# TGenReport (ReportU.Pas) – Base Reporting Object

This is the basic reporting object in Exchequer, the other reporting objects are all descendants of this object or descendants of other descendants. It descends from TThreadQueue in ExBtTH1U.Pas

*NOTE: The reporting system uses legacy Object types rather than Classes.*

### TGenReport Methods

#### Function Start : Boolean

This function is typically called after the reporting object has been created, but before it has been added into the thread queue.

After calling *GetReportInput* it displays the Printer Selection Dialog and then, if not already open, opens the global reporting files (ExBtTh1U – RepExLocal : TdPostExLocalPtr) which are used by all reports for data access. A reference to these is stored internally in MTExLocal and finally *InitRep1* is called.

This function is overridden in Report8U.

#### Procedure Process; Virtual

This function is called by the thread controller to start the printing process.

It updates the printer details on the internal TReportFiler object, updates the progress information and then calls the TReportFiler.Execute method to print the report, this will result in the *RepPrint* method being called.

This function is overridden in most of the reporting objects, some of them just call inherited whilst others use it to build temporary files which the report is based upon.

#### Procedure Finish; Virtual

This function is called by the thread controller once the *Process* call has returned.

This method prints the report by filling a TPrintParam structure with the printjob details and passing the address of it as a LongInt into LThPrintJob which is actually a reference to TBackThread.RunPrintJob (ExThrd2U.Pas). This routine accesses the MTMonRecs records on the Thread Controller to wait until any previous print jobs are complete before setting this print job to print. These details are then picked up by TBackThread.PrintPrintJob which posts a WM\_FormCloseMsg / 98 message to EParentU with the LongInt-Pointer-TPrintParam. This is picked up by TMainForm.WMFormCloseMsg (EParentU.Pas) which then calls Print\_File which then calls TBackThread.Print\_ThJob (ExThrd2U.Pas) which then calls PrintFileTo (SbsForm.Dll). This means that the actual output of the report is running in the main thread and not the thread that generates the temporary file.

#### Function GetReportInput : Boolean; Virtual

Misleadingly named – this function has nothing to do with the popup dialogs asking for criterion on the front of a lot of the reports.

Generally this function seems to be involved with setting up report titles and initialising the various data access flags.

This function is overridden in all reporting objects, all variants appear to return BOn.

#### Function InitRep1 : Boolean

Configures the internal RepFiler1 : TReportFiler instance.

#### Procedure RepSetTabs; Virtual

Sets the tab positions for the report which controls the columns widths and positions.

Overridden in all decendant reporting objects.

#### Procedure RepPrint(Sender : TObject); Virtual

Main method for generating the report, does data access and filtering checks.

This function is overridden in Report4U, Report5U, Report6U, Report7U, ReportAU, ReportBU.

#### Procedure RepPrintHeader(Sender : TObject); Virtual

Prints the Page Header, calls *RepPrintPageHeader*.

#### Procedure RepPrintPageHeader; Virtual

Prints the column titles, called by *RepPrintHeader*.

Overridden in all decendant reporting objects.

#### Function IncludeRecord : Boolean; Virtual

Called by RepPrint to check the current record is wanted. Must be overridden in each descendant of TGenReport unless they have their own RepPrint method.

#### PrintReportLine

Called to output a report line for the current record.

Must be overridden in each descendant of TGenReport unless they have their own RepPrint method.

#### SetReportDrillDown

Adds the Drill-Down information

# Adding SQL Support

The following section is based on a straightforward report such as a Customer List which is printed in a straightforward manner. Those reports which fall out of this category by building temporary files or by using other non-standard behaviour will need to be dealt with on a case-by-case basis but might still be able to apply the same principles in a different manner.

## Using a Custom Pre-Fill Cache

This appears to be the most straightforward method of optimising reports as it should require the minimum of changes to the code as the data will still be accessed using the normal Btrieve commands and will return our standard record structures.

With a Custom Pre-Fill Cache we issue a caching command which specifies a SQL-style Where clause to identify the data being cached, e.g.

WHERE acCustSupp = ‘C’

This means that each read of the database will return a record that we need to print. Presumably we will need to get the Data Team to validate our where clauses to ensure they are optimal, Catherine’s indexing mods may affect which fields we should be using.

In addition the Custom Pre-Fill Cache is being extended currently to allow us to specify a subset of a table’s columns to return, currently every read we do will cause every column to be read from the table for each row which is very inefficient.

Reducing the number of columns we return will reduce the load on the SQL Server and reduce the network traffic. An example is the Customer List Report which has 7 columns whilst the CustSupp Table the report runs from has 91 columns, a simple test of this in SQL Server Maintenance Studio using Demo Data showed the time taken to do the query dropped from 0.443 seconds for all 91 columns to 0.003 seconds for 7 columns (0.7%) potentially other gains will be made on top of that as there will be less information to send over the network and the emulator should have significantly less work to do processing the columns.

