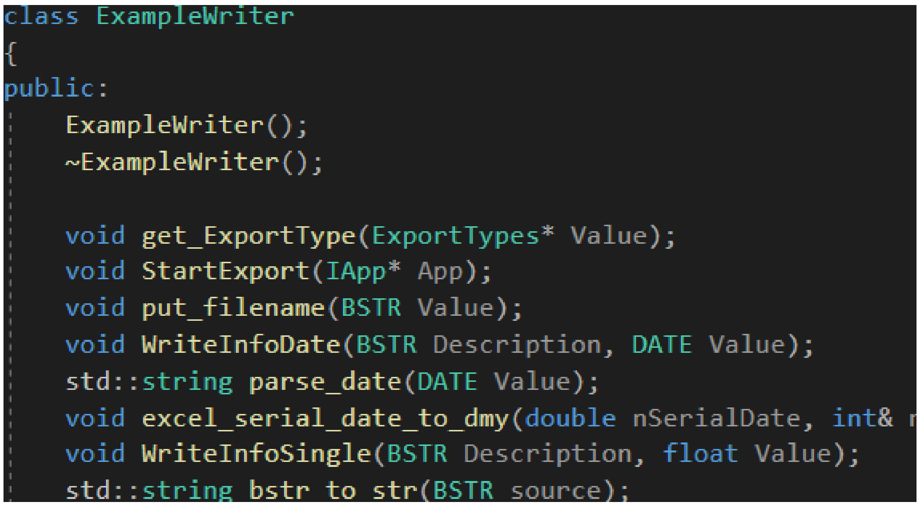
C++ Export Plugin



```
class ExampleWriter
{
public:
    ExampleWriter();
    ~ExampleWriter();

    void get_ExportType(ExportTypes* Value);
    void StartExport(IApp* App);
    void put_filename(BSTR Value);
    void WriteInfoDate(BSTR Description, DATE Value);
    std::string parse_date(DATE Value);
    void excel_serial_date_to_dmy(double nSerialDate, int& r
    void WriteInfoSingle(BSTR Description, float Value);
    std::string bstr_to_str(BSTR source);
```

Introduction to C++ Export Plugin

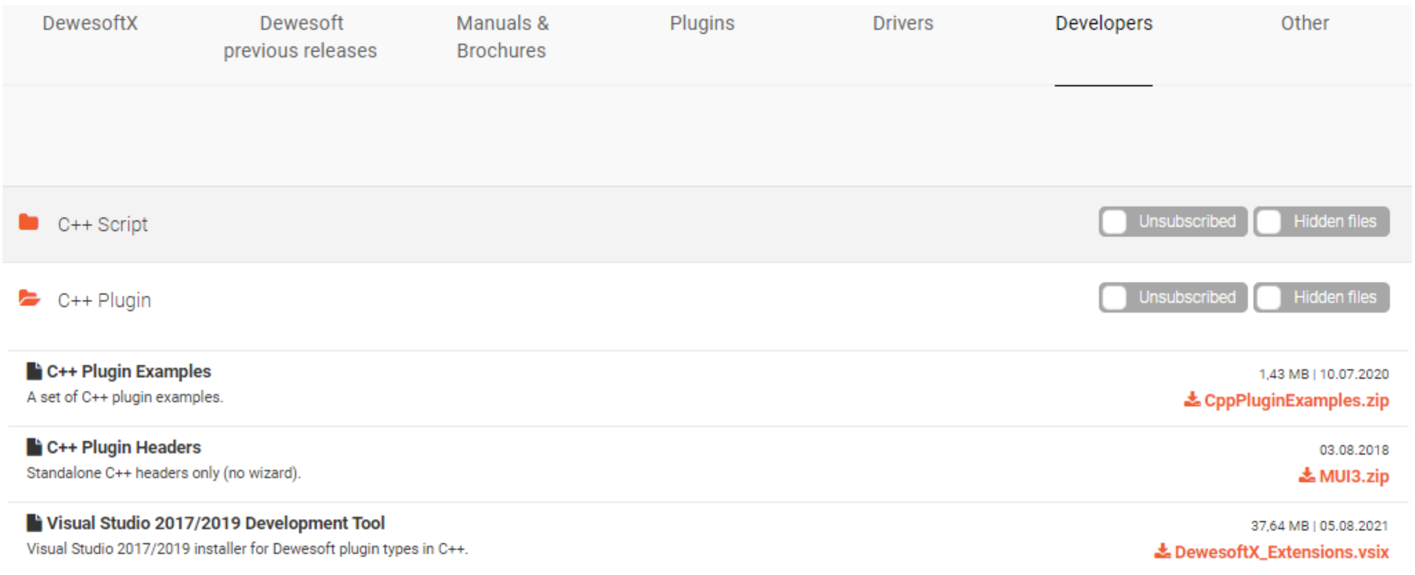
Imagine the scenario where Dewesoft data can be analyzed and post-processed in any software you want. Of course, we already support quite a lot of export formats, but not nearly all of them. With the help of the C++ Export Plugin, you can extend Dewesoft to support whichever format comes to your mind. The output of the C++ Export Plugin is compiled into an external library with .exp extension and will be automatically located and loaded on startup by Dewesoft.

A C++ Export Plugin example that will be shown throughout this tutorial can be obtained using the VSIX installer and will be further explained in the next pages.

How to install the Dewesoft plugin template?

In order to start using C++ Plugin, you must have Visual Studio 2017 or Visual Studio 2019 IDE installed on your system. Some of the reasons we have chosen Visual Studio are its functionalities, powerful developer tooling, like IntelliSense code completion and debugging, fast code editor, easy modification/customization and many more.

Dewesoft plugin for Visual Studio development can be found on the Dewesoft webpage under Visual Studio 2017/2019 Development Tool. Note that you have to be logged in to access the C++ Plugin section. After downloading, just double-click the VSIX file and the installer will guide you through the installation process.



