NSort for SSIS

A High Performance Sort Component for SQL Server Integration Services

Version 1.1.18 – March 20, 2011

Ordinal Technology Corp <support@ordinal.com>



Introduction

The NSort transformation for Microsoft SQL Server 2005 and 2008 Integration Services, TxNsort, can be used alongside standard components during the design and execution of an SSIS project. Compared to the standard SSIS sort component, the NSort transformation:

- 1. Is not limited by the amount of available main memory during execution time. NSort uses temporary files to handle arbitrarily large datasets.
- 2. Uses less CPU and is faster.
- 3. Can accept multiple input streams.
- 4. Can partition its output data into multiple output streams which are emitted in parallel, facilitating parallel downstream processing.

Support

Support can be obtained for NSort for SSIS by emailing <u>support@ordinal.com</u>, or by phone +1-925-253-9204 during normal business hours: M-F 9AM – 5PM Pacific Time.

Installation

An NSort for SSIS zip file along with a trial NSort license key can be obtained from http://www.ordinal.com/try.cgi

After completing the online form, you will be able to download the NSort for SSIS 2005 or 2008 zip file (x86 or x64 version) and will be emailed a trial (temporary) license key. The NSort zip file contains the following files:





To install NSort for SSIS, double-click on the setup icon. You will be asked to select the installation folder:

🔀 Nsort for SSIS		
Select Installation Folde	۶r	
The installer will install Nsort for SSIS to t	he following folder.	
To install in this folder, click "Next". To in	nstall to a different folder, enter it be	elow or click "Browse".
<u>F</u> older:		
C:\Program Files\Nsort\		Browse
		Disk Cost
Install Nsort for SSIS for yourself, or for	r anyone who uses this computer:	
• Everyone		
⊂ Just <u>m</u> e		
	Cancel < <u>B</u> ack	<u>N</u> ext >

You should note the installation folder, as you need to know it to install your temporary license key. Once the NSort for SSIS installation is complete, you can install the trial license key that has been emailed to you by copying it to the *license.txt* file in the NSort installation folder. You can use the *notepad* application to do this:



If your license key is not up-to-date or is missing, you can design SSIS projects with NSort. However the NSort component will fail during project execution and the following error message will appear in the SSIS output:

Error: 2006-01-25 10:26:04.37 Code: 0xC020CC0A Source: Data Flow Task Nsort Description: Nsort library reports nsort_define() error: (<current program>) NO_LICENSE: A -license="<license string>" statement is needed to run nsort New license info for yourservername: nsort-ssis 3.3.11 x86 00C09F25C1A8 1 2

You can usually get another temporary license key at <u>http://www.ordinal.com/temporary.cgi</u> by copying the last line of the error message (the one starting with "New license info...") and pasting it into the form on the website. You will be emailed a new temporary license key that you will need to copy into the *license.txt* file.

Getting the NSort in the Toolbox

Once NSort for SSIS is installed, you should configure your Business Intelligence Development Environment so that NSort appears in the Toolbox alongside the standard Data Flow Transformations:



To have the NSort component appear in the toolbox with the standard data flow transformations, select *Choose Toolbox Items*... from the *Tools* menu, select the *SSIS Data Flow Items* tab, find the NSort component and make sure it is checked, then click on the *OK* button.

Choose Toolbox Items		? 🛛
.NET Framework	Components	COM Components
Maintenance Tasks	SSIS Data Flow Items	SSIS Control Flow Items
Name	Path	Type Name
Import Column	C:\Program Files\Microsoft SOL Serv	TxEileInserter, Inserter, 1
	C:\Program Files\Microsoft SOL Serv	DTSTransform.Lookup. 1
Merge	C: Program Files Microsoft SOL Serv	DTSTransform.Merge. 1
Merge Join	C:\Program Files\Microsoft SOL Serv	DTSTransform.MergeJoin.1
Multicast	C: Program Files Microsoft SOL Serv	DTSTransform.Multicast.1
Nsort	C:\Program Files\Wsort\TxNsort.dll	DTSTransform.Nsort.1
OLE DB Command	C:\Program Files\Microsoft SQL Serv	DTSTransform.OLEDBCo
OLE DB Destination	C: Program Files Microsoft SQL Serv	DTSAdapter.OLEDBDesti
OLE DB Source	C:\Program Files\Microsoft SQL Serv	DTSAdapter.OLEDBSourc
Partition Processing	C:\Program Files\Microsoft Visual Stu	MSMDPP.PXPipelineProce
Percentage Sampling	C:\Program Files\Microsoft SQL Serv	DTSTransform.PctSamplin 🗸
Neort	· · · ·	
The second secon		
A Language: English	(United States)	
version: 1.1.1.	1	
L		
		X Cancel <u>R</u> eset

