Column Sort Views - Design Notes

Subject

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Status** | **Author** | **Description** | **Date** |
| 0.1 | New | M Higginson | Initial Version | 12/01/2009 |
| 0.2 | Review | M Higginson | Applied suggestions from P Rutherford on Apply Button and using User Profile record for default flag. | 16/01/2009 |
| 0.3 | Review | M Higginson | Added Timescales | 19/01/2009 |
| 1.0 | Issues | M Higginson | Adjusted timescales, fixed typo’s | 19/01/2009 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Distribution List

|  |
| --- |
| **Name** |
| Paul Swift Head of Development |
| Mark Roke Product Manager |
| Paul Rutherford Sanity Check |
|  |
|  |

# 1.0 Overview

This design is derived from the Column Sort Views SRS generated by Mark Roke.

The principal driver behind this design was that it is not possible to add indexes into the database for many of the on-screen fields that Product Management wanted to sort on because behind the scenes they were actually calculations taking in many different fields, including system setup flags and exchange rates.

Also this design is more flexible than database indexes as any field, or combination of fields, that we choose to support can be sorted on and we can actually do filtering as well.

The down-side of this design is that is does require some significant changes to the affected lists (see 2.1.x) and a lot of work.

## 1.1 Affected Areas of the Exchequer System

These modifications apply to Enter1.Exe only, the sorting/filtering functionality will not be available outside Enter1.Exe and no API, import functionality or DDF’s will be provided for external access.

## 1.2 Task Breakdown

### The breakdown of tasks is split into three sections, the first section ‘1.2.1 Sort View Implementation’ contains the basic infrastructure changes, the elements common to all lists and implementing Sort Views in the Customer/Supplier Lists. This is the basic starting package and must be completed by one person.

The second section ‘1.2.2 Other Lists’ contains estimates for the other lists documented in the SRS, this cannot start before 1.2.1 is completed but the root items in this section could be programmed in parallel by different programmers. The estimates are based on the timescales predicted for the Customer/Supplier Lists and take into account the relative complexity of the lists and also that we will have learnt lessons from the Customer/Supplier List making things easier.

The third section ‘1.2.3 Other Related Work’ contains other work required in order to ship Exchequer with Sort Views.

Pre-Reqs - Mark Roke to have made a decision on how the graphics are applied (see 2.1) and Creative to have supplied required graphics.

### 1.2.1 Sort View Implementation

|  |  |
| --- | --- |
| New File / File Changes (3.4.2.1 / 3.4.2.2) | 1d |
| User Permissions (3.4.1.1 / 3.4.2.3) | 1.5d |
| Common Routines | (11d) |
| Sort View Options Dialog (2.2)  Basic dialog setup (.5) , add (1), edit (.5), copy (.25), delete (.25) , set as default (.5) | 3d |
| Sort View Configuration Dialog (2.3)  Basic dialog setup (1), Sorting (0.5), Filtering (1), Save (0.5) | 3d |
| Base Sort View Object (3.2.1)  Object Setup (0.25), Sort Storage/Properties (0.5), Filter Storage/Properties (0.5), BuildTempFile (1), CheckSortViewFilter (0.25), EvaluateFilter Methods (0.25), MangleStringValue (0.25), List Repair Functions (2) | 5d |
| Customer/Supplier List (2.1.1) | (9d) |
| Implement Sort View Buttons/Menu Options (2.1.1) and Security | 1d |
| Customer/Supplier List Object (3.2.1/3.2.2) | 2.75d |
| Object Setup (0.25), Sort Definition (0.25), Filter Definition (0.5), CheckListFilter (0.5), GetFilterDataType (0.25), GetFilterValue (0.25), GetSortValue (0.75) |  |
| Customer/Supplier List Changes/Testing/Repair  SV Object Management (0.5), Test/Repair (4) | 4.5d |
| Sort View Graphic on Tabs (2.1) | 0.25d |
| Refresh/Close/Options Menu on Sort View buttons (2.1) | 0.5d |