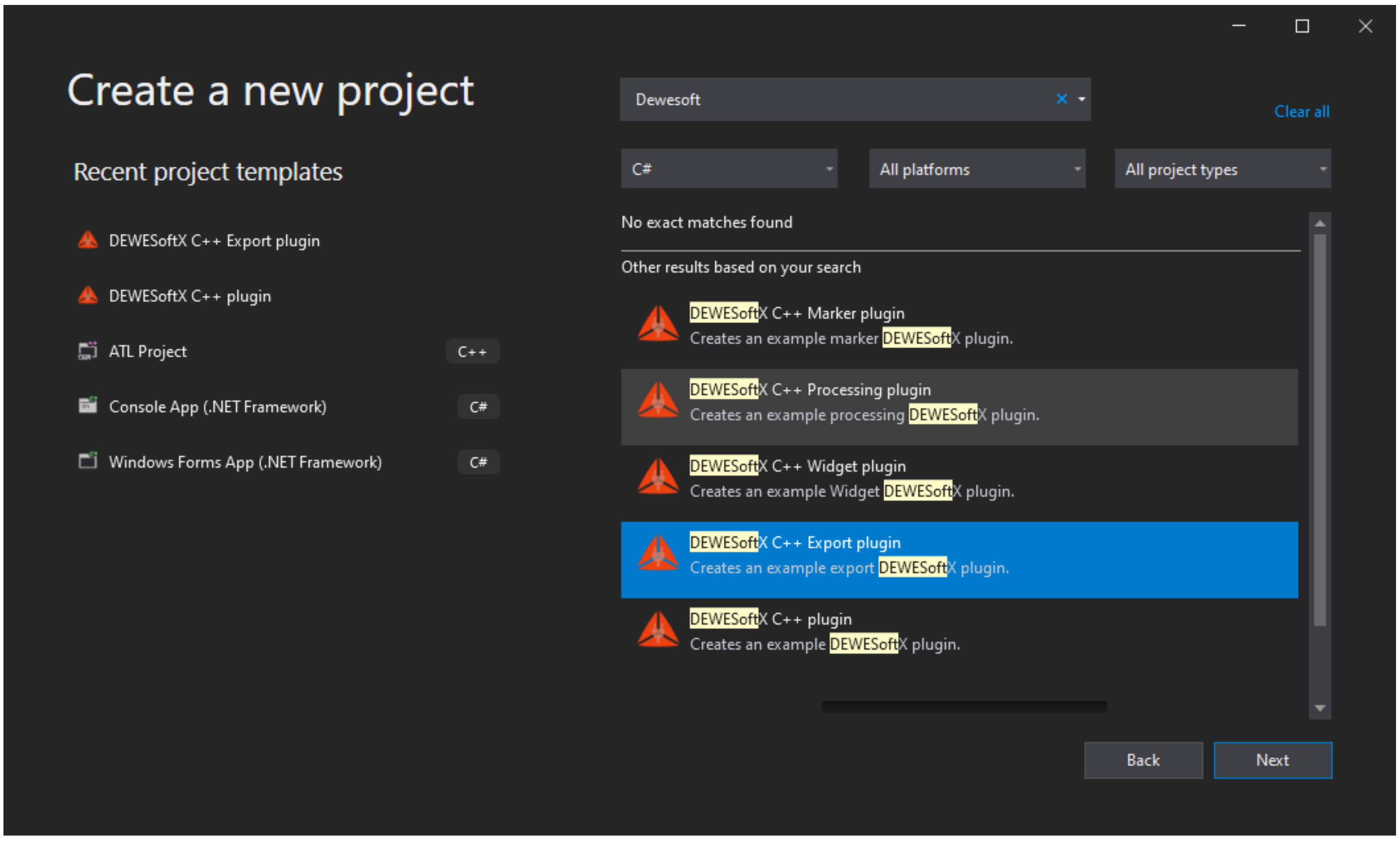
C++ Plugin Examples
A set of C++ plugin examples.
1,43 MB | 10.07.2020
CppPluginExamples.zip

C++ Plugin Headers
Standalone C++ headers only (no wizard).
03.08.2018
MUI3.zip

Visual Studio 2017/2019 Development Tool
Visual Studio 2017/2019 installer for Dewesoft plugin types in C++.
37,64 MB | 05.08.2021
DewesoftX_Extensions.vsix

Once VSIX plugin is downloaded and installed you will be able to create the Dewesoft C++ plugin using the New project window and selecting the DewesoftX C++ Export Plugin project. The template can be found with the help of the Search text box (right top corner) in the online tab.

The new project window is accessed in *File tab* -> *New* -> *Project*.


Create a new project

Recent project templates

DEWESoftX C++ Export plugin

DEWESoftX C++ plugin

ATL Project C++

Console App (.NET Framework) C#

Windows Forms App (.NET Framework) C#

Dewesoft

Clear all

C#

All platforms

All project types

No exact matches found

Other results based on your search

DEWESoftX C++ Marker plugin
Creates an example marker DEWESoftX plugin.

DEWESoftX C++ Processing plugin
Creates an example processing DEWESoftX plugin.

DEWESoftX C++ Widget plugin
Creates an example Widget DEWESoftX plugin.

DEWESoftX C++ Export plugin
Creates an example export DEWESoftX plugin.

DEWESoftX C++ plugin
Creates an example DEWESoftX plugin.

