'Add dataset...', 'Delete', 'Add mask...', 'Delete', 'Add...', 'Delete', 'OK', and 'Cancel' are visible.

6. One more example of replica use

2) Create a flow with Trace Input and Seg-Y Output modules

In the Seg-Y Output module parameters, we will use the FFID variable in the file name string:

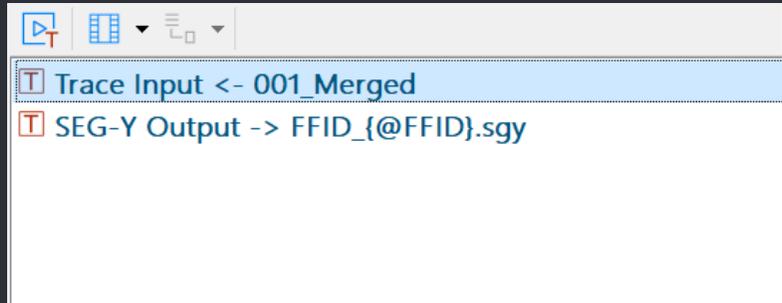


The screenshot shows the 'SEG-Y Output' dialog box with the following settings:

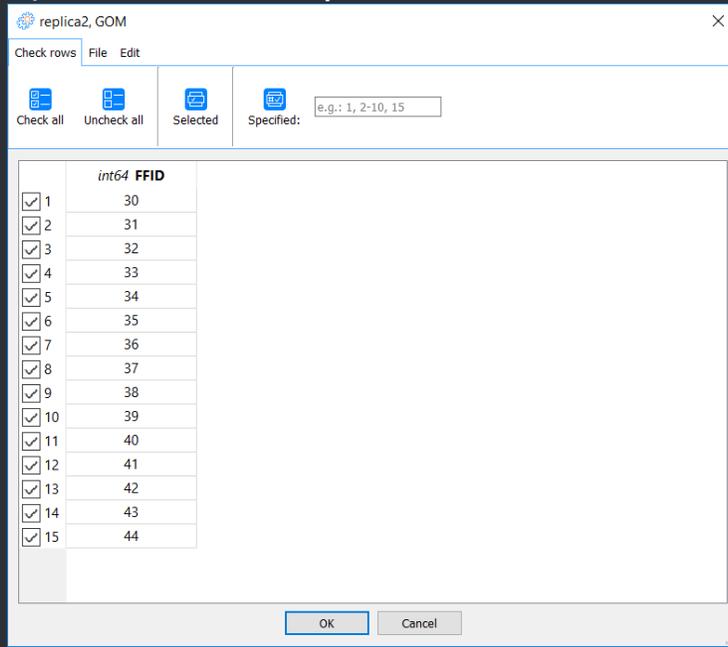
- File name: `Data\SingleShotSgy\FFID_{@FFID}.sgy` (highlighted in blue)
- From batch list:
- Batch output settings...:
- Sample format: I1, I2, I4, R4, IBM floatng point
- Byte order: Big-endian byte order (SEG-Y standard), Little-endian byte order
- Trace weighting: Allow trace weighting, Allow negative weighting factor, Suppress out-of-range warnings
- Scalars:
- Coordinate units:

6. One more example of replica use

3) Run the template flow



4) Select the replica table



5) Here is the result – a set of Seg-Y files

FFID_30.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_31.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_32.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_33.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_34.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_35.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_36.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_37.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_38.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_39.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_40.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_41.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_42.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_43.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ
FFID_44.sgy	07/03/2018 15:29	Файл "SGY"	1,174 КБ

Appendix

Notes on format specifiers for conversion of numbers to strings:

The format specifiers used are a subset of those used in Python standard.

1. Left before a number can only be filled by either zeros or spaces. The default is space, start format specifier with '0' conversion flag to make the numbers zero-padded.
2. d -- string output is signed integer decimal; if a variable converted is real it is rounded to the nearest integer before conversion (0.5 -> 1).
3. f -- string output is a real decimal with a fixed number of decimal; e.g. 6.2f means that the output will always have 2 decimal places.
4. e – string output is real exponential format
5. Format symbol (d, f, e) can be omitted. In this case, the output format will depend on the format number: e.g. 06 would result in zero-padded 6-digit integer and 6.2 in space-padded 6-digit real with 2 decimal places.
6. 7f is equivalent to 7.0f (7-digit number, 0 decimal places), the result will be the same as for 7d.
7. If the length of the output string is not important, use .3f, .3e or just .3. You can also use 0.3f with the same result.