### 1.2.2 Other Lists

|  |  |
| --- | --- |
| Customer/Supplier Ledger (2.1.2)  Sort View Button/Menu (0.5), List Object (1), List Fixes (3), Graphic (0.25), Popup Menu (0.25) | 5d |
| Job Ledger (2.1.3)  Sort View Button/Menu (0.5), List Object (1.5), List Fixes (1), Graphic (0.25), Popup Menu (0.25) | 3.5d |
| Stock List (2.1.4) | (12d) |
| Stock List Tab (2.1.4.1)  Sort View Button/Menu (.5), List Object (1), List Fixes (3), Graphic (.25), Popup Menu (.25) | 5d |
| Re-Order List Tab (2.1.4.2)  Sort View Button/Menu (.25), List Object (1.5), List Fixes (2), Graphic (.125), Popup Menu (.125) | 4d |
| Stock Take Tab (2.1.4.3)  Sort View Button/Menu (0.25), List Object (1), List Fixes (1.5), Graphic (0.125), Popup Menu (0.125) | 3d |
| Stock Ledger (2.1.5)  Sort View Button/Menu (0.5), List Object (2), List Fixes (1), Graphic (0.25), Popup Menu (0.25) | 4d |

### 1.2.3 Other Related Work

|  |  |
| --- | --- |
| Master Blank/Demo Data (3.4.1.2)  Add SortView.Dat (.5 MH), Add User Permissions (.25 QA), New Schemas (1 CS), Import Data (.25 CS) | 2d  (1.75d Dev, .25d QA) |
| Setup - Upgrade Mods (3.4.1.3)  Replicate SortView.Dat (.5 MH), SQL Conversion (3d MH/CS) | 3.5d |

|  |  |
| --- | --- |
| **Grand Total – All Phases – no contingency or SQL enhancement time included** | **52.5d** |

# 2.0 User Interface

## 2.1 Hosting List Windows

It is advisable that the hosting windows have a mechanism to identify that a Sort View is in use and after some discussion the most popular method we have come up with of indicating that a Sort View is active is to populate the tab containing the list with a graphic (fortunately all the specified lists are in tabs):-

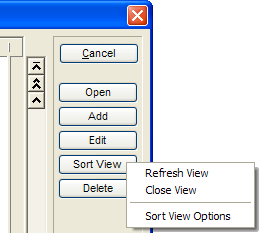


This can work in two different ways (MR to ponder upon and decide):-

1. No graphic as standard and a graphic is shown when a Sort View is active.
2. Two graphics – one to indicate the Sort View is Inactive and one to indicate it is in use.

*NOTE: The graphics are 16x16 pixels and appear to support 24-bit colour, MR to come up with what the graphic should be and to arrange delivery with Creative.*

The Sort View functionality will be accessed via Sort View buttons and popup menu options, if no Sort View is active in the hosting list then the Sort View button will take the user to the Sort View Options dialog, however if a Sort View is currently active in the list then the Sort View button will popup a menu with options to Refresh the current view, to Close the current view and to go into the Sort View Options dialog.



NOTE: The Sort View functionality is only available if the appropriate user permission is set.

The following sections list the windows in Exchequer that will be extended to host the Sort View functionality together with any information specific to that window.

### 2.1.1 Customer and Supplier Lists (CustLst2.Pas)

A new ‘Sort View’ button will be inserted between the existing ‘Ledger’ and ‘History’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘Ledger’ and ‘History’ menu options in the popup menu.

### 2.1.2 Customer and Supplier Records - Ledger (CustR3U.Pas)

For the ‘Ledger’ tab only a new ‘Sort View’ button will be inserted between the existing ‘Find’ and ‘Switch’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘Find’ and ‘Switch’ menu options in the popup menu.

### 2.1.3 Job Record – Ledger (JobMn2U.Pas)

For the ‘Ledger’ tab only a new ‘Sort View’ button will be added after the existing ‘Links’ button and a new ‘Sort View’ menu option will be inserted after the existing ‘Links’ menu options in the popup menu.