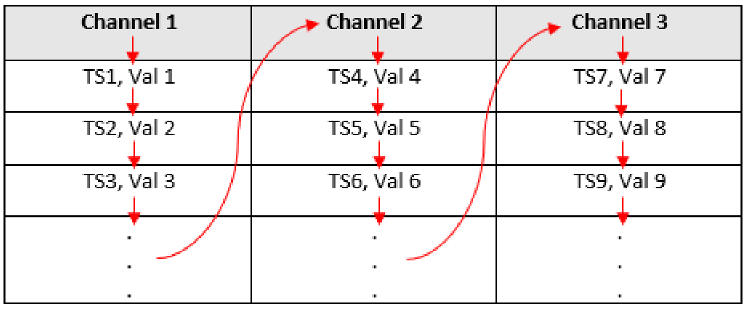
Back

Next

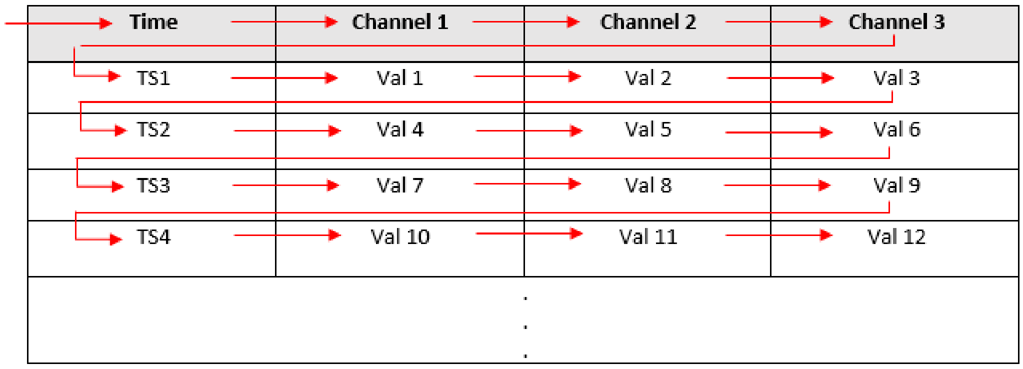
Custom export call diagram

Before we jump to the programming part, we first need to explain how exporting in Dewesoft works. There are two types of exports, channel-based, and value-based.

- Channel-based export - writes all data (and times) from one channel. Once all the samples are written, we start writing another channel.



- Value-based export - writes the N-th sample for each channel where N represents the time from start to finish. When all N-th samples are written, N is increased by the time step of the channel with the highest synchronous rate.



Export type can be defined inside *get_ExportType()* function.

The main difference between these two types of export is the sequence of functions that are called from start to finish of the export. The sequence of function calls and their brief explanation can be found in the tables below.

Function	**Comment**
StartExport	Beginning of export, can be used to create file to be exported to
StartInfo WriteInfoString WriteInfoDate WriteInfoInteger WriteInfoSingle EndInfo	Used to obtain basic information about exported settings like start time, number of exported channels, storing type, datafile name, datafile storing time…
StartEvents WriteEvent (called N times) EndEvents	Used to obtain event information about the exported datafile like start and stop storing, text events…
StartDataFolder	Called on each separate start of storing
StartChannelEx WriteValue	Write single values. Called multiple times for each stored single value.
StartTimeField StartDataField	StartTimeField – used to obtain time unit StartDataField – used to obtain channels unit and sample rate These two functions are called for each channel
EndHeader	Starting point when Dewesoft starts returning sample values
StartValue WriteValue EndValue	StartValue – returns relative time StartAbsValue – return absolute time Dewesoft returning sample values
EndDataFolder	Called when whole section from start to stop is exported
EndExport	Called at the end of export

Value-based sequence of calls

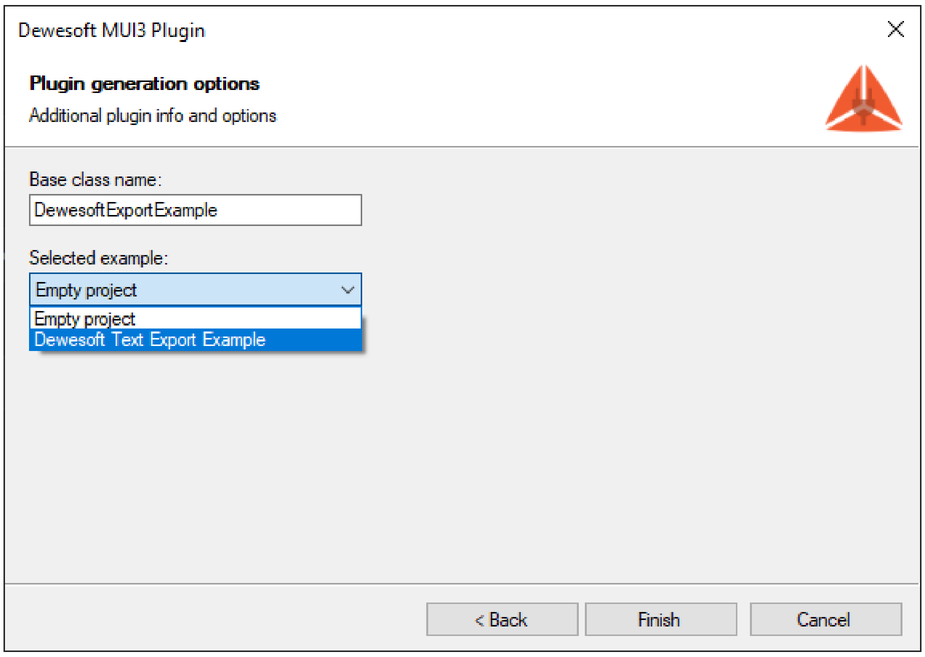
Function		Comment
StartExport		Beginning of export, can be used to create file to be exported to
StartInfo		Used to obtain basic information about exported settings like start time, number of exported channels, storing type, datafile name, datafile storing time…
WriteInfoString		
WriteInfoDate		
WriteInfoInteger		
WriteInfoSingle		
EndInfo		
StartEvents		Used to obtain event information about the exported datafile like start and stop storing, text events…
WriteEvent		
StopEvents		
StartDataFolder		Called at the beginning of each new data folder.
Set_DataCount		SetChannel – used to obtain IChannel interface Set_DataCount – used to obtain number of samples per channel StartDataField – used to obtain channels unit and sample rate
StartTimeField		
SetChannel	Called for each exported channel	
Set_DataCount		
StartDataField		
EndHeader		Starting point where Dewesoft starts returning sample values
StartValue		First timestamp of sync channels
SetChannel		Dewesoft returning sample values
StartChannel		
WriteVal		
EndChannel		
EndDataFolder		Called when whole section from start to stop is exported
EndExport		Called at the end of export

Channel-based sequence of calls

Example - exporting data as text to be imported into Dewesoft

As already mentioned, this example can be obtained using the *MUI Plugin Wizard*. If you are not familiar with the basic settings MUI Plugin Wizard provides, you can read more about them in Basic custom plugin development in C++ under Example: New C++ Plugin.

To create the same example as we are using in this tutorial, you have to select "Dewesoft Test Export Example" in the combo box, as seen in the picture.