Unfortunately it will take time to identify which columns are required by each report as the report code will need to be manually analysed in detail.

Please note that the Pre-Fill Cache does not hold the data in-memory, it just holds the key information needed to load the data which is then read from the database row by row as we access the cache.

### Implementing a Pre-Fill Cache

Requirements

1. A method of turning on SQL Pre-Fill Caching for the report
2. A method of specifying the Where clause for the report
3. A method of specifying the columns to be returned
4. A method of building the cache using CICOREUseCustomPrefillCacheEmulator.
5. Switching the Btrieve calls to use the cache using CICOREUsecustomPrefillCacheForNextCall.
6. A method of destroying the cache using CICOREDropCustomPrefillCacheEmulator.

Note: report objects often print several different reports and they might not all use the same method.

1. TGenReport – Declare a UseSQLPrefillCache : Boolean variable.
2. TGenReport.Create – Initialise UseSQLPrefillCache to False.
3. In GetReportInput across all reporting objects set UseSQLPrefillCache to True if required.
4. TGenReport – Declare a SQLWhereClause : ANSIString variable.
5. In GetReportInput across all reporting objects set SQLWhereClause if required.
6. TGenReport – Declare a SQLColumns : ANSIString variable.
7. In GetReportInput across all reporting objects set SQLColumns if required.
8. In TGenReport.RepPrint build the cache by calling CICOREUseCustomPrefillCacheEmulator if UseSQLPrefillCache is TRUE.
9. In TGenReport.RepPrint call CICOREUsecustomPrefillCacheForNextCall prior to the LFind\_Rec calls to get them to use the Pre-Fill Cache.
10. In TGenReport.RepPrint use the UseSQLPrefillCache flag to prevent the LPresrv\_BTPos calls from executing under SQL. Any data access outside of the RepPrint loop will not be using the Prefill Cache and so will not affect the positioning.
11. In TGenReport.RepPrint drop the cache by calling CICOREDropCustomPrefillCacheEmulator if UseSQLPrefillCache is TRUE.

## Using Stored Procedures

It should also be possible to use a Stored Procedure to return a RecordSet to the application, this will probably require direct access to SQL from the report code and require major changes to the report to allow it to run from the recordset instead of our standard record structures.

Due to the scope of changes required to implement this option for each report I do not plan to investigate this option any further unless the Pre-Fill Cache option does not provide the necessary performance.

# ReportU.Pas

TGenReport is the base object for the reporting object directly prints the following reports:-

Customer List (Mode=0)

Supplier List (Mode=1)

General Ledger Code List (Mode=4)

Customer VAT Registration No List (Mode=9)

Supplier VAT Registration No List (Mode=10)

TADebReport descends from TGenReport

Receipts Due

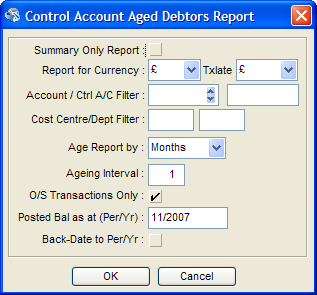
Payments Due

Aged Debtors Report

Aged Creditors Report

TMDCReport descends from TADebReport but builds its own temporary file which the report is run from. The routine to build the temporary file (Build\_MDCAgedDoc) can probably use a pre-fill cache to avoid some of the filtering checks that it is currently doing. NOTE: There are some extended Btrieve operations going on in some modes.

Multiple Debtors/Creditors versions of Aged Debtors/Creditors



TStaReport descends from TGenReport and appears to be used to print a series of forms:-

Statement Run (mode = 2)

Debt Chase Letters Run (mode = 3)

Trading Ledger Run (mode = 5)

Account Details Run (mode = 8)

Account Label Run (mode = 10)

Stock Label Run (mode = 11 -> 14)

### Public Procedures

AddGenRep2Thread – Used to print reports printed by TGenReport.

AddADebRep2Thread - Used to print reports printed by TADebReport

AddAMDCRep2Thread – Used to print reports printed by TMDCReport

AddStaRep2Thread – Used to print reports printed by TStaReport.

# REPORT2U.PAS

TPostReport descends from TGenReport

Posting Report

AddPostRep2Thread

# REPORT3U.PAS

TDocReport descends from TGenReport:-

Document Analysis Report

Sales Daybook Report

Purchase Daybook Report

Sales Receipts - Daybook Report

Purchase Payments - Daybook Report

Sales Order Daybook Report

Purchase Order Daybook Report

Nominal Transfer - Daybook Report

Reconciliation Report (mode = 1)

General Ledger History Report (mode = 2)

Audit Trail (mode = 3)

Document Posting Report (mode = 5)

Document Audit Trail (mode = 7)

Customer Trading History Report (mode = 8/11 (summary))

Supplier Trading History Report (mode = 9/12 (summary))

It appears that these reports are no longer directly printed but are wrapped and used by the reports in RepCCShu.Pas.