### 2.1.4 Stock List (StkLstU.Pas)

### 2.1.4.1 Stock List – Stock List Tab

For the ‘Stock List’ tab only a new ‘Sort View’ button will be inserted between the existing ‘Sort’ and ‘Build’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘Ledger’ and ‘Build’ menu options in the popup menu.

Whilst a Sort View is being displayed in the Stock List tab the ‘Sort’ button and ‘Filter List by Location’ menu option off the ‘Location’ button should be disabled as their functionality is not compatible with Sort Views.

### 2.1.4.2 Stock List – Re-Order Tab

For the ‘Re-Order’ tab only a new ‘Sort View’ button will be inserted between the existing ‘History’ and ‘Notes’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘History’ and ‘Notes’ menu options in the popup menu.

Whilst a Sort View is being displayed in the Re-Order tab the ‘Filter’ button and ‘Filter List by Location’ menu option off the ‘Location’ button should be disabled as their functionality is not compatible with Sort Views.

### 2.1.4.3 Stock List – Stock Take Tab

For the ‘Re-Order’ tab only a new ‘Sort View’ button will be inserted between the existing ‘History’ and ‘Notes’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘History’ and ‘Notes’ menu options in the popup menu.

Whilst a Sort View is being displayed in the Stock Take tab the ‘Sort’ button and ‘Filter List by Location’ menu option off the ‘Location’ button should be disabled as their functionality is not compatible with Sort Views.

Please note that the Freeze and Process buttons will not respect the Sort View and will process all records as per their usual functionality.

### 2.1.5 Stock Ledger (StockU.Pas)

For the ‘Ledger’ tab only a new ‘Sort View’ button will be inserted between the existing ‘Find’ and ‘Print’ buttons and a new ‘Sort View’ menu option will be inserted between the existing ‘Find’ and ‘Print’ menu options in the popup menu.

NOTE: The code controlling the Links button seems a little dodgy as it is disabled but visible and is not being moved to be contiguous with the other buttons.

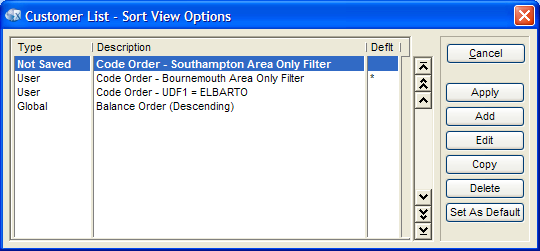
## 2.2 Sort View Options Dialog

The Sort View Options dialog will be displayed when the users click the Sort View buttons and menu options as defined in section 2.1.

The list of Sort Views available for opening will be taken from SortView.Dat and will list Sort Views for the current logged in user first in Description order, followed by the global views in Description order.

If an unsaved Sort View is currently running then this will be inserted first in the list with a type of ‘Not Saved’ and automatically selected.

If a Sort View is already being displayed in the host list then the matching Sort View in the list will be shown in Bold and automatically selected.



This dialog is context sensitive and lists the available options for the user:-

Cancel – Always available – closes the Sort View Options dialog and returns to the hosting list without making any changes.

Apply – Always available – causes the currently selected Sort View to be built and applied to the host list, any previously active Sort View will be closed automatically. Should be disabled if the currently selected view is the already being used in the host list.

Add - Available if the Add Sort View user permission is enabled. Allows the user to create a new Sort View using the Sort View Configuration dialog.

Edit – Available if the Edit Sort View user permission is enabled. Allows the user to change the properties of the currently selected Sort View using the Sort View Configuration dialog.

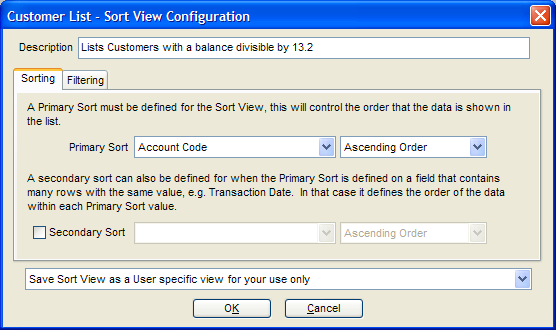
Copy - Available if the Add Sort View user permission is enabled. Allows the user to create a new Sort View taking the currently selected Sort View as a template.