Dewesoft MUI3 Plugin

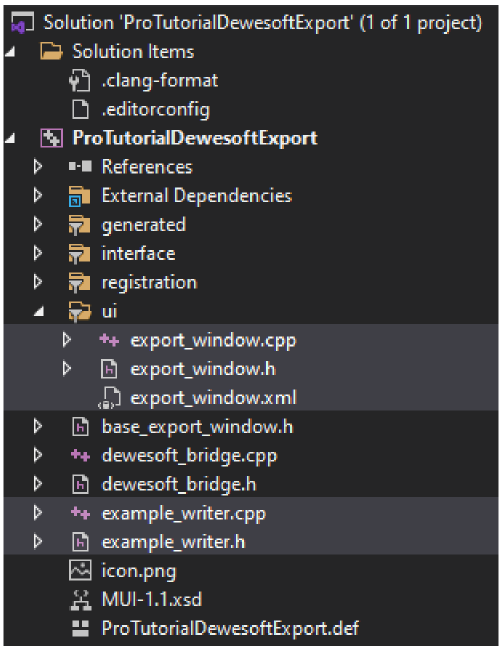
Plugin generation options

Additional plugin info and options

Base class name: DewesoftExportExample

Selected example: Empty project

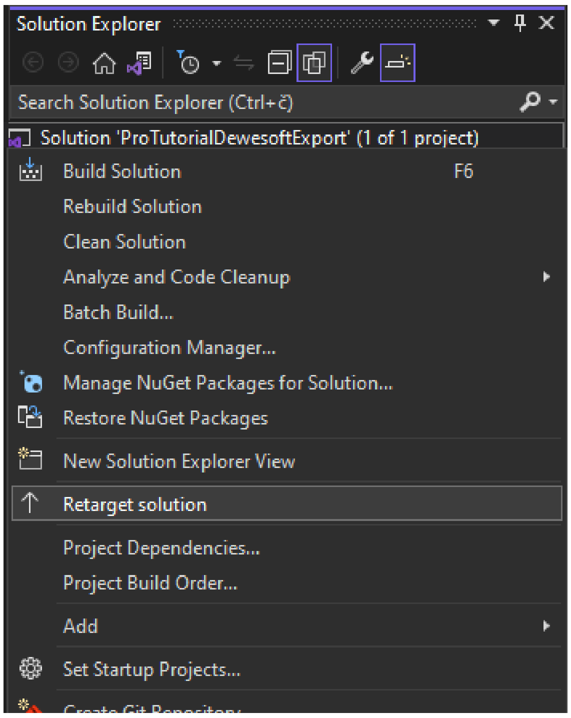
This example shows a complete example of how to create a text file, which can later be imported into the Dewesoft. It will also serve as a solid foundation for your very own custom export. Once the project is created, you should be able to see the following project structure.


```
Solution 'ProTutorialDewesoftExport' (1 of 1 project)
  Solution Items
    .clang-format
    .editorconfig
  ProTutorialDewesoftExport
    References
    External Dependencies
    generated
    interface
    registration
    ui
      export_window.cpp
      export_window.h
      export_window.xml
    base_export_window.h
    dewesoft_bridge.cpp
    dewesoft_bridge.h
    example_writer.cpp
    example_writer.h
    icon.png
    MUI-1.1.xsd
    ProTutorialDewesoftExport.def
```

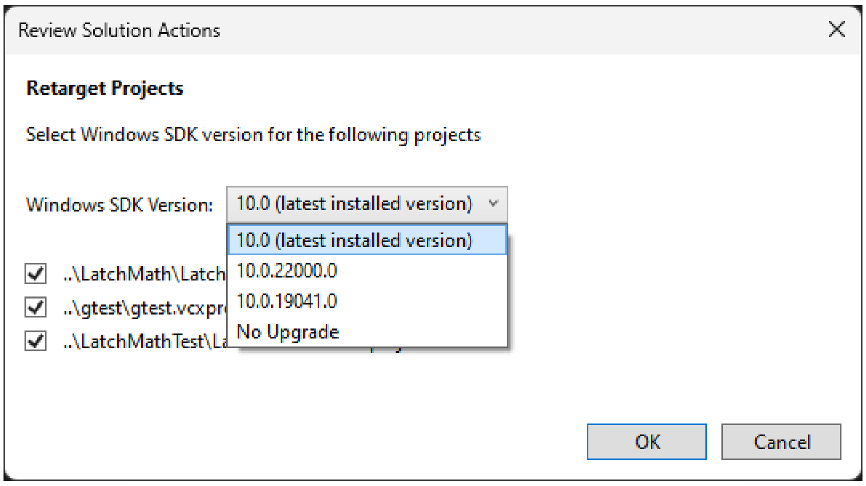
Most of the work is done in the selected files:

- *ui folder* - user interface and its actions are defined there
- *example_writer files* - endpoint where all export functionality should be added

Once the plugin is created and you stumble upon Error MSB8036: The Windows SDK version sdk_version was not found. You should retarget your solution to use the one you have installed. You can do this by right clicking on your solution node in Solution explorer and selecting Retarget solution.



Once the Retarget solution window is opened, you can select "Latest installed version" item.



User interface and settings storing

To simplify export as much as possible, the user can only change one export setting - delimiter character. For that reason, the MUI library has been added to the custom exports. In general, this library enables you to create a custom user interface for interacting with users. If you are not yet familiar with the workflow of the MUI library, you can read about it in Basic custom plugin development in C++. In our case, the user can choose between three characters (tab, comma, and semicolon).

Every time Dewesoft is closed, the settings of the export plugin are reset. To avoid this we can store these settings and reload them next time when Dewesoft opens. Functions for reading and writing settings are already prepared in the example.

```
void ExampleWriter::write_ini_file(
{
    tortellini::ini ini
    ini["UI"]["DELIMITER"] = data_delimiter; // here we store the delimiter setting

    update_location_of_dll();
    std::ofstream out((location_of_dll_folder + "example_export.ini").c_str());
    out << ini;
}
```

In our case, we only have one setting to save. To save (write) the setting, the following line of code is used where the first bracket (eg. UI) represents the group and the second bracket (eg. Delimiter) represents id for setting inside the group. User can change the *data_delimiter* value by changing the selected item in ComboBox.