The NSort component should now appear in the toolbox with the Data Flow Transformations, albeit at the end of that group. Its place in the toolbox can be modified by dragging it to the desired (alphabetical) location

Using the NSort Transformation

To use the NSort transformation, drag the NSort icon from the toolbox onto the Data Flow design surface. Once the upstream components are properly configured, their output can be connected to the NSort component:



With the NSort component's input connected, you can double-click it to open its transformation editor:

💥 Nsort Transform	ation Editor				×
General Outputs					
Input Column	Data Type	Usage	Sort Order	Comparison Flags	Output Alias
Column 0	DT_STR	Pass-Thru 💌			Column 0
Column 1	DT_STR	Pass-Thru 💌			Column 1
•					F
Temp File Director					
remptie Directory	r	,			
Memory Megabyte	s 0 📑			Maximum Thread	🗄 🛛 🛛
🔲 Eliminate rows	with duplicate k	ey values			
				ок	Cancel
					//

Input Column Usage

Each NSort input column is displayed in the editor. Input columns can be used in 4 different ways:

- *Pass-Thru* (default) the column is passed through the NSort transformation without being used as a sort key.
- *Key Ascending* the column is used as a sort key in ascending (lowest to highest) order.
- *Key Descending* the column is used as a sort key in descending (highest to lowest) order.
- *Delete* the column is not used in the sort, and does not appear in the output.

The input column's usage can be specified in its Usage box:

ieneral Outputs	1				
Input Column	Data Type	Usage	Sort Order	Comparison Flags	Output Alias
Column O	DT_STR	Key Ascending 💌			Column 0
Column 1	DT_STR	Pass-Thru Kay Assending			Column 1
		Key Descending			
		Delete			
1					
I	011		_		
(Temp File Direct	ory				
(Temp File Direct Memory Megaby	ory 🚺	3		Maximum Thread	ds 0 st
✓ Temp File Direct Memory Megaby	ory 🚺 📑 ites 0 📑] key values		Maximum Thread	
Temp File Direct Memory Megaby Eliminate rov	ory Ites 0 🔹 vs with duplicate I] key values		Maximum Thread	ds 0
Temp File Direct Memory Megaby Eliminate rov	ory Ites 0 📑] key values		Maximum Thread	
 Temp File Direct Memory Megaby ☐ Eliminate row 	ory ites 0 vs with duplicate I] key values		Maximum Thread	

Note that NSort orders NULL columns as lower than the lowest non-NULL value.

Multiple Keys

If multiple keys are defined, NSort will sort records primarily on the first key. If it finds records whose first key values are equal, NSort will use the second key to determine the row order. An arbitrary number of keys can be defined. The *Sort Order* attribute displays the order in which the keys will be used, and may be modified to change the key order. The easiest way to define keys is in their intended order as NSort assigns the key sort orders sequentially:

💥 Nsort Transform	ation Editor				×
General Outputs					
Input Column	Data Type	Usage	Sort Order	Comparison Flags	Output Alias
Column 0	DT_STR	Key Ascending 💌	1	Ignore case	Column 0
Column 1	DT_STR	Key Descending 💌	2		Column 1
Temp File Directory Memory Megabytes	y s 0 📑 with duplicate] key values		Maximum Thread	• • •
				ОК	Cancel

Comparison Flags

String data type keys (**DT_STR** and **DT_WSTR**) can also have collation options specified. There are 6 standard SSIS collation options, and two non-standard. These are defined in the *Comparison Flags* box:

Column 0 DT_STR Key Ascending 1 Column 0 Column 1 DT_STR Pass-Thru Collation Options Image: SSIS Compatible Ignore case Ignore case Image: Ignore character width Ignore character width Ignore character width Image: I	Input Column	Data Type	Usage	Sort Order	Comparison Flags	Output Alias
Column 1 DT_STR Pass-Thru Collation Options Image: Column 1 DT_STR Pass-Thru Collation Options Image: Column 1 Image: Column 1 Image: Column 1 Collation Options Image: Column 1 Image: Column 1 Image: Column 1 Image: Column 1 Image: Column 1 Image: Column 1 Image: Column 1 Image: Column 1 <t< td=""><td>Column 0</td><td>DT_STR</td><td>Key Ascending 💌</td><td>1</td><td></td><td>Column 0</td></t<>	Column 0	DT_STR	Key Ascending 💌	1		Column 0
 Ignore case Ignore kana type Ignore nonspacing character Ignore character width Ignore symbols Sort punctuation as symbols Sort punctuation as symbols Non-SSIS Collations BIN2 (stremp, wesemp) BIN (legacy SQL Server binary) BIN (legacy SQL Server binary) OK 	Column 1	DT_STR	Pass-Thru 💌		Collation SSIS Compatib) Options Ile
Temp File Directory Non-SSIS Collations Memory Megabytes Image: Signature Signatu	• (Ignore case Ignore kana Ignore nons Ignore chara Ignore chara Ignore symb Sort punctua	type pacing characte acter width ols ation as symbols
Memory Megabytes 0 1 0 BIN (legacy SQL Server binary) Eliminate rows with duplicate key values 0K Cancel	Temp File Director	у <u> </u>	1		Non-SSIS Collation	ns wesemp)
Eliminate rows with duplicate key values OK Cancel	Memory Megabyte	es 0 🚊			U BIN (legacy SU	L Server binary
	Eliminate rows	with duplicate k	ey values		<u> </u>	Cancel

The 6 standard options require more CPU usage, and must be used when NSort's output is directed to standard SSIS transformations that require sorted input (e.g. Merge Join).

There are also two binary, non-SSIS collations available which are generally faster than the SSIS-compatible collations.. While the *BIN* and *BIN2* collating sequences are supported by the SQL Server database system, they are not recognized by the standard SSIS components which require sorted input (e.g. MergeJoin).

Normally, NSort indicates in its output metadata how its output is sorted so that downstream components can utilize that order. To prevent the erroneous use of BIN or BIN2 NSort output by standard SSIS components, NSort will make the following modifications to its output metadata:

- The key column with a BIN or BIN2 option will not be marked as a sort key.
- Any subsequent keys (keys with a higher sort order) will not be marked as sort keys.
- If the first NSort key uses BIN or BIN2, NSort will indicate its output is not sorted.

Temporary File Specification

The *Temp File Directory* box can be used to specify a temporary file to hold data which does not fit in the NSort transformation's memory. If the data set fits in NSort's virtual memory (including NSort control data), the temporary files are not created. If NSort needs to write data to a temp file and none has been specified, it will create a temp file in the default system temp directory.

If the project data is being read from a file or written to a file, best performance can be obtained by using a temp directory on a disk that is physically distinct from the disk(s) containing input file and output file. Click on the small box at the right of the Temp File Directory box to choose a temp file directory:

XNsort Transformation E	ditor	×
General Outputs		
General Outputs Input Column Data Column 0 DT_S Column 1 DT_S Imput Column 1 D	Browse For Folder Nsort temporary file directory Image: Second stress Image: Second stress	? × Output Alias Column 0 Column 1 •
		OK Cancel

Duplicate Key Elimination

You can check the *Eliminate row with duplicate key values* box to have NSort allow only one row for each set of unique key values.

NSort Threads

The *NSort Threads* box controls the number of sort threads used by the NSort. Leave it at 0 to have NSort use one thread for each processor in the system.

Memory Megabytes

The *Memory Megabytes* box may be used to specify the megabytes of main memory that NSort can use to sort. In the below example, the specified memory size is 200 megabytes:

Input Column	Data Type	Usage	Sort Order	Comparison Flags	Output Alias
Column O	DT_STR	Key Ascending 📃	1	Ignore case	Column 0
Column 1	DT_STR	Key Descending 💌	2		Column 1
(
• Temp File Direct	tory H:\nsort				
 Temp File Direct Memory Megaby 	tory H:\nsort			Maximum Thread	
Temp File Direct Memory Megaby Eliminate rov	tory H:\nsort utes 200 📑 vs with duplicate] key values		Maximum Thread	

For each row that NSort stores in its allocated memory, it will use an extra 12 bytes of memory with 32-bit installations, or 16 bytes on 64-bit installations. For data sets that will not fit in the available physical memory, NSort can write partially-sorted data to temporary files and then read it back.

The specified amount of memory should never be more than 90% of the available physical memory in the machine as this will cause paging activity. NSort temporary file usage is always much faster than Windows paging.

When the SSIS package is run in 32-bit mode, each NSort output is restricted to using no more than 2 gigabytes of memory. A multiple-output NSort component can use more memory, as its memory use can be allocated in multiple sections (one per output). NSort will try to allocate the specified amount of memory. If it can't, it will repetitively multiply the requested amount by .75 until its memory allocation succeeds. The Information Events produced by NSort report the actual amount of memory used.

Parallel Inputs

NSort can accept an arbitrary number of inputs, provided that all inputs contain the same column names and metadata. This can facilitate the parallelism of upstream processing. NSort will combine and sort the rows it receives from its multiple inputs. The number of NSort inputs does not need to be explicitly declared. Rather, new inputs can simply be attached to the NSort component as desired. The behavior of multiple NSort inputs is similar the standard SSIS *Union All* component. However using multiple NSort inputs can be faster than connecting those same streams to a Union All component and then connecting its output to NSort.

The following screen shot was taken at the end of a debug run in the Business Intelligence Design Environment. It shows an NSort component with 2 inputs and 1 output.



Parallel Outputs

NSort can produce multiple, partitioned outputs. This facilitates the parallelism of downstream processing (e.g. Merge Join or file writing) using large numbers of CPUs or disks.

The multiple outputs are partitioned according to the first key defined for the NSort component. Currently, this first key must be either a string (**DT_STR, DT_WSTR**) or integer (**DT_I1, DT_I2, DT_I4, DT_I8, DT_UI1, DT_UI2, DT_UI4, DT_UI8**) data type.

The following two screen shots were taken at the end of debug runs in the Business Intelligence Design Environment. In the first case, NSort's input consists of 1,000,000 rows which are partitioned among 4 outputs. The rows in each output are in sorted order.

Package.dtsx [Design]
🚰 Control Flow 🔱 Data Flow 👌 Event Handlers 🎦 Package Explorer 🌩 Progress
Data Flow Task:
Flat File Flat File Nsort Nsort Nsort Nsort

The second example also sorts 1,000,000 rows, but the input rows now come from 4 different input files. In addition, *Derived Column* components transform the input streams in parallel. Likewise, the 4 output streams of NSort are transformed by *Merge Join* components.



To specify multiple outputs, click on the Outputs tab to view the outputs dialog:

💥 Nsort Transform	ation Editor			×
General Outputs				
Number of Outputs	4 ÷			
Output Partition Bo	oundry Values		Range: 💿 Left	O Right
1	2	3		
grape	mango	strawberry		
	_	_	_	
			OK _	Cancel

The *Number of Outputs* box can be used to specify the output count. Partitioning must currently be specified manually by entering the partitioning key values. The first key of each row is compared to the partitioning values to determine the output for that row. Secondary keys have no effect on the output partitioning, but will affect the row order within each output.

In the above example there are 4 outputs. The three partitioning values are "grape", "mango" and "strawberry". *Range Left* is selected, meaning that rows with keys equal a partitioning value will go in the output to the left of partitioning value. For instance, rows whose first key is "mango" will appear at the end of the second output. If *Range Right* had been selected, these rows would appear at the beginning of the third output.

Information Events

During and after its execution, NSort raises information events that display statistics about its performance. If you run an SSIS project in Visual Studio in debug mode, NSort's information events will automatically be displayed in the *Output* window. If a project is run with *dtexec*, by default only the E (error), W (warning) and P (Progress) events are reported. To also display I (information) events for NSort and other components, the following arguments should be added to the dtexec command line: */reporting ewpi*

The NSort SSIS component utilizes the *libnsort* library, which is not specific to SSIS. The NSort component passes its unsorted input to libnsort, and receives back the rows in sorted order.

An example of NSort component information events follows, color-coded for clarity in this document. It shows:

- the input rows and bytes processed by the NSort component
- libnsort version
- amount of dtexec virtual memory used by libnsort
- input statistics for the libnsort
- temporary file write and read statistics for libnsort only shown if temporary files(s) are actually used
- output statistics for libnsort

```
Info: 2006-01-08 16:50:12.32
  Code: 0x4020CC00
  Source: Data Flow Task NSort
  Description: Input of 380000000 rows, 39520000000 bytes processed
End Info
Info: 2006-01-08 17:15:59.54
  Code: 0x4020CC01
  Source: Data Flow Task NSort
  Description: Nsort library version 3.3.10 (Windows debug 32-bit)
End Info
Info: 2006-01-08 17:15:59.54
  Code: 0x4020CC02
  Source: Data Flow Task NSort
  Description: Usage: 319M bytes of memory
  Nsort I/O
                     Busy Wait MB/sec Xfers
                                                       Bytes
                                                                   Rows
                       50%
                           2.38 33.88 37707 3952000000
                                                              380000000
  Input
  Temporary Writes
    0:\nsort a03100
                       31% 0.00 33.99 75683
                                                 39538757632
  Temporary Reads
    0:\nsort_a03100
                       76% 47.47 25.19
                                         75683 39538757632
                       99% 1517 25.08 37690 3952000000
  Output
                                                              380000000
End Info
```

Limitations

The current version of the NSort SSIS transformation has the following limitations:

- The blob column types are not supported: **DT_TEXT, DT_NTEXT, DT_IMAGE**.
- When using sort keys of type **DT_STR** and **DT_WSTR**, the temporary file data can be larger than expected.
- With multiple outputs, the first key type must be either string (DT_STR, DT_WSTR) or integer (DT_I1, DT_I2, DT_I4, DT_I8, DT_UI1, DT_UI2, DT_UI4, DT_UI8). With DT_I8 and DT_UI8 key types, the output partition boundry values are currently limited to those that fit in the DT_I4 and DT_UI4 types, respectively.
- With SSIS 2008, the DT_DBTIMESTAMPOFFEST column type cannot be used as a key, although it may be passed through the NSort component.