The AddDocRep2Thread procedure merely calls AddCCDepReportThread in RepCCShu.Pas.

# REPORT4U.PAS

TNOMReport descends from TGenReport

Trial Balance

Profit & Loss

Balance Sheet

S+A of Funds

AddNomRep2Thread

# REPORT5U.PAS

TCCDepReport descends from TNomReport (Report4U.Pas)

Cost Centre and Department Analysis

AddCCDepRep2Thread

# REPORT6U.PAS

TStkReport descends from TGenReport

Sales Analysis

Comparative Sales Analysis

Price List

AddStkRep2Thread

# REPORT7U.PAS

TCCStkReport descends from TStkReport (Report6U.Pas)

Cost Centre/Department Sales Analysis

AddCCStkRep2Thread

# Report8U.pas

TVATRReport descends from TGenReport

VAT Returns

AddVATRRep2Thread

# REPORT9U.PAS

TCVATReport descends from TGenReport

Cash Accounting Payment & Receipt Report

AddCVATRep2Thread

# REPORTAU.PAS

TECVATReport descends from TGenReport

VAT - EC Sales List

AddECVATRep2Thread

# REPORTBU.PAS

TISVATReport descends from TGenReport

Intra EC Trade Statistics

AddISVATRep2Thread

# REPORTCU.PAS

TPBrkReport descends from TGenReport

Stock Sales by Product Report

Stock Sales by Customer Report

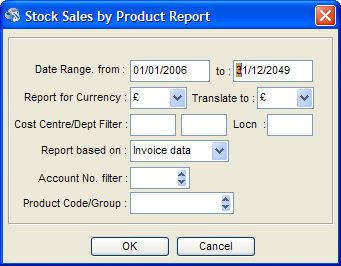
Stock Reconciliation Report (mode = 15)

Stock Reconciliation Summary Report (mode = 16)

Finished Goods Reconciliation Report (mode = 15)

Finished Goods Reconciliation Summary Report (mode = 16)

AddProdRep2Thread



# REPORTDU.PAS

TSListReport descends from TGenReport

Stock Re-Order Report (mode=1)

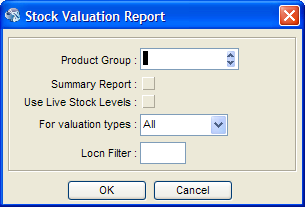
Stock Take Report (mode=2)

Stock List (mode=3)

Stock Aging Report as at (mode=44)

Stock Valuation Report (Mode=4)

AddSListRep2Thread



## Data Access

The Aging and Valuation variants of this report runs off a temporary file all other variants run off Stock.Dat.

# REPORTEU.PAS

TSHistReport descends from TGenReport

Stock History Report (mode = 2)

Stock History Summary Report (mode = 1)

AddSHistRep2Thread

# REPORTFU.PAS

TSKitReport descends from TGenReport

Kitting Report

Kitting Shortages Report

Sales Back Order Report

Procedures - AddSKitRep2Thread

# REPORTGU.PAS

TSShortReport descends from TGenReport

Stock Shortages Report

Expected Stock Delivery Report

Procedures - AddSShortRep2Thread

# REPORTHU.PAS

TMemoReport descends from TGenReport and basically prints the contents of a stringlist which is built up externally and passed into the report. It appears to be used for the following:-

Batch Receipt/Payment Status

Update Build Costing Errors

Process Bank Matching Entries Status Report

Error Log

Password Access Listing

Module Release

EDI Export

Obviously any optimization for these reports will need to be done on the routine(s) building the stringlists.

Procedures - AddMemoRep2ThreadMode / AddMemoRep2Thread

# REPORTIU.PAS

TBatchReport descends from TGenReport

Batch Receipt/Payment Report

Procedures - AddBatchPRep2Thread

# REPORTJU.PAS

TBankRReport descends from TGenReport

Auto G/L Reconciliation Matching

Manual Reconciliation Matching

Procedures - AddBankRecRep2Thread

# REPORTKU.PAS

TVATRReport2 descends from TVATRReport (Report8U.Pas)

VAT Return Audit Trail Report???

Procedures - AddVATRRep22Thread

# ReportRV.pas

TRVReport descends from TGenReport

Daily/Company Rate Currency Revaluation Report

Procedures - AddRVRep2Thread

# REPORTWU.PAS

TWOR1Report descends from TGenReport

Works Order Fulfilment Report

Works Order Status Report

Work In Progress Reconciliation Report

Works Order Explosion Report

Procedures - AddWOP1Rep2Thread / TESTWOR1Rep

# RepRet1U.Pas

TRet1Report descends from TGenReport

PRN/SRN Analysis Report

Procedures AddRet1Rep2Thread / TESTRET1Rep

# RepAlcU.Pas

TAllocReport descends from TGenReport

Purchase/Sales Allocation Report for XXX

Unallocation Matching Report for XXX

CIS Matching Report ???

