SRP_Sort_Array

Sorts a two dimensional dynamic array.

Syntax

```
NewArray = SRP_Sort_Array(Array, SortInfo, IsList, MajorDelim, MinorDelim, CaseSensitive)
```

Returns

The sorted array.

Parameters

Parameter	Description
Array	The dynamic array to be sorted. **Required.**
SortInfo	An @FM delimited array listing the order columns are sorted and how they are to be sorted. **Required.** See Remarks for details.
IsList	Setting this to 1 Indicates that the Array parameter is in LIST format. Setting it to 0 indicates that it is in ARRAY format. **Optional.** If you omit this parameter, 0 (ARRAY format) will be assumed.
MajorDelim	The delimiter that separates rows if Array is in LIST format or columns if it is in ARRAY format. **Optional.** If you omit this parameter, @FM will be assumed.
MinorDelim	The delimiter that separates elements in within each column (ARRAY format) or row (LIST format). **Optional.** If you omit this parameter, @VM will be assumed.
CaseSensitive	Determines if the sort is done case sensitively (1) or insensitively (0). **Optional.** If you omit this parameter, the sort will be case sensitive.

Remarks

OpenInsight provides sorting capabilities via it's V119 routine. However, it can be cumbersome to prep your data to meet its strict requirements. SRP_Sort_Array gives you standardized sorting and ease of use:

- Most OI programmers are familiar with the LIST and ARRAY properties of the OI Edit Table. SRP_Sort_Array can sort using either format, whereas V119 requires your data to be in a LIST format.
- V119 requires specific delimiters. SRP_Sort_Array lets you specify your data's delimiters.
- V119 often requires you to reorder your data since it always sorts starting at the first column. SRP_Sort_Array lets you specify the order in which columns are sorted.

SRP_Sort_Array is fast. Though it is not technically as fast as a V119 sort by itself, it is many times faster in situations in which you must reorganize your data before and after a V119 sort. So, if your data is already delimited by @RM and @FM; is already appended with an extra @RM; is in LIST format; and you want to sort starting from the first column – then V119 is for you. If however, you usually sort data pulled from an OpenInsight Edit Table or table record, then SRP_Sort_Array will provide a significant increase in performance and simplicity.

SortInfo

The most important component of SRP_Sort_Array is the SortInfo parameter. This is where you specify the order in which columns are sorted, their sort directions, and their sort justifications. The SortInfo parameter is an @FM delimited list of codes. Each code uses the following format:

[A|D][L|R|N]x

[A|D] determines if the column is sorted in the ascending (A) or descending (D) direction. Ascending will be assumed if you don't provide this part of the code. [L|R|N] determines if the column data is left aligned (L), right aligned (R), or converted to numbers (N) during sort. Left aligned sorting is for text sorts, and right aligned sorts are for sorting simple numbers. Numerical sorts actually convert the data to numbers for a more accurate sort, but non-numerical data won't sort well in this mode. If you don't specify any part of this code, then Left aligned sort is assumed. Lastly, the 'x' portion of the code is the column index you wish to sort. You cannot omit this portion of the code since the service will not know which column to sort.

Sort order is determined by the order in which columns appear in the array. Note that you don't have to specify all columns. SRP_Sort_Array will start sorting on the column you specify, and if two rows match on all those columns, it will do a left aligned ascending sort against all unspecified columns in the order they appear in your array. For example, if you specify to sort against columns 2 and 4, and there are 4 columns in your array, then SRP_Sort_Array will sort against column 2, 4, 1, then 3.

Lets look at an example. To sort columns 2 and 4 in a ten column table using the defaults, pass 2:@FM:4. This is the same as saying, "sort text in column 2 in ascending order, and if elements in column 2 are equal, then sort text in column 4 in ascending order." Since the order of SortInfo matters, it should be obvious to you that passing 4:@FM:2 has different results.

As another example, if you wish to sort number columns 3 and 5 in descending order, then pass “DR3”:@FM:“DR5” to SortInfo. Or, you can sort column 3 as ascending and column 5 as descending: “AR3”:@FM:“DR5”. For that matter, we can sort on both text columns and number columns: “AR3”:@FM:“DL2”:@FM:“DR5”:@FM:“AL4”

SRP_Sort_Array is optimized to handle ARRAY formats differently from LIST formats, and as a result, you can get better performance using one format over the other when sorting large amounts of data. Here’s the rule of thumb:

- If you are sorting less than 4 megabytes of data, then use either format since the difference is negligible
- If you are sorting 4 megabytes of data or more, then do one of the following:
    - If you have more rows in your data than columns, then use the LIST format if possible
    - If you have more columns in your data than rows, then use the ARRAY format if possible

Examples

```
* Sort an OI EditTable on column 2
List = Get_Property(@Window:".EDT_TEST", "LIST")
NewList = SRP_Sort_Array(List, "AL2", 1)
Set_Property(@Window:".EDT_TEST", "LIST", NewList)

* Sort a list using custom delimiters
List = "Don,Bakke*Paul,Simonsen*Frank,Tomeo*Bob,Fernandes*Kevin,Fournier"
List = SRP_Sort_Array(List, 2:@FM:1, 1, "*", ",")
// List will be: "Don,Bakke*Bob,Fernandes*Kevin,Fournier*Paul,Simonsen*Frank,Tomeo"
```