Delete - Available if the Delete Sort View user permission is enabled. Allows the user to delete a Sort View after confirming that they want to delete it.

Set As Default – Available if the currently selected Sort View has been saved. Marks the currently selected Sort View as the users default. This flag is stored on the User Profile record in MLocStk.Dat.

## 2.3 Sort View Configuration Dialog

This dialog is used for adding and editing the detail of a Sort View from the Sort View Options dialog.



*Description* – A brief (100 character) description so that users can identify Sort Views in the list of available Sort Views in the Sort View Options dialog.

*Primary Sort Field* – This drop-down list contains a list of the fields from the host list which can be sorted upon. The Primary Sort is mandatory.

*Primary Sort Order* – Allows the sorting order to be switched between Ascending and Descending. This should default to Ascending.

*Secondary Sort Checkbox* – Tick this to enable the optional secondary sort.

*Secondary Sort Field* – Only available if the Secondary Sort Checkbox is ticked - this drop-down list contains a list of the fields from the host list which can be sorted upon, it should exclude the field selected in the Primary Sort. See section 3.2 for details on which fields are available for each list.

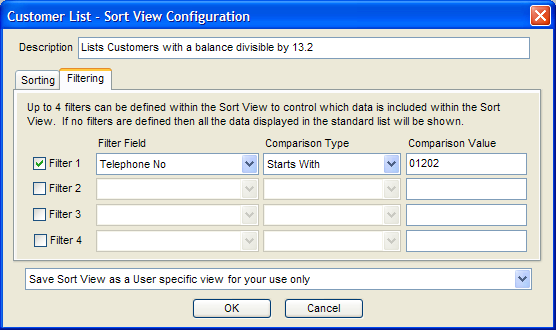
*Secondary Sort Order* – Only available if the Secondary Sort Checkbox is ticked - Allows the sorting order to be switched between Ascending and Descending. This should default to Ascending.

*Save Sort View* – This is only available when adding (including Copy) a new Sort View or if the Sort View hasn’t been saved and is running in memory. Three options are available:-

Do not save this Sort View for future use – The Sort View details are not saved to file and run in memory only, this mode is intended to be used for one-off queries to avoid clogging up the Sort View list.

Save Sort View as a User specific view for your use only – Saves the Sort View against the current user.

Save Sort View as a Global view for all users to use – Saves the Sort View as a global Sort View which will be accessible to all users.



Up to four filters can be defined for each Sort View, to be included within the results the data must pass all enabled filter conditions.

*Filter Checkbox*  – The user should tick this checkbox to enable the filter.

*Filter Field -* Only available if the Filter Checkbox is ticked - this drop-down list contains a list of the fields from the host list which can be filtered upon. See section 3.2 for details on which fields are available for each list.

*Comparison Type* – Only available if the Filter Field has been set – this list contains the comparisons available for the selected filter field which is dependant on the data type:-

|  |  |  |
| --- | --- | --- |
|  | String Fields | Float Fields |
| Equals | Yes | Yes |
| Not Equals | Yes | Yes |
| Less Than | Yes | Yes |
| Less Than or Equal | Yes | Yes |
| Greater Than | Yes | Yes |
| Greater Than or Equal | Yes | Yes |
| Starts With | Yes | n/a |
| Contains | Yes | n/a |

*Comparison Value* - Only available if the Filter Field has been set – this is the value that the user is comparing against. It should be a basic edit field for string fields (Max Length = 100) and a numerical edit field (max 6 dp) for floating point type fields.

# 3.0 Boring Technical Details

## 3.1 Overall Concepts

The basic master plan behind this design is to enable custom sorting and filtering on the specified standard lists by building a temporary file and modifying the list to run from that instead of from the standard data.