Procedures - AddAllocRep2Thread

# RepBnskU.Pas

TBin1Report descends from TGenReport

Stock Availability by Bin Code Report

Bin Stock Availability by Stock Code Report

Stock by Bin Code History Report

Bin by Stock Code History Report

Procedures - AddBIN1Rep2Thread

# RepBSupU.Pas

TISVATReport descends from TGenReport

Intra EC Trade Statistics

Procedures - AddISVATRep3Thread

# RepCCShu.Pas

**TQCCDepReport** descends from TThreadQueue

Procedures - AddCCDepReportThread

**TQCCDepTBReport** descends from TQCCDepReport

Procedures AddTBCCDepReportThread

**TQCCDepCDReport** descends from TQCCDepTBReport

Procedures - AddCDCCDepReportThread

# RepJCA1U.Pas

TJCActualReport descends from TJCXposeReport (RepJCX1U.Pas)

Job Detailed Analysis Report (Mode = 1/2)

Procedures - AddJCAnalRep2Thread

# RepJCA2U.Pas

TJCApps1Report descends from TGenReport

Job Costing JPA’s/JSA’s Report

Job Costing JPA’s/JSA’s Due Report

Procedures - AddJCAppsRep2Thread

# RepJCB1U.Pas

TJBudgReport descends from TGenReport

Job Budget Analysis

Procedures - AddJBudgRep2Thread

# RepJCB2U.Pas

TJBillReport descends from TGenReport

Job Billing Report

Procedures - AddJBillRep2Thread

# RepJCE1U.Pas

TJCEListReport descends from TGenReport

Employee Report

Global Time Rates Report

Procedures - AddEmpListRep2Thread

# RepJCE2U.Pas

TJCCISReport1 descends from TGenReport

Sub-Contract Employee Report

Global Time Rates Report

Sub-Contract Continuous Working Period Report

Something to do with Forms???

Detailed or RCT47/RCT48/RCT30 Contractor's Return (See RepJCE3U.Pas)

Procedures - AddCISListRep2Thread

# RepJCE3U.Pas

TJCCISReport2 descends from TJCCISReport1

Detailed or RCT47/RCT48/RCT30 Contractor's Return (See RepJCE2U.Pas)

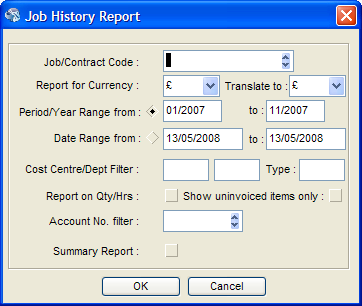
Procedures - AddCISListRep3Thread

# RepJCH1U.Pas

TJHistReport descends from TGenReport

Job History Report (mode = 2) – runs through the Job Actuals records in the Job Details table.

Procedures - AddJHistRep2Thread – called from the TRepJCHInp form (RepIJCHU.Pas):-



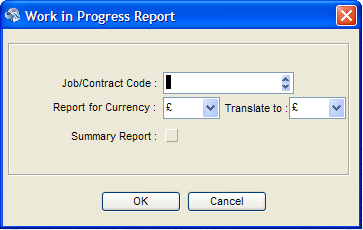
# RepJCW1U.Pas

TJWIPReport descends from TGenReport

Work in Progress Report (mode = 2)

Work In Progress Summary Report (mode = 1)

Procedures - AddJWIPRep2Thread



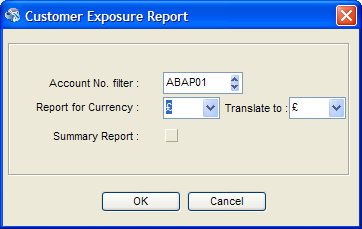
# RepJCX1U.Pas

TJCXposeReport descends from TGenReport

Job Costing. Customer Exposure Report/Summary (mode = 3/4)

NOTE: TJCActualReport (RepJCA1U.Pas) descends from this (mode = 1/2)

Procedures - AddJCxposeRep2Thread



# Customer/Supplier List (ReportU.Pas)



Inefficient – uses Idx 0 (acCode) so it has to process all records in Account Code order until EOF, this means that it is processing Customers and Suppliers. If Idx 5 (acCustSupp + acCode) was used instead then we could jump in at the correct block (C or S) and stop as soon as the block was finished – all records would be hits. (NOTE: Report pre-dates Idx 5).

If we rewrote this to use a pre-fill cache we could also limit the columns being returned, but this report is probably too basic to justify that much work initially.