To read the setting value, we use the following function, where we need the same group and unique id for each value. If the value is not stored at the moment of reading (eg. first-time running of application), we specify the default value for the variable. In our case, that will be the comma (",") character.

```
void ExampleWriter::read_ini_file()
{
    tortellini::ini ini;
    update_location_of_dll();
    std::ifstream in((location_of_dll_folder + "example_export.ini").c_str());

    in >> ini;
    data_delimiter = ini["UI"]["DELIMITER"] | ","; // here we are reading the stored delimiter setting
}
```

Export output

Before we continue, we need to understand the format of the exported target format. In our simple case, our format requires value-based data as well as single-value channels. Therefore, we will set the export type to etValueBased inside get_ExportType function.

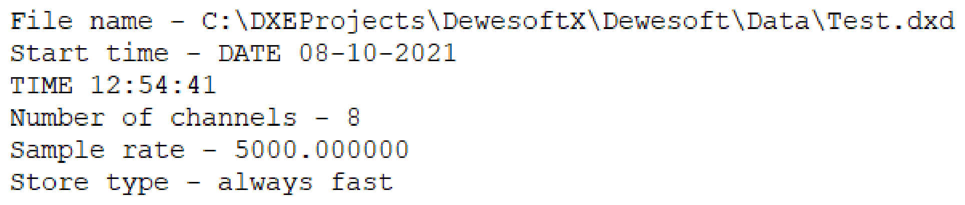
```
void ExampleWriter::get_ExportType(ExportTypes* Value)
{
    *Value = etValueBased;
}
```

Custom export in this example generates only one file, consisting of the header and the data. File name is obtained in put_filename function, the file stream is created in the StartExport function. Data will be written later. To generate the header part of the file we simply output all the available metadata about the datafile, separated with the new line.

```
void ExampleWriter::WriteInfoString(BSTR Description, BSTR Value)
{
    write_to_output_file(bstr_to_str(Description) + " - " + bstr_to_str(Value), true);
}
```

This is done in all functions that provide export metadata.

The header content can be seen in the picture below.



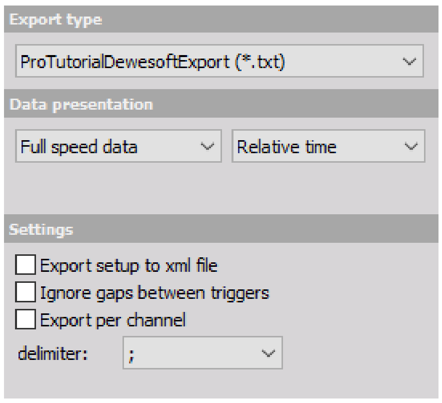
```
File name - C:\DXEProjects\DewesoftX\Dewesoft\Data\Test.dxd
Start time - DATE 08-10-2021
TIME 12:54:41
Number of channels - 8
Sample rate - 5000.000000
Store type - always fast
```

As per the data part of the file, the format requires the first row to contain channel names, while all other rows contain values (each row contains N values, where N is the number of channels). To output the channel names, we use SetChannel function, where we also check for export sample rate to be the same for all channels (which is also a requirement). If the sample rates do not match, the export will exit and nothing will be exported.

To export the data, we use *WriteValue* function, its body can be seen in the code below. This function is called for each sample, which is written to the file and is later followed with the delimiter or new line (if we just wrote to the last channel). *write_to_output_file* was implemented in the example and will add string content to the output file.

```
void ExampleWriter::WriteValue(float Value)
{
   write_to_output_file(std::to_string(Value));
   sample_counter++;
   if (sample_counter % number_of_channels == 0)
   {
      write_to_output_file("", true);
      sample_counter = 0;
   }
   else
      write_to_output_file(get_data_delimiter_char());
}
```

If we now take a look at the data part of the output file, we can see that the data is organized and supports the Dewesoft format. As you can see, the delimiter is semicolon (";") character because it was selected in the combo box as a default delimiter.



Export type

ProTutorialDewesoftExport (*.txt)

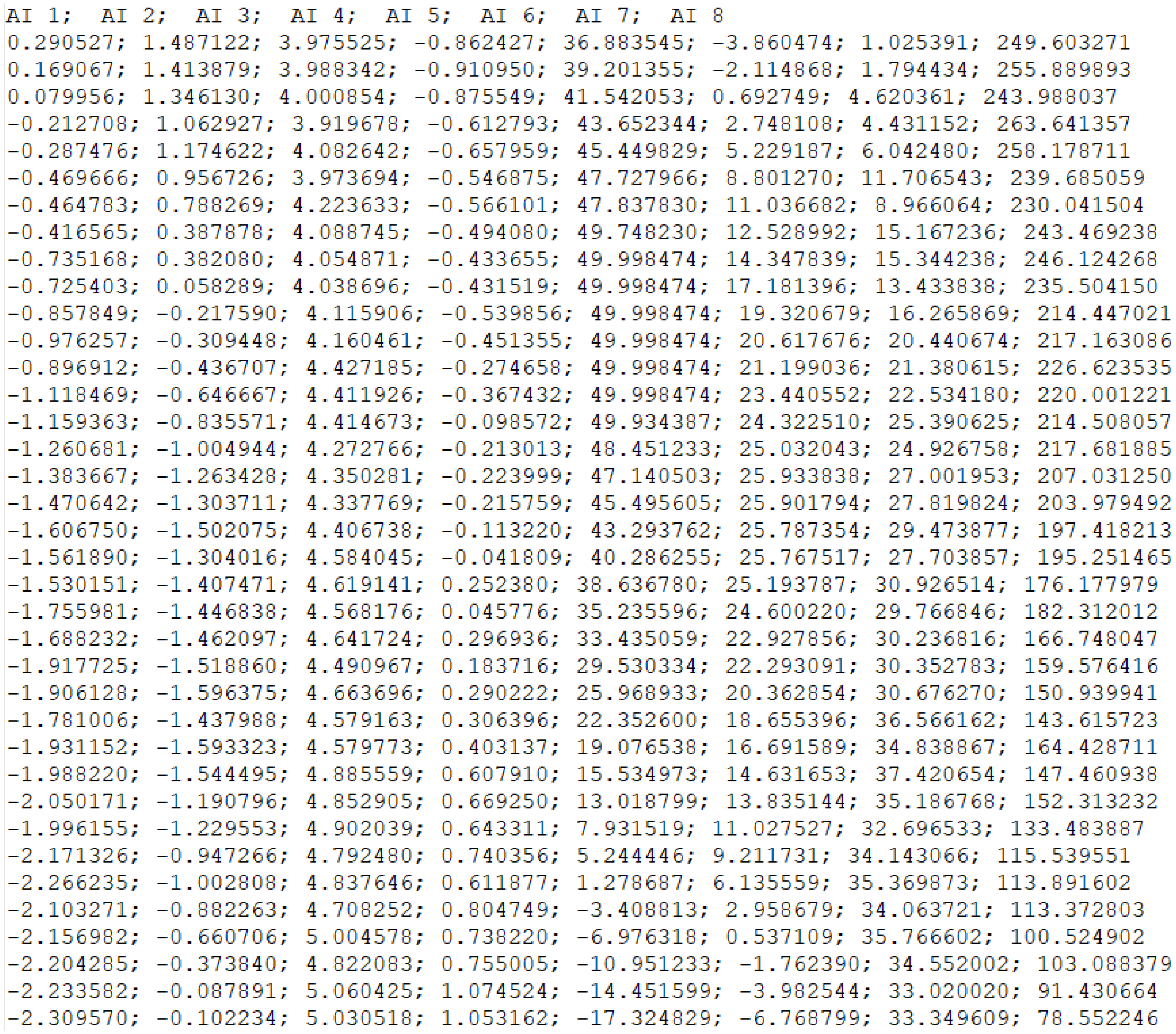
Data presentation

Full speed data | Relative time

Settings

- [ ] Export setup to xml file
- [ ] Ignore gaps between triggers
- [ ] Export per channel

delimiter: ;



```
AI 1;  AI 2;  AI 3;  AI 4;  AI 5;  AI 6;  AI 7;  AI 8
0.290527; 1.487122; 3.975525; -0.862427; 36.883545; -3.860474; 1.025391; 249.603271
0.169067; 1.413879; 3.988342; -0.910950; 39.201355; -2.114868; 1.794434; 255.889893
0.079956; 1.346130; 4.000854; -0.875549; 41.542053; 0.692749; 4.620361; 243.988037
-0.212708; 1.062927; 3.919678; -0.612793; 43.652344; 2.748108; 4.431152; 263.641357
-0.287476; 1.174622; 4.082642; -0.657959; 45.449829; 5.229187; 6.042480; 258.178711
-0.469666; 0.956726; 3.973694; -0.546875; 47.727966; 8.801270; 11.706543; 239.685059
-0.464783; 0.788269; 4.223633; -0.566101; 47.837830; 11.036682; 8.966064; 230.041504
-0.416565; 0.387878; 4.088745; -0.494080; 49.748230; 12.528992; 15.167236; 243.469238
-0.735168; 0.382080; 4.054871; -0.433655; 49.998474; 14.347839; 15.344238; 246.124268
-0.725403; 0.058289; 4.038696; -0.431519; 49.998474; 17.181396; 13.433838; 235.504150
-0.857849; -0.217590; 4.115906; -0.539856; 49.998474; 19.320679; 16.265869; 214.447021
-0.976257; -0.309448; 4.160461; -0.451355; 49.998474; 20.617676; 20.440674; 217.163086
-0.896912; -0.436707; 4.427185; -0.274658; 49.998474; 21.199036; 21.380615; 226.623535
-1.118469; -0.646667; 4.411926; -0.367432; 49.998474; 23.440552; 22.534180; 220.001221
-1.159363; -0.835571; 4.414673; -0.098572; 49.934387; 24.322510; 25.390625; 214.508057
-1.260681; -1.004944; 4.272766; -0.213013; 48.451233; 25.032043; 24.926758; 217.681885
-1.383667; -1.263428; 4.350281; -0.223999; 47.140503; 25.933838; 27.001953; 207.031250
-1.470642; -1.303711; 4.337769; -0.215759; 45.495605; 25.901794; 27.819824; 203.979492
-1.606750; -1.502075; 4.406738; -0.113220; 43.293762; 25.787354; 29.473877; 197.418213
-1.561890; -1.304016; 4.584045; -0.041809; 40.286255; 25.767517; 27.703857; 195.251465
-1.530151; -1.407471; 4.619141; 0.252380; 38.636780; 25.193787; 30.926514; 176.177979
-1.755981; -1.446838; 4.568176; 0.045776; 35.235596; 24.600220; 29.766846; 182.312012
-1.688232; -1.462097; 4.641724; 0.296936; 33.435059; 22.927856; 30.236816; 166.748047
-1.917725; -1.518860; 4.490967; 0.183716; 29.530334; 22.293091; 30.352783; 159.576416
-1.906128; -1.596375; 4.663696; 0.290222; 25.968933; 20.362854; 30.676270; 150.939941
-1.781006; -1.437988; 4.579163; 0.306396; 22.352600; 18.655396; 36.566162; 143.615723
-1.931152; -1.593323; 4.579773; 0.403137; 19.076538; 16.691589; 34.838867; 164.428711
-1.988220; -1.544495; 4.885559; 0.607910; 15.534973; 14.631653; 37.420654; 147.460938
-2.050171; -1.190796; 4.852905; 0.669250; 13.018799; 13.835144; 35.186768; 152.313232
-1.996155; -1.229553; 4.902039; 0.643311; 7.931519; 11.027527; 32.696533; 133.483887
-2.171326; -0.947266; 4.792480; 0.740356; 5.244446; 9.211731; 34.143066; 115.539551
-2.266235; -1.002808; 4.837646; 0.611877; 1.278687; 6.135559; 35.369873; 113.891602
-2.103271; -0.882263; 4.708252; 0.804749; -3.408813; 2.958679; 34.063721; 113.372803
-2.156982; -0.660706; 5.004578; 0.738220; -6.976318; 0.537109; 35.766602; 100.524902
-2.204285; -0.373840; 4.822083; 0.755005; -10.951233; -1.762390; 34.552002; 103.088379
-2.233582; -0.087891; 5.060425; 1.074524; -14.451599; -3.982544; 33.020020; 91.430664
-2.309570; -0.102234; 5.030518; 1.053162; -17.324829; -6.768799; 33.349609; 78.552246
```

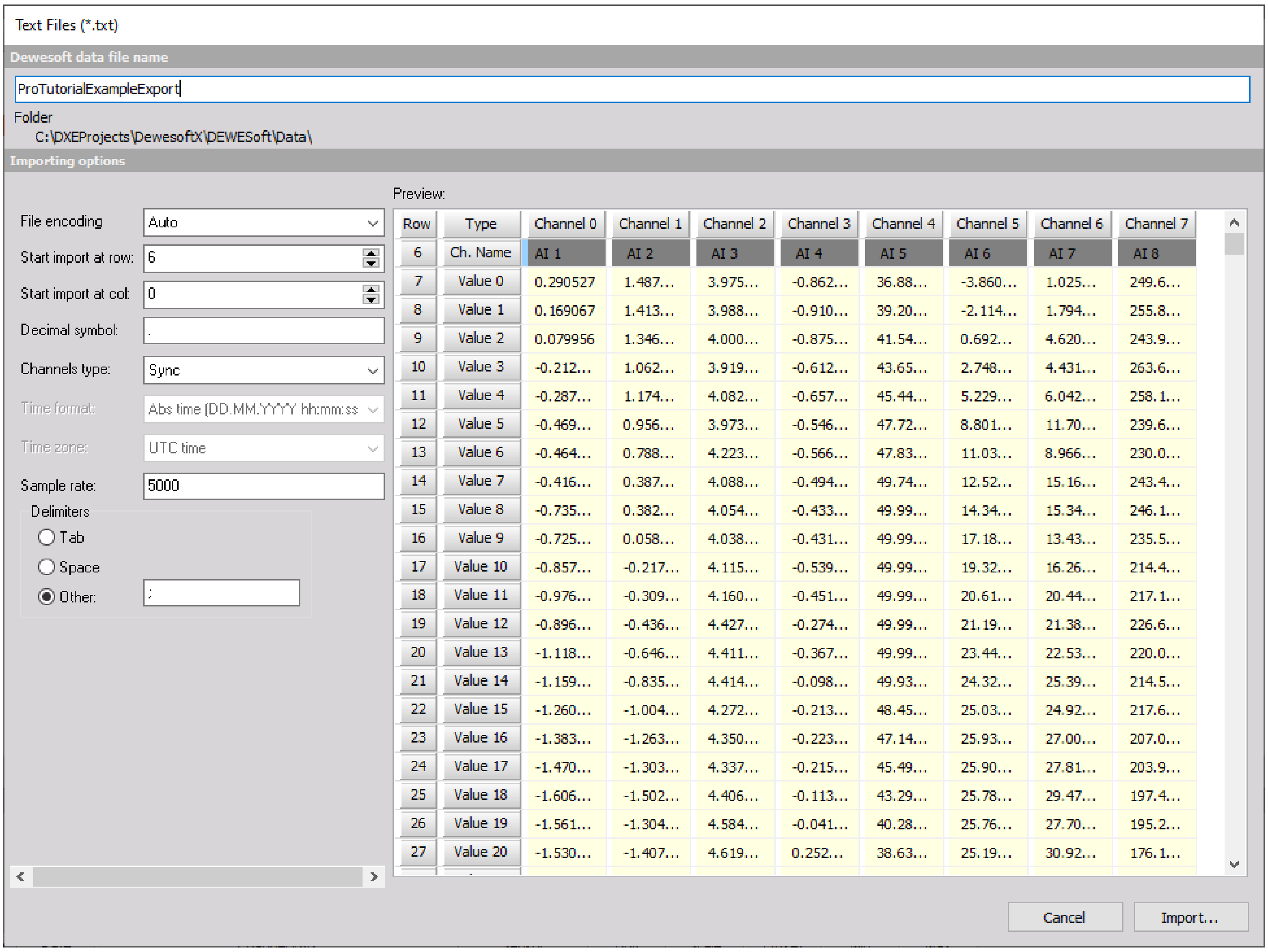
For the purpose of this Pro tutorial, we did not have to modify any Dewesoft internals. But if we wanted to, for example, export any other channel type or stop the export we would need to send a notification to the Dewesoft. For example, if we wanted to export array channels, we notify Dewesoft by returning *True* when calling the evSupportsArray function. This needs to be done inside *DewesoftBridge::OnEvent* function. There are many other events that can be used for notifying or sending info to the Dewesoft but they will not be covered in this tutorial.

Testing the output

Of course, the whole point of exporting the data is to be able to get the data in another form (to be able to open it in another software), but we will just open the exported data file again in Dewesoft. Exporting from Dewesoft and importing into Dewesoft does not make much sense, but for the purpose of this pro tutorial, this should wrap the whole thing into a complete example.

In order to test this, you need the TxtImport plugin. Once downloaded and added to the addons folder, you should be able to see it under Analyse and Import.

After selecting newly exported file, you should set correct settings (panel on the left). **The delimiter should be the same as the one you exported with!** In our case, that was semicolon (";").



Text Files (*.txt)

Dewesoft data file name

ProTutorialExampleExport

Folder
C:\DXEProjects\DewesoftX\DEWESoft\Data\

Importing options

File encoding	Auto
Start import at row:	6
Start import at col:	0
Decimal symbol:	.
Channels type:	Sync
Time format:	Abs time (DD.MM.YYYY hh:mm:ss
Time zone:	UTC time
Sample rate:	5000

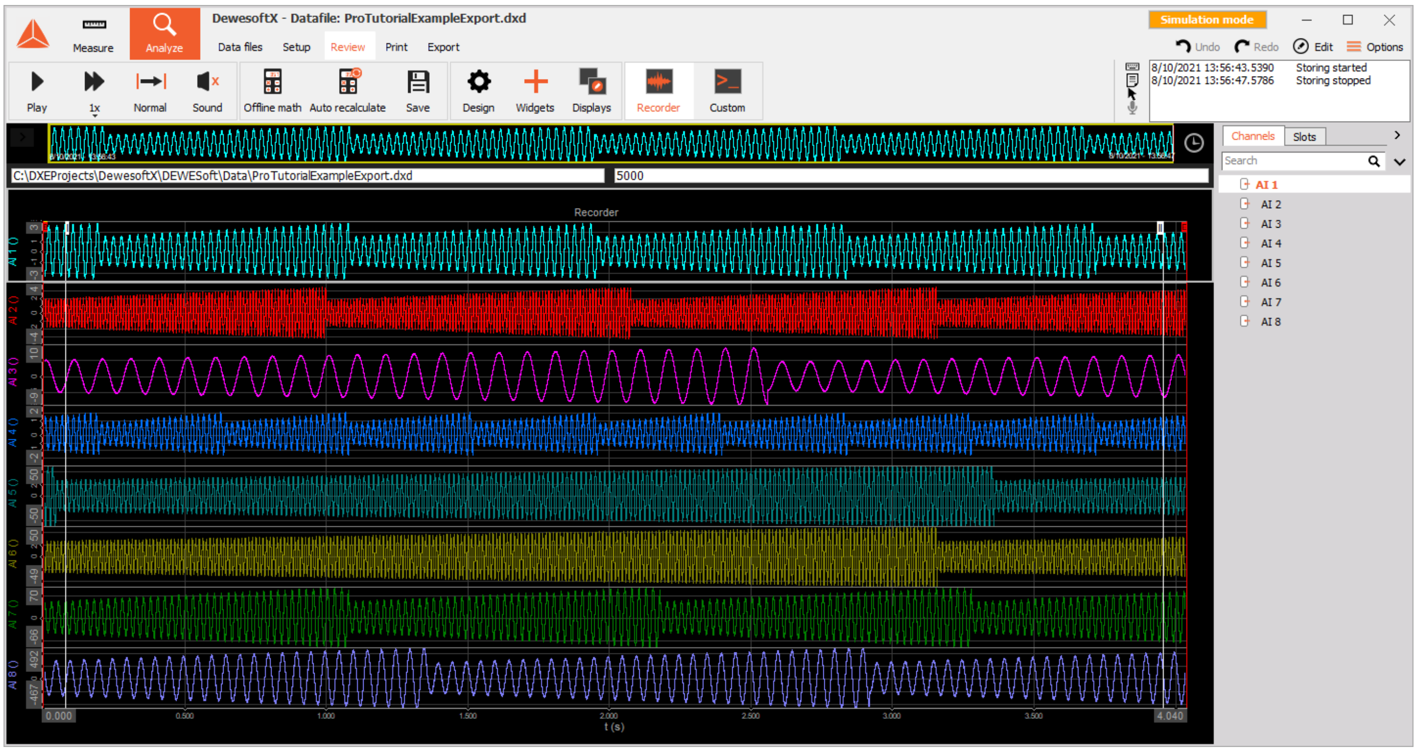
Delimiters
Tab
Space
[✓ Other:] ;

Preview:

Row	Type	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
6	Ch. Name	AI 1	AI 2	AI 3	AI 4	AI 5	AI 6	AI 7	AI 8
7	Value 0	0.290527	1.487...	3.975...	-0.862...	36.88...	-3.860...	1.025...	249.6...
8	Value 1	0.169067	1.413...	3.988...	-0.910...	39.20...	-2.114...	1.794...	255.8...
9	Value 2	0.079956	1.346...	4.000...	-0.875...	41.54...	0.692...	4.620...	243.9...
10	Value 3	-0.212...	1.062...	3.919...	-0.612...	43.65...	2.748...	4.431...	263.6...
11	Value 4	-0.287...	1.174...	4.082...	-0.657...	45.44...	5.229...	6.042...	258.1...
12	Value 5	-0.469...	0.956...	3.973...	-0.546...	47.72...	8.801...	11.70...	239.6...
13	Value 6	-0.464...	0.788...	4.223...	-0.566...	47.83...	11.03...	8.966...	230.0...
14	Value 7	-0.416...	0.387...	4.088...	-0.494...	49.74...	12.52...	15.16...	243.4...
15	Value 8	-0.735...	0.382...	4.054...	-0.433...	49.99...	14.34...	15.34...	246.1...
16	Value 9	-0.725...	0.058...	4.038...	-0.431...	49.99...	17.18...	13.43...	235.5...
17	Value 10	-0.857...	-0.217...	4.115...	-0.539...	49.99...	19.32...	16.26...	214.4...
18	Value 11	-0.976...	-0.309...	4.160...	-0.451...	49.99...	20.61...	20.44...	217.1...
19	Value 12	-0.896...	-0.436...	4.427...	-0.274...	49.99...	21.19...	21.38...	226.6...
20	Value 13	-1.118...	-0.646...	4.411...	-0.367...	49.99...	23.44...	22.53...	220.0...
21	Value 14	-1.159...	-0.835...	4.414...	-0.098...	49.93...	24.32...	25.39...	214.5...
22	Value 15	-1.260...	-1.004...	4.272...	-0.213...	48.45...	25.03...	24.92...	217.6...
23	Value 16	-1.383...	-1.263...	4.350...	-0.223...	47.14...	25.93...	27.00...	207.0...
24	Value 17	-1.470...	-1.303...	4.337...	-0.215...	45.49...	25.90...	27.81...	203.9...
25	Value 18	-1.606...	-1.502...	4.406...	-0.113...	43.29...	25.78...	29.47...	197.4...
26	Value 19	-1.561...	-1.304...	4.584...	-0.041...	40.28...	25.76...	27.70...	195.2...
27	Value 20	-1.530...	-1.407...	4.619...	0.252...	38.63...	25.19...	30.92...	176.1...

Cancel
Import...

After clicking "Import" button, the final result can be seen inside Dewesoft.



```
8/10/2021 13:56:43.5390    Storing started
8/10/2021 13:56:47.5786    Storing stopped
```

C:\DXEProjects\DewesoftX\DEWESoft\Data\ProTutorialExampleExport.dxd

5000