The temporary file will only contain the keys required to sort the data correctly and a key allowing the appropriate standard data record to be loaded, e.g. OurRef, Line Folio + Abs Line No, Stock Code or Stock Folio, etc…

The UI will be generic and will access any list specific details required, e.g. List Name, Sorting Fields, etc…, from a list specific class which descends from a common base class implementing common functionality.

If this is successfully done then the bulk of the work required will be within the lists themselves making them Sort View aware and repairing the functionality broken by switching the data file the list runs from. It is expected that common routines can be written into the common base class to make the tasks of repairing the lists easier.

**3.1.1 SQL Edition**

This design should work unchanged in the SQL Edition, however to improve performance it may be desirable to modify the routines which build the temporary file to use either a Pre-Fill Cache or possibly direct SQL to get the data.

The SQL Edition performance should be evaluated once each Sort View has been implemented and a decision as to whether additional performance work needs to be done should be made by Product Management.

**3.1.2 Files**

This spec is in the v6.01 repository in the \Entrprse\DOCS\v6.01\Column Sorting\ directory, a Column Sorting subdirectory off that contains the Delphi Forms, Visio diagrams etc… used to write this spec.

## 3.2 Sort View List Classes

### 3.2.1 Base Class

The base class is going to provide the common interface that the UI and build routines will use, it is also expected that common routines can be placed here to be used by the list windows themselves for loading standard records from sort records, etc....

**3.2.1.1 Properties**

Example properties:-

Property Enabled : Boolean

This indicates whether a Sort View is currently running, the host list code will use this a lot to determine how to work.

*Property FilterCount : LongInt;*

*Propery Filters[Index: LongInt]:* TSortViewFilterInfoRecType;

These properties allow the UI to access the filtering information.

Property ListDesc : ShortString

This should be set in the Constructor of the list specific descendants to a description of the list, e.g. Customer List, for use on-screen in the Sort View Options and Configuration dialogs.

Property SortCount : LongInt;

Property Sorts[Index: LongInt]: TSortViewSortInfoRecType;

These properties allow the UI to access the sorting information.

Property SortViewDesc : ShortString;

This is the description of the Sort View as entered by the user.

Property HostListSearchKey : ShortString;

Property HostListFileNo : LongInt;

Property HostListIndexNo : LongInt;

These properties tell the BuildTempFile routine which file to access, which index to use and what SearchKey to match against.

**3.2.1.2 Methods**

Example Methods:-

Function BuildTempFile : Boolean; Virtual;

This function builds the temporary file using the HostListFileNo, HostListIndexNo and HostListIndexNo to access the data, something like this:-



If the method required for loading a list is incompatible with this, e.g. needs Extended Btrieve, then this method should be overridden in the list specific class.

Function CheckListFilter : Boolean; ABSTRACT;

This abstract method will be implemented in the list specific classes to check whether the current record would be shown in the Host List and returns True if it should be included. Ideally this method should re-use any pre-existing list filter method to avoid any possibility of mismatches between the two, but this may not be practical in which case comments should be placed in both routines reminding us to keep the other method in sync.

Function CheckSortViewFilter : Boolean;

This method will check the specified Sort View filters against the current record to determine whether it should be included within the Sort View. The data must pass ALL the enabled filters in order to be included.

Result := True;

For iFilter := Low(svrFilters) To High(svrFilters) Do

Begin

If svrFilters[iFilter].svfEnabled Then

Begin

Case GetFilterDataType (svrFilters[iFilter].svfFieldId) Of

fdtString : Result := Result And EvaluateStringFilter(svrFilters[iFilter]);

fdtFloat : Result := Result And EvaluateFloatFilter(svrFilters[iFilter]);

Else

Raise Exception.Create (‘CheckSortViewFilter – Invalid Data Type’);

End; // Case GetFilterDataType

End; // If svrFilters[iFilter].svfEnabled

If (Not Result) Then

Break;

End; // For iFilter

*Note: The unused and spare filters will have their svfEnabled flags set to false.*

TSortViewFilterDataType = (fdtString, fdtFloat);

Function GetFilterDataType (Const FilterFieldId : LongInt) : TSortViewFilterDataType; Abstract;

This abstract method will be implemented in the list specific classes to return the data type for the specified filter field.

Function EvaluateStringFilter(Const Filter : TSortViewFilterInfoRecType) : Boolean;

Function EvaluateFloatFilter(Const Filter : TSortViewFilterInfoRecType) : Boolean;

These methods are used to evaluate the filter, they will use GetFilterValue methods to retrieve the data and then perform the appropriate comparison as defined by Filter.svfComparison against the value specified in Filter.svfValue.

Function GetFilterValue (Const FilterFieldId : LongInt) : ShortString; ABSTRACT;

Function GetFilterValue (Const FilterFieldId : LongInt) : Double; ABSTRACT;

These abstract methods will be implemented in the list specific classes to return the current value for the specified filter field.

Function MangleSortStringValue (Const SortString : ShortString) : ShortString;

Due to limitations in the SQL Emulator this function will be used by the GetSortValue method in the descendant classes to ‘adjust’ string values so that they sort correctly when the sort field is defined as descending.

0 -> 255

1 -> 254

…

127 -> 128

128 -> 127

…

254 -> 1

255 -> 0

After converting the SortString to uppercase to avoid case sensitivity issues each character in the string will be modified to emulate a descending sort:-

Result := ’’;

For iChar := 1 To Length(SortString) Do

Result := Result + Chr(255 – Ord(SortString[iChar]));

Function GetSortValue (Const Sort : TSortViewSortInfoRecType) : ShortString; ABSTRACT;

This abstract method will be implemented in the list specific classes to return a string version of the field to sort on, unfortunately due to limitations in the SQL Emulator this function also has to ‘adjust’ the sort strings to emulate a descending index.

*String Sorts*

For String fields with an ascending sort this can simply be the string, for descending sorts the string will have to be adjusted by MangleSortStringValue in order to sort correctly.

*Floating Point Sorts*

For Floating Point fields with an ascending sort we can simply use the floating point value, but for descending sorts the value will have to be multiplied by -1 to emulate a descending sort correctly.

Also all numbers must be formatted using a common format that excludes commas and has leading and trailing zeros so that numbers are compared correctly e.g. 000000001234.560000.

An addition problem with that is that the ‘+’ and ‘-‘ characters have ASCII values of 43 and 45 respectively so ‘+’ will sort before ‘-‘ when sorting in ascending order – which is the wrong order. A workaround is to use put negative numbers numbers in brackets and have a leading ‘+’ on zero/positive numbers:-

(000000001234.560000)

+000000000000.000000

+000000001234.560000

*NOTE 1: : Don’t forget to pad the return values to the full length of the sort fields in the record structure.*

*NOTE 2: Don’t forget to take into account the variable number of decimal places on Cost, Sales and Quantity fields.*

### 3.2.2 Customer and Supplier List Class

This class will descend from the base class and add Customer & Supplier List specific details, it is expected that a flag will be passed into the Constructor to configure it as either Customer or Supplier specific.

**3.2.2.1 Sort Fields**

The following columns will be available for sorting:-

Account Code

Company Name

Balance

Telephone No

See CustLst2.Pas for details of calculating individual columns.

**3.2.2.1 Filter Fields**

The following fields on the Customer/Supplier record will be available for filtering:-

Area

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.3 Customer and Supplier Ledger Class

This class will descend from the base class and add Customer & Supplier Ledger specific details, it is expected that a flag will be passed into the Constructor to configure it as either Customer or Supplier specific.

**3.2.3.1 Sort Fields**

The following columns will be available for sorting:-

Our Ref

Date

Amount Settled

Outstanding

Original Value

Your Ref

Date Due

See CustLst2.Pas for details of calculating individual columns.

**3.2.3.1 Filter Fields**

The following fields on the Transaction Header record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.4 Job Ledger Class

This class will descend from the base class and add Job Ledger specific details.

**3.2.4.1 Sort Fields**

The following columns will be available for sorting:-

Account Code

Our Ref

Date

Analysis

Hours/Qty

Cost

Charge

Uplift

See JobMn2U.Pas for details of calculating individual columns.

**3.2.4.1 Filter Fields**

The following fields on the Transaction Header record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

The following fields on the Transaction Line record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.5 Stock List – Stock List Tab Class

This class will be used for the Stock List Tab within the Stock List window. It will descend from the base class and add Stock List Tab specific details.

**3.2.5.1 Sort Fields**

The following columns will be available for sorting:-

Stock Code

Description

In Stock Qty

Free Stock Qty

On Order Qty

Supplier

See StkLstU.Pas for details of calculating individual columns.

**3.2.5.1 Filter Fields**

The following fields on the Stock record will be available for filtering:-

Cost Centre Code

Department Code

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.6 Stock List –Re-Order Tab Class

This class will be used for the Re-Order Tab within the Stock List window. It will descend from the base class and add Re-Order Tab specific details.

**3.2.6.1 Sort Fields**

The following columns will be available for sorting:-

Stock Code

Free Stock Qty

On Order Qty

Supplier

Min Stock Qty

Need Qty

See StkLstU.Pas for details of calculating individual columns.

**3.2.6.1 Filter Fields**

The following fields on the Stock record will be available for filtering:-

Cost Centre Code

Department Code

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.7 Stock List – Stock Take Tab Class

This class will be used for the Stock Take Tab within the Stock List window. It will descend from the base class and add Stock Take Tab specific details.

**3.2.7.1 Sort Fields**

The following columns will be available for sorting:-

Stock Code

Description

In Stock Qty

Bin Location

See StkLstU.Pas for details of calculating individual columns.

**3.2.7.1 Filter Fields**

The following fields on the Stock record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

### 3.2.8 Stock Ledger Class

This class will be used for the Stock Ledger Tab within the Stock Record window. It will descend from the base class and add Stock Ledger specific details.

**3.2.8.1 Sort Fields**

The following columns will be available for sorting:-

Account Code

Our Ref

Date

Qty In

Qty Out

Allocated

On Order

Allow WOR

Issued Word

Sales Ret

Purch Ret

Unit Price

See StockU.Pas for details of calculating individual columns.

**3.2.8.1 Filter Fields**

The following fields on the Transaction Header record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

The following fields on the Transaction Line record will be available for filtering:-

User Defined Field 1

User Defined Field 2

User Defined Field 3

User Defined Field 4

## 3.3 Database Changes

### 3.3.1 New Table - <CompanyDirectory>\Misc\SortView.Dat

This new table is required to store the Sort View details.

The following record structure is proposed:-

// Enumeration for defining location that Sort View applies to

TSortViewListType = (svltCustomer, svltSupplier, svltCustLedger, svltSuppLedger,

svltStockList, svltStockReOrder, svltStockTake,

svltStockLedger, svltJobLedger);

// Enumeration for controlling how filters are compared against values

TSortViewFilterComparisonType = (svfcEqual, svfcNotEqual,

svfcLessThan, svfcLessThanOrEqual,

svfcGreaterThan, svfcGreaterThanOrEqual,

svfcStartsWith, svfcContains);

// Sub record used for storing the sort details (56 Bytes)

TSortViewSortInfoRecType = Record

svsEnabled : Boolean;

svsFieldId : LongInt; // Unique Id No within each list for sorted field

svsAscending : Boolean;

svsSpare : Array [1..50] Of Byte; // For future use

End; // TSortViewSortInfoRecType

// Sub record used for storing the filter details (207 Bytes)

TSortViewFilterInfoRecType = Record

svfEnabled : Boolean;

svfFieldId : LongInt; // Unique Id No within each list for filtered field

svfComparison : TSortViewFilterComparisonType;

svfValue : String[100];

svfSpare : Array [1..100] Of Byte; // For future use

End; // TSortViewFilterInfoRecType

// Main Sort View record (2917 bytes)

SortViewRecType = Record

svrViewId : LongInt; // Unique Id automatically allocated on creation

svrUserId : String[30]; // Blank for Global, else Exchequer User Id

svrListType : TSortViewListType; // Exchequer list this Sort View applies to

svrDescr : String[100]; // On-screen description of view

svrSorts : Array [1..4] Of TSortViewSortRecType; // 2 spare for future use

svrFilters : Array [1..8] Of TSortViewFilterInfoRecType; // 4 spare for future use

svrSpare : Array [1..900] Of Byte; // For future use

End; // SortViewRecType

Index 0: svrViewId – Unique – for direct loading/updating of Sort View records

Index 1: svrListType + svrUserId + svrDescr - Unique – for Sort View Options list

*Note 1: Deliberately used String[30] for svrUserId to allow for the User Id to be increased in length in future versions or should we start linking into the Windows NT Security system at any point..*

*Note 2: Deliberately put 50 spare in Sort record and 100 spare in filter record as I view the filters as more likely to be extended. Under Btrieve the compression should squash that down to a few bytes of disk space anyway and under SQL the server should always have lots of free disk space for us to depopulate ☺*

*Note 3: Deliberately put 2 spare sort elements and 4 spare filter elements into the arrays for future use. It is a lot easier to add them now than either convert the file later or modify the code to work off several different arrays. However if the additional sorts are used then the structure of the temporary file will need to be changed.*

### 3.3.2 New Temporary File - < CompanyDirectory>\Swap\SV######.Dat

This temporary file will be created by the Sort View BuildTempFile process as required to store the sort information, it will only contain the sort fields and enough information to load the original data, e.g. OurRef/Folio/etc….

SortViewTempRecType = Record

svtFolio : LongInt; // Unique, sequential folio number

svtField1 : String[100]; // Primary sorted field

svtField2 : String[100]; // Secondary sorted field (optional)

// Index fields to actual data – usage will depend on which list is being used

svtSourceDataStr : ShortString;

svtSourceDataFolio : LongInt;

svtSourceLineNo : LongInt;

End; // SortViewTempRecType

Index 0: svtFolio

Index 1: svtField1 + svtField2 + svtFolio

In order to guarantee unique indexes and the ability to re-load specific records the svtFolio field will be set to a unique number within the temporary file (start at one and increment for every record added).

The two sorting fields, svtField1 and svtField2, will be set to sort compatible versions of the selected fields, padded out with spaces to the full length of the fields. See GetSortValue and MangleSortStringValue for details on how these fields will be set.

## 3.4 Related Work

This section details work required to implement and support Sort Views that is not detailed in other sections of this document.

### 3.4.1 Installation

This section covers work required to support Sort Views within the Installation area.

3.4.1.1 GEUpgrde.Dll

This DLL is used by the Upgrade process to add new user permissions records and is also used outside of the installer through TestUpgr.Exe to update the master data files.

3.4.1.2 Master Blank/Demo Data

For Pervasive add Misc\SortView.Dat as standard, update existing files with new user permissions records (QA would normally handle this).

For SQL generate new schemas for Misc\SortView.Dat and Swap\SV\*.Dat, update import data to include new user permissions records.

3.4.1.3 Upgrade

For the Pervasive Edition a process needs to be put in place to add a blank SortView.Dat into all companies – might be able to use the existing Replication process.

For the SQL Edition we need to put implement the conversion process so that the new SortView.Dat table is created and the new DefaultSortView column is added for MLocStk.Dat.

### 3.4.2 Exchequer

This section covers work required to support Sort Views within Exchequer itself.

3.4.2.1 Create/Open files

Add new record structures and file definitions for SortView.Dat and SV\*.Dat.

Extend Open\_Files to open SortView.Dat.

For SQL generate new schema for Misc\SortView.Dat and Swap\SV\*.Dat.

3.4.2.2 User Profile Record

Modify User Profile record (tPassDefType in VarRec2U.Pas) in MLocStk.Dat to include DefaultSortView.

For SQL extend the schema for MLockStk.Dat for new field.

3.4.2.3 Update User Permissions dialog

Update User Permissions Tree to include a new Sort View section and the following permissions:-

Access Sort Views

Add Sort Views

Edit Sort Views

Delete